**Введение**

На курсовое проектирование была поставлена задача разработать программный продукт для автоматизированного рабочего места диспетчера производственного кооператива имени В.И. Кремко. Цель данного проекта заключается в создании веб-приложения, предназначенного для оптимизации рабочего процесса диспетчера. Разрабатываемый ресурс предоставляет возможность эффективного взаимодействия с данными, обеспечивает оперативный мониторинг.

Данное веб-приложение рассчитано на широкий круг пользователей, включая диспетчеров, ответственных за планирование и распределение задач, а также исполнителей, выполняющих эти задачи.

Далее приведено краткое описание разделов пояснительной записки.

Первый раздел «Анализ задачи» содержит описание предметной области и круга задач, которые должны быть автоматизированы, построения диаграммы вариантов использования и модели данных. Также раздел содержит обоснование выбора среды разработки и требования к приложению.

Второй раздел «Проектирование задачи» посвящен построению структуры приложения. Здесь описываются основные аспекты проектирования программного продукта: архитектура системы, организация данных, а также интерфейс и функциональные элементы приложения.

Третий раздел «Реализация» содержит структуру и описание функций пользователя и элементов управления, за которыми они закреплены.

Четвертый раздел называется «Тестирование». Данный раздел описывает тестирование программного продукта, список ошибок, выявленных при тестировании, а также содержит вывод о результатах тестирования.

Пятый раздел «Применение» содержит назначение и описание среды функционирования программы. В разделе описан процесс инсталляции, перечень сообщений системы, а также краткое руководство по эксплуатации.

«Заключение» подводит итоги выполненной работы, содержит краткую формулировку задачи, выводы о достижении поставленных целей и описание достигнутых результатов. Приводится анализ эффективности программного обеспечения.

В разделе «Список использованных источников» представлен перечень литературы, справочных материалов и источников, использованных при выполнении проекта.

В приложении к пояснительной записке содержатся тесты на использование и листинг программы, которые иллюстрируют архитектуру и основные аспекты реализации программного обеспечения.

**1 Анализ задачи**

**1.1 Описание предметной области**

Разрабатываемый программный продукт направлен на оптимизацию рабочего процесса диспетчера, управления задачами, их распределения, мониторинга статусов и взаимодействия с исполнителями.

Программный продукт будет использоваться ежедневно в течение рабочего дня. Исполнители будут использовать систему для получения задач и предоставления отчетов о выполнении. Период использования данной программы не ограничен.

В разработанной программе будут использоваться два вида данных.

Входными данными будут являться данные для поиска и фильтрации, включая управление задачами.

Постоянная информация – данные о сотрудниках.

Основные задачи для разработки программы:

– просмотр списков управления задачами;

– возможность редактирования названия задачи;

– удаление задачи;

– осуществление поиска задач по фильтру;

– осуществление поиска задач по имени или фамилии;

– фильтрация и сортировка списка задач;

**1.2 Диаграмма вариантов использования**

Диаграмма вариантов использования UML описывает функциональность и требования системы с помощью актеров (пользователей или других систем) и вариантов использования. Варианты использования представляют собой функциональные возможности системы и то, как актеры будут взаимодействовать с системой.

Диаграмма важна в процессе разработки программного обеспечения, обеспечивая визуальное представление взаимодействий между пользователями (или другими версиями) и программным продуктом. У нее есть несколько ключевых целей и преимуществ, которые делают ее незаменимой на различных этапах разработки. В данном приложении имеется два пользователя, которые взаимодействует с системой для выполнения различных задач. Диаграмма вариантов использования представлена на рисунке 1.

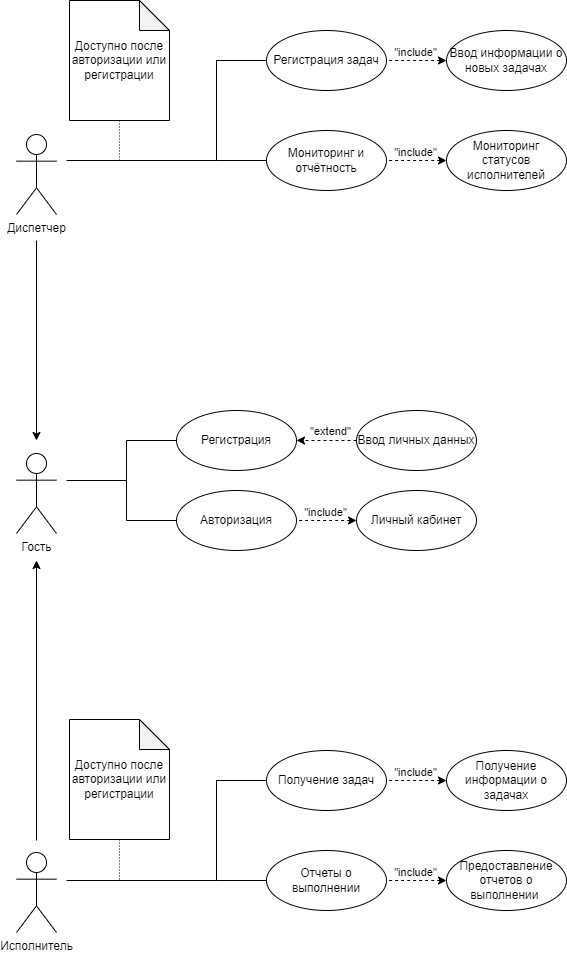


Рисунок 1 – Диаграмма вариантов использования

**1.3 Модель данных**

Цель моделирования данных состоит в определении концептуальной схемы совокупности данных в форме одной или нескольких локальных моделей, которые относительно легко могут быть отображены в любую форму представления данных.

Первый шаг моделирования – извлечение информации и выделение сущностей. Сущность – это реальный объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению в проекте, которая представлена на рисунке 2.



Рисунок 2 – Графическое изображение сущности

Применимо к разрабатываемой задаче можно выделить следующие сущности:

Таблица 1 – Описание сущностей

|  |  |
| --- | --- |
| Сущность | Описание |
| Пользователи | Содержит информацию о пользователях системы, включая диспетчеров и исполнителей |
| Задачи | Хранит данные о задачах, созданных диспетчерами, включая статус и исполнителя |
| Статус\_задачи | Определяет возможные статусы задач, такие как "в процессе", "завершено" и т.д. |
| Уведомления | Содержит уведомления, связанные с задачами, которые могут быть отправлены пользователям |

Следующим шагом в моделировании данных является идентификация связей. Связь – это ассоциация между двумя сущностями, значимая для рассматриваемой предметной области, которая представлена на рисунке 3.

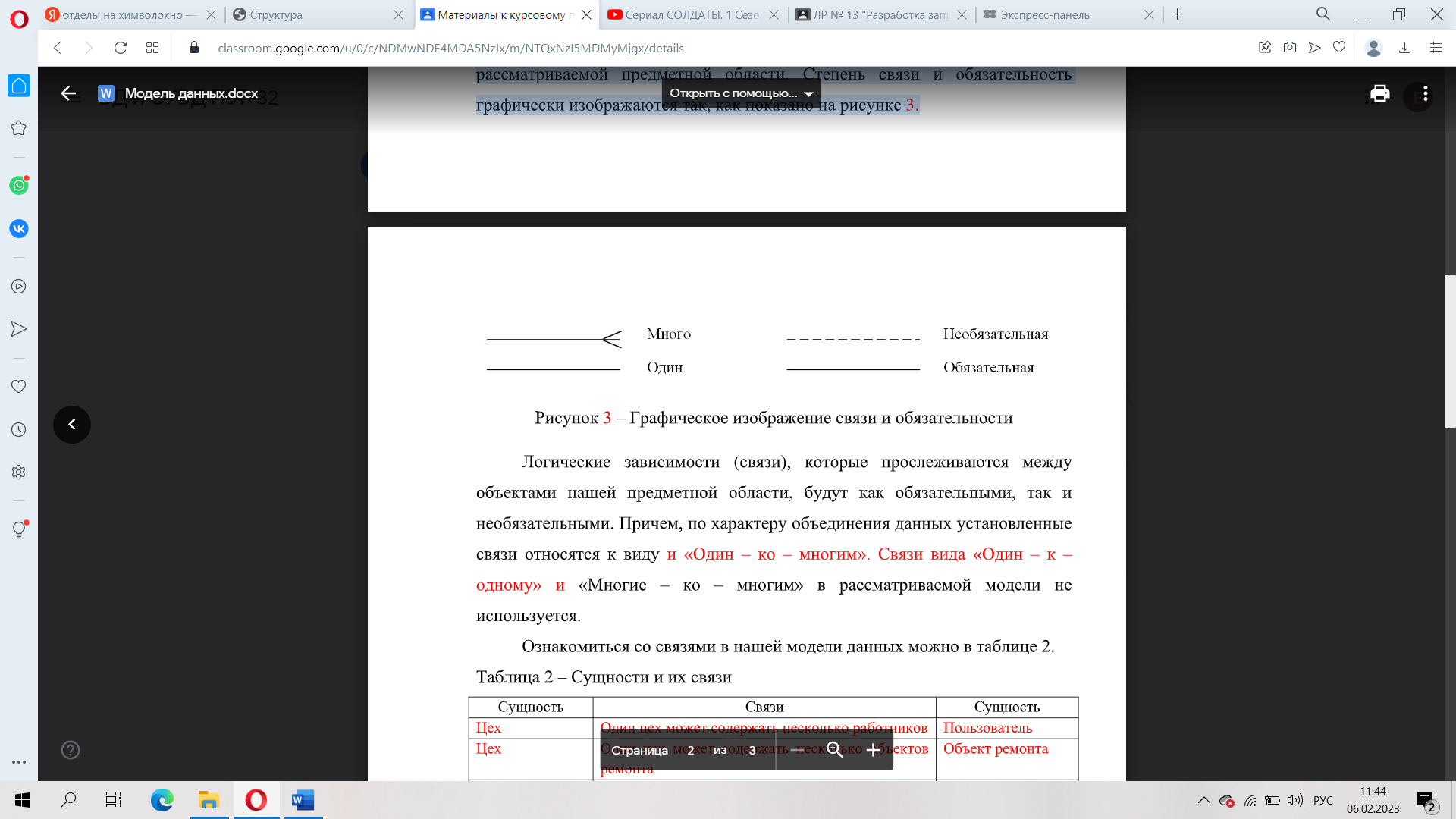


Рисунок 3 – Графическое изображение связи и обязательности

Логические зависимости (связи), которые прослеживаются между объектами нашей предметной области, будут как обязательными, так и необязательными. Причем, по характеру объединения данных установленные связи относятся к виду и «Один – ко – многим», связи вида «Один – к – одному». Связь «Многие–ко–многим» не используются.

Таблица 2 – Сущности и их связи

|  |  |  |
| --- | --- | --- |
| Сущность | Связи | Сущность |
| Пользователи | Один диспетчер может иметь много задач, но каждая задача принадлежит только одному диспетчеру | Задачи |
| Статус\_задач | Один статус может быть применен к множеству задач | Задачи |
| Задачи | Множество задач может быть назначено одному исполнителю | Исполнители |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Статус\_сотрудников | Каждый пользователь имеет уникальный статус | Пользователи |
| Задачи | Одна задача может иметь много уведомлений, и одно уведомление может относиться к нескольким задачам. | Уведомления |

Последним шагом моделирования является идентификация атрибутов.

Атрибут – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. В данном проекте все атрибуты таблиц являются обязательными.

Таблица 3 – Проведение тестов программы

|  |  |  |
| --- | --- | --- |
| Сущность | Атрибут | Описание атрибута |
| 1 | 2 | 3 |
| Пользователи | Код пользователя | Ключевое поле, обозначающее код пользователя |
| Имя | Имя пользователя |
| Фамилия | Фамилия пользователя |
| Роль | Роль пользователя в системе |
| Пароль | Пароль пользователя |
| Задачи | Код задачи | Ключевое поле, обозначающее код задачи |
| Название | Название задачи |
| Код статуса | Идентификатор статуса задачи |
| Код исполнителя | Идентификатор исполнителя, которому назначена задача |
| Статус задачи | Код статуса задачи | Ключевое поле, обозначающее код статуса задачи |
| Название статуса задачи | Название статуса задачи |
| Уведомления | Код уведомления | Ключевое поле, обозначающее код уведомления |
| Код задачи | Ключевое поле, обозначающее код задачи, к которой относиться уведомление |
| Сообщение | Текст уведомления |

Таким образом, при моделировании была получена модель данных, представленная в графической части.

**1.4 Инструменты разработки**

Для разработки программы будет использоваться интегрированная среда разработки Visual Studio Code.

Microsoft Visual Studio – данный программный продукт позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ поддерживаемых Windows.

VS Code включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встроенные инструменты включают в себя интегрированный терминал, инструменты для работы с базами данных и широкий выбор расширений, доступных в Marketplace. Visual Studio Code является кроссплатформенной средой разработки, доступной для Windows, Linux, и широко используется разработчиками для создания программного обеспечения различных типов.

База данных будет разработана с помощью программного обеспечения MySQL Workbench. MySQL Workbench – это приложение для управления реляционными базами данных MySQ. Данный инструмент предоставляет широкий спектр возможностей, включая выполнение SQL-запросов, управление таблицами, создание и редактирование записей, настройку связей между таблицами, импорт и экспорт данных в различных форматах.

Разработка диаграмм будет осуществляется с помощью программы Draw.io.

При разработке проектной документации будет использоваться Microsoft Word.

Разработка проекта будет происходить на компьютере со следующими параметрами:

– процессор Ryzen 5 5600H;

– объем ОЗУ 16 GB;

– объем места на SSD: 512 GB;

– ОС: Windows 11 home.

**1.5 Требования к приложению**

На этапе исследования предметной области был установлен целый ряд требований, которые предъявляются к разрабатываемой программе:

– быстрая работа программы;

– добавление, редактирование и удаление информации о задачах;

– редактирование таблиц базы данных;

– поиск сотрудников по заданным параметрам;

– учёт и отображение информации о задачах;

– обработка ошибок в программе;

– корректное отображение таблиц данных.

Предъявлены следующие требования к интерфейсу:в связи с частым использованием веб-приложения она должна быть приятной, в какой-то степени яркой, цветовой гаммой.

Следовательно, каждое окно должно иметь ясную визуальную иерархию своих элементов. Фрагменты текста должны располагаться на экране так, чтобы пользователю было понятно принимать информацию. На одной форме нельзя допускать избытка и нагромождения данных. Формы должны быть эффектно оформлены согласно тематике разрабатываемого проекта.

**2 Проектирование задачи**

**2.1 Организация данных**

Важным условием при разработке данного проекта является описание организации данных, то есть логическая и физическая структуры данных в контексте среды обработки.

Физическая модель данных может отличаться от логической, она определяет, каким образом предоставляются данные, и содержит все детали, необходимые СУБД для создания базы данных. В данном случае они совпадают.

База данных будет включать в себя пять таблиц и разработана с помощью MySQL Workbench. Подробное описание таблиц представлено в таблице 4.

Таблица 4 – Описание полей таблиц базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование таблицы | Наименование поля | Описание | Тип поля |
| 1 | 2 | 3 | 4 |
| Task\_status | id | Ключевое поле, обозначающее код статуса | Целочисленный |
| name | Название статуса | Текстовый |
| Users | id | Ключевое поле, обозначающее код пользователя | Целочисленный |
| Name | Имя пользователя | Текстовый |
| SurName | Фамилия пользователя | Текстовый |
| Email | Электронная почта пользователя (уникальная) | Числовой |
| Password | Пароль пользователя | Дата |
| Role | Роль пользователя (диспетчер или исполнитель) | Перечисление |
| Created\_at | Дата и время создания записи | Текстовый |
| Updated\_at | Дата и время последнего обновления записи | Целочисленный |
| Tasks | id | Ключевое поле, обозначающее код задачи | Целочисленный |
| Title | Название задачи | Текстовый |
| Description | Описание задачи | Вещественный |
| address | Адрес выполнения | Вещественный |
| Deadline | Срок выполнения | Дата |
| Status\_id | Ссылка на статус задачи | Целочисленный |
| Dispatcher\_id | Ссылка на диспетчера | Целочисленный |
| Executor\_id | Ссылка на исполнителя | Целочисленный |
| Created\_at | Дата и время создания записи | Дата |
| Updated\_at | Дата и время последнего обновления записи | Дата |
|  | | | |
| Продолжение таблицы 4 | | | |
| 1 | 2 | 3 | 4 |
| Notifications | id | Ключевое поле, обозначающее код уведомления | Целочисленный |
| User\_id | Получатель уведомления | Текстовый |
| message | Текст уведомления | Текстовый |
| Is\_read | Прочитано или нет | Целочисленный |
| Created\_at | Дата и время создания уведомления | Целочисленный |
| Employee\_status | id | Ключевое поле, обозначающее код статуса | Целочисленный |
| Status\_name | Название статуса | Текстовый |
| Description | Описание | Текстовый |
| Created\_at | Дата создания | Дата |

**2.2 Процессы**

Разрабатываемый продукт представляет собой веб-приложение, предназначенное для использования сотрудниками организации.

Пользователю должны быть предоставлены следующие функции:

– добавление данных в таблицы;

– редактирование данных в таблицах;

– удаление данных из таблиц;

– поиск сотрудников по введенным данным;

– сортировка данных;

– фильтрация данных;

**2.3 Описание внешнего пользовательского интерфейса**

Организация диалога между пользователем и программой является важным условием при проектировании программного продукта.

Внешний вид программы должен быть организован так, чтобы пользователь без затруднений мог выполнять работу с данными. Программа должна иметь структурированную навигацию и элементы управления не должны перекрывать друг друга, а цвета фонов не должны быть слишком яркие.

При открытии веб-приложения пользователь сразу увидит форму авторизации. После успешного входа он попадает на главную страницу, где находится меню для навигации по разделам, такие как «Статус работников», «Добавить задачу», «Админ панель», «Просмотр задач» и кнопка «Выйти» после нажатия на которую происходит выход из аккаунта пользователя.

**Раздел «Статус работников»** – позволяет диспетчеру просматривать информацию о текущем статусе исполнителей и их задачах. Она отображает список исполнителей с указанием их имен, фамилий и связанных с ними задач, а также статусов этих задач (например, "выполняется", "завершена"). Это помогает диспетчеру отслеживать прогресс выполнения задач и управлять рабочими процессами.

Раздел «Добавить задачу» – позволяет диспетчеру создавать новые задачи для исполнителей. В этой вкладке диспетчер может ввести информацию о задаче, такую как название, описание, адрес выполнения и срок выполнения. Также необходимо указать исполнителя, которому будет назначена задача. После заполнения всех необходимых полей диспетчер может отправить форму, чтобы сохранить новую задачу в системе. Это упрощает процесс управления задачами и распределения работы среди исполнителей.

Раздел «Админ панель» – позволяет диспетчеру управлять задачами и пользователями. Здесь можно просматривать, редактировать и добавлять задачи, а также управлять пользователями системы. Это центральное место для мониторинга и управления рабочими процессами.

Раздел «Просмотр задач» – позволяет исполнителям видеть список всех назначенных им задач. Здесь отображаются названия задач, информация о диспетчере и текущий статус каждой задачи. Исполнители могут изменять статус задач, что помогает отслеживать прогресс выполнения работы.

**2.4 Средства защиты и разграничения доступа к данным**

При организации доступа к веб-приложению существенное значение имеет аутентификация пользователей. Аутентификация – процедура проверки подлинности, что является средством защиты данных.

В данном программном продукте реализована аутентификация, то есть при загрузке веб-приложения, у пользователя будет запрашиваться пароль, при правильном вводе которого будет осуществлен вход в систему.

**2.5 Тесты на использование**

Для того чтобы проверить корректность работы всех элементов программы необходимо разработать тест-кейсы для приёмочного тестирования и привести отчёт о дефектах, если таковые будут найдены. Тесты составлены таким образом, чтобы предусмотреть максимум возможных действий пользователя. Тесты на использование представлены в приложении А.

**3 Реализация**

**3.1 Функции: логическая и физическая организации**

В ходе разработки программного продукта было написано множество процедур и функций. Их описание приводится в таблице 5.

Таблица 5 – Функции программы

|  |  |  |
| --- | --- | --- |
| Название функции | Модуль | Назначение |
| 1 | 2 | 3 |
| session\_start | php/logout.php | Начало сессии |
| require | php/workers.php | Подключение конфигурации |
| header | php/workers.php | Перенаправление на страницу входа |
| execute | php/tasks.php | Выполнение SQL-запроса |
| prepare | php/config.php | Подготовка SQL-запроса |
| fetchAll | php/tasks.php | Получение всех задач |
| file\_get\_contents | php/tasks.php | Чтение содержимого шаблона |
| str\_replace | php/admin.php | Замена плейсхолдеров в шаблоне |
| htmlspecialchars | php/admin.php | Экранирование специальных символов |
| eval | php/admin.php | Выполнение PHP-кода из строки |
| json\_encode | php/admin.php | Кодирование данных в JSON |
| preg\_replace | php/workers.php | Замена шаблонов в строке |
| setAttribute | php/config.php | Установка атрибутов PDO |
| catch | php/config.php | Обработка исключений |
| die | php/config.php | Завершение скрипта с сообщением |
| fetchAll | php/admin.php | Получение всех статусов |
| htmlspecialchars | templates/admin\_template.html | Экранирование данных для HTML |
| echo | templates/workers\_template.html | Вывод данных на страницу |
| isset | php/tasks.php | Проверка существования переменной |
| exit | php/logout.php | Завершение выполнения скрипта |
| join | php/admin.php | Объединение таблиц в SQL-запросе |
| order by | php/admin.php | Сортировка результатов |
| session\_unset | php/logout.php | Удаление всех данных сессии |
| session\_destroy | php/logout.php | Уничтожение сессии |
| htmlspecialchars | templates/add\_task\_template.html | Экранирование данных для HTML |
| preg\_replace | php/tasks.php | Замена шаблонов в строке |
| header | php/tasks.php | Перенаправление на страницу задач |
| prepare | php/tasks.php | Подготовка SQL-запроса |
| execute | php/tasks.php | Выполнение SQL-запроса |
| fetchAll | php/tasks.php | Получение всех задач |
| file\_get\_contents | php/tasks.php | Чтение содержимого шаблона |
|  | | |
| Продолжение таблицы 5 | | |
| 1 | 2 | 3 |
| str\_replace | php/admin.php | Замена плейсхолдеров в шаблоне |
| htmlspecialchars | php/admin.php | Экранирование специальных символов |
| eval | php/admin.php | Выполнение PHP-кода из строки |
| json\_encode | php/admin.php | Кодирование данных в JSON |
| preg\_replace | php/workers.php | Замена шаблонов в строке |
| setAttribute | php/config.php | Установка атрибутов PDO |
| catch | php/config.php | Обработка исключений |
| die | php/config.php | Завершение скрипта с сообщением |
| fetchAll | php/admin.php | Получение всех статусов |
| htmlspecialchars | templates/admin\_template.html | Экранирование данных для HTML |
| echo | templates/workers\_template.html | Вывод данных на страниц |

**3.2 Функции и элементы управления**

Для того чтобы пользователь мог работать с веб-приложением необходимо разработать пользовательский интерфейс. Написанные функции закреплены за элементами управления. Они перечислены в таблице 6.

Таблица 6 – Элементы управления

|  |  |  |
| --- | --- | --- |
| Название элемента управления | Форма | Описание |
| 1 | 2 | 3 |
| select | templates/admin\_template.html | Фильтр по статусу задач |
| input type="text" | templates/admin\_template.html | Поиск по фамилии исполнителя |
| button type="submit" | templates/admin\_template.html | Кнопка для поиска исполнителя |
| select | templates/tasks\_template.html | Выбор статуса задачи |
| input type="hidden" | templates/tasks\_template.html | Скрытое поле для ID задачи |
| button type="submit" | templates/tasks\_template.html | Кнопка для изменения статуса задачи |
| input type="text" | templates/add\_task\_template.html | Поле для имени исполнителя |
| input type="text" | templates/add\_task\_template.html | Поле для фамилии исполнителя |
| textarea | templates/add\_task\_template.html | Поле для описания задачи |
| button type="submit" | templates/add\_task\_template.html | Кнопка для отправки формы добавления задачи |

**4 Тестирование**

Одним из важнейших этапов в процессе разработки программного обеспечения является тестирование. Оно проводится с целью выявления недочетов и ошибок, допущенных при разработке, а также их устранения. В таблице 7 описана команда тестировщиков.

Таблица 7 – Команда тестировщиков

|  |  |  |
| --- | --- | --- |
| ФИО | Должность | Роль |
| Шапский Алексей Сергеевич | Тестировщик, разработчик | Ответственный за обеспечение качества |

Расписание проведения тестирования представлено в таблице 8.

Таблица 8 – Расписание проведения тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| ФИО | Дата | Описание | Длительность, ч |
| Шапский Алексей Сергеевич | 19.02.2025 | Создание тест-кейсов | 4 |
| Шапский Алексей Сергеевич | 20.02.2025 | Выполнение тест-кейсов | 6 |
| Шапский Алексей Сергеевич | 21.02.2025 | Анализ выполнения тест-кейсов | 2 |
| Шапский Алексей Сергеевич | 21.02.2025 | Исправление найденных ошибок | 3 |
| Шапский Алексей Сергеевич | 22.02.2025 | Повторное выполнение тест-кейсов | 6 |

Выявленные по результатам тестирования ошибки представлены в таблице 9.

Таблица 9 – Статистика по выявленным ошибкам

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Статус | Количество ошибок | Важность | | | |
| Низкая | Средняя | Высокая | Критическая |
| Исправлено | 3 | 1 | 1 | 1 | 0 |
| Проверено | 5 | 2 | 2 | 1 | 0 |
| Открыто заново | 0 | 0 | 0 | 0 | 0 |
| Найдено | 3 | 2 | 0 | 1 | 0 |
| Отклонено | 0 | 0 | 0 | 0 | 0 |

Таким образом после проведения тестирования, были выявлены следующие ошибки:

– после входа не проверялась роль пользователя;

– после выбора фильтра убиралось поле с фамилией;

–при нажатии на кнопку «Сохранить изменения» данные не обновлялись и некорректно отображались.

Все найденные ошибки были устранены, соответственно, программный продукт характеризуется как реализованный и готовый к работе.

**5 Применение**

**5.1 Назначение**

Разработанный программный продукт называется «Учет сотрудников на предприятии». Веб-приложение предназначено для ведения учета сотрудников на предприятии, включая добавление, редактирование, удаление данных, а также поиск и формирование отчетов. Периодичность использования зависит от кадровых процессов и необходимости обновления информации.

**5.2 Среда функционирования**

После проведения требуемых тестов было выявлено, что минимальная конфигурация компьютера для запуска и функционирования программы без каких-либо проблем:

– операционная система: Windows 7 и выше;

– процессор: Intel® Pentium® CPU N4200 @ 1.10GHz;

– жесткий диск с объемом 128Гб;

– ОЗУ: 2,00Гб;

– Microsoft Office;

– клавиатура;

– компьютерная мышь;

**5.3 Сообщения системы**

В программном продукте предусмотрена проверка вводимых данных на корректность. Если какое-либо поле остается пустым, система отображает сообщение, которое представлено на рисунке 4, 5, 6.

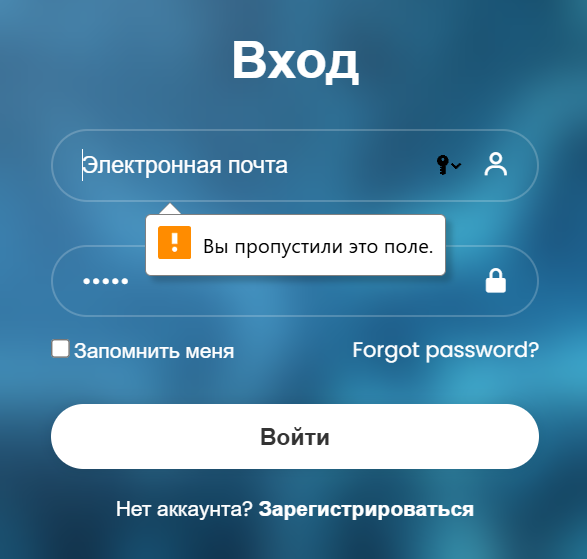


Рисунок 4 – Сообщение системы при незаполненных полях

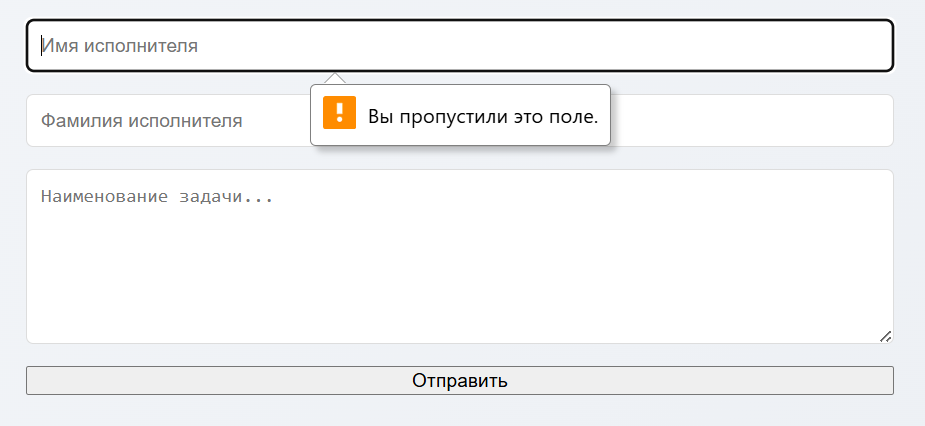


Рисунок 5 – Сообщение системы при незаполненных полях

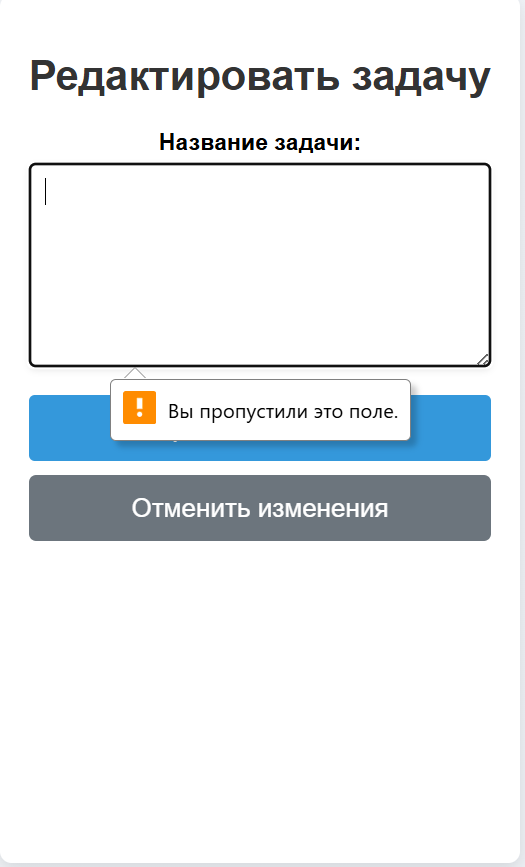


Рисунок 6 – Сообщение системы при незаполненных полях

**5.4 Обучение**

При открытии веб-приложения пользователь сначала проходит процедуру авторизации, при которой вводит почту и пароль. Для входа следует ввести почта «alexgudkov2000@gmail.com» и пароль «12345» и нажать на кнопку «Войти». Авторизация представлена на рисунке 7.

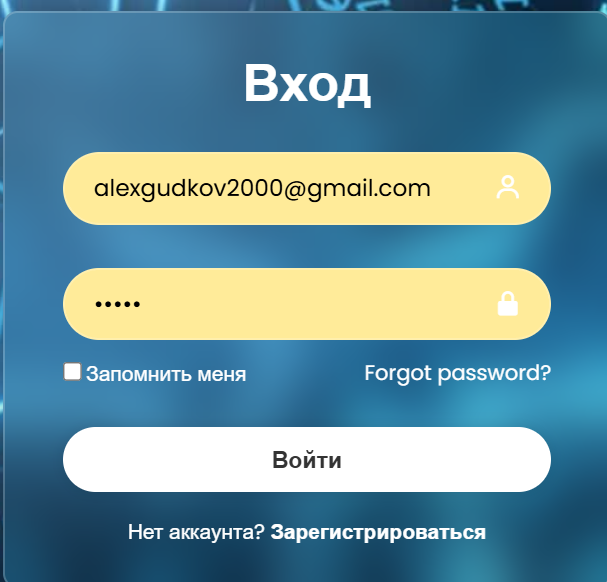


Рисунок 7 – Авторизация

После успешной авторизации пользователю открывается главное меню, представленное на рисунке 8.

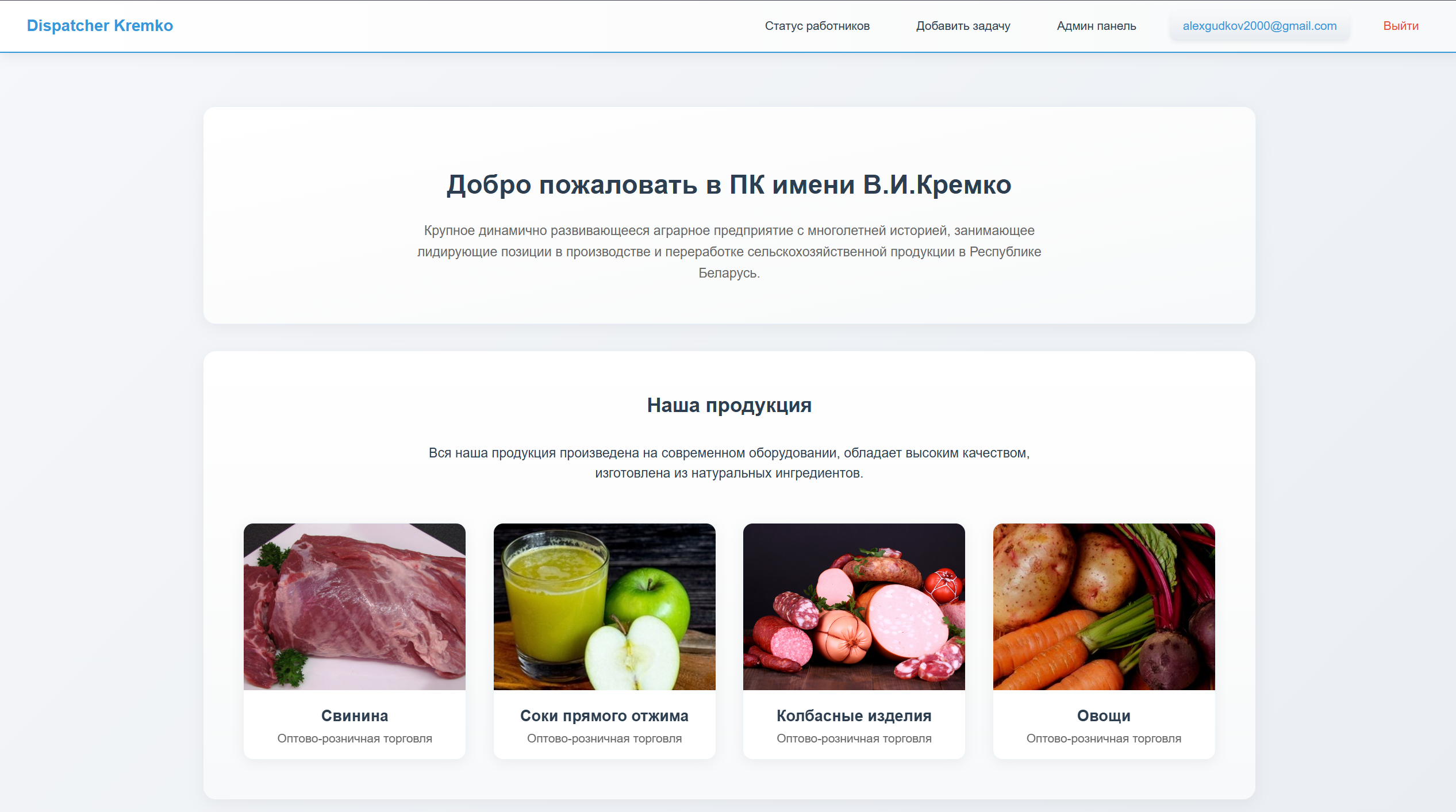


Рисунок 8 – Главное меню

Далее, при нажатии на кнопку **«Статус работника»**, открывается раздел для просмотра задач работников. Представленной на рисунке 9.

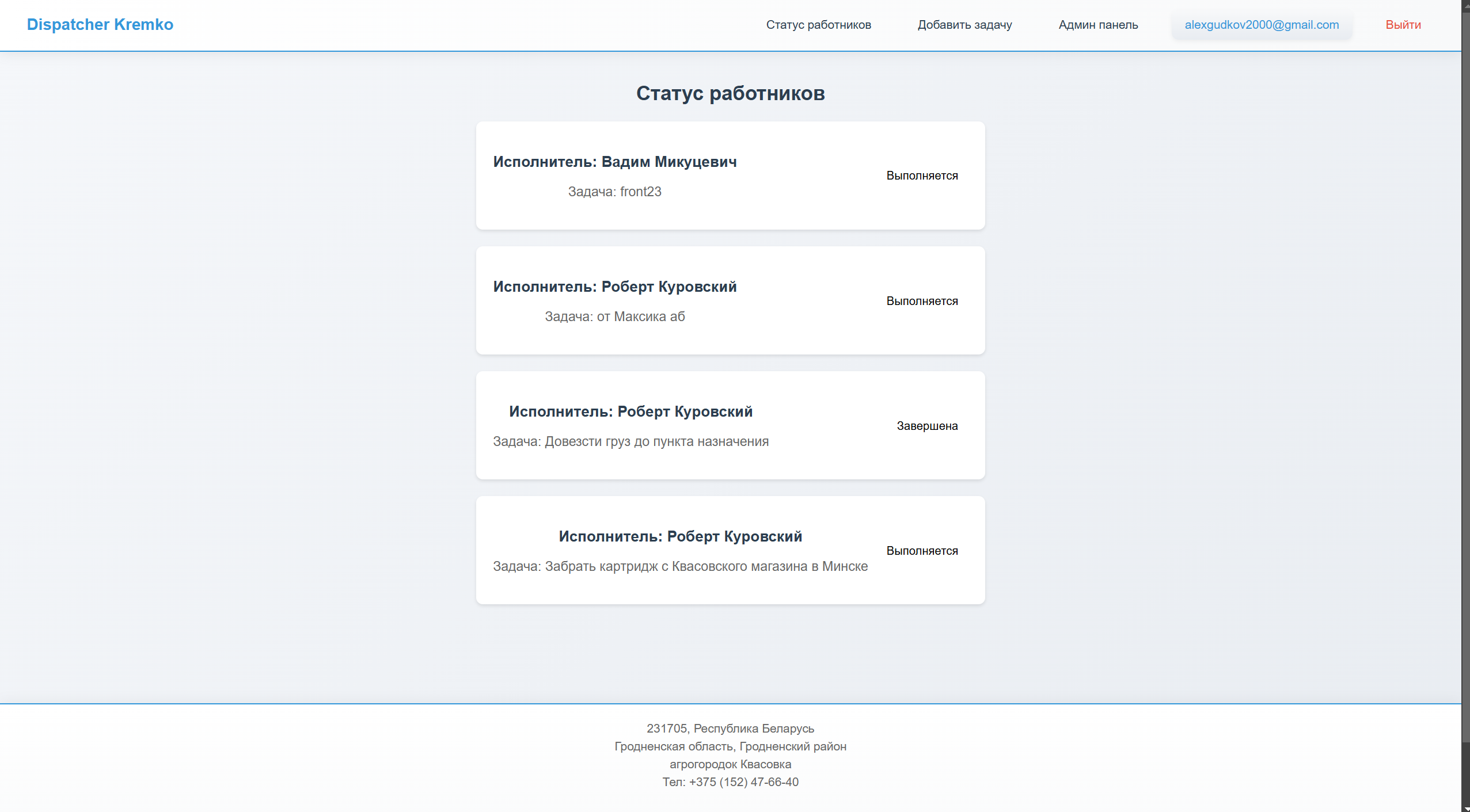


Рисунок 9 – Страница «**Статус работника»**

Ниже расположены формы с именами исполнителей, их задач и статус. Представлено на рисунке 10.

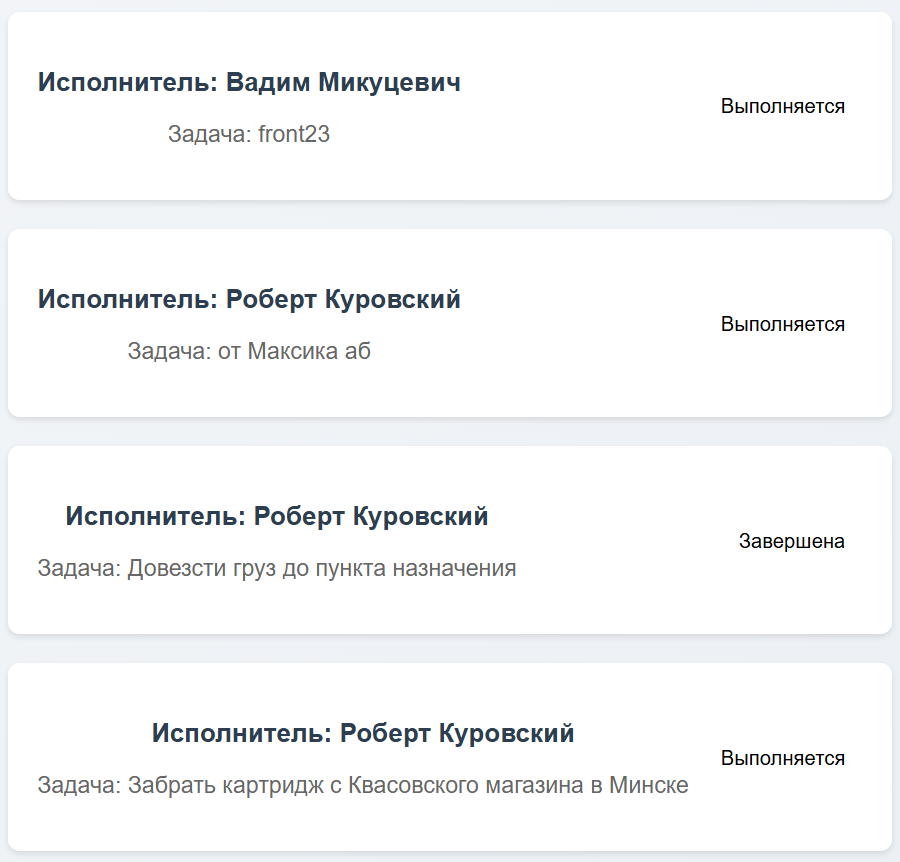


Рисунок 10 – Формы с задачами

Далее нажав на раздел «Добавить задачу», открывается страница для добавления задачи, после ввода данных и нажатия кнопки «отправить», задача переходит в раздел «Статус работника», представлен на рисунке 11 и 12.

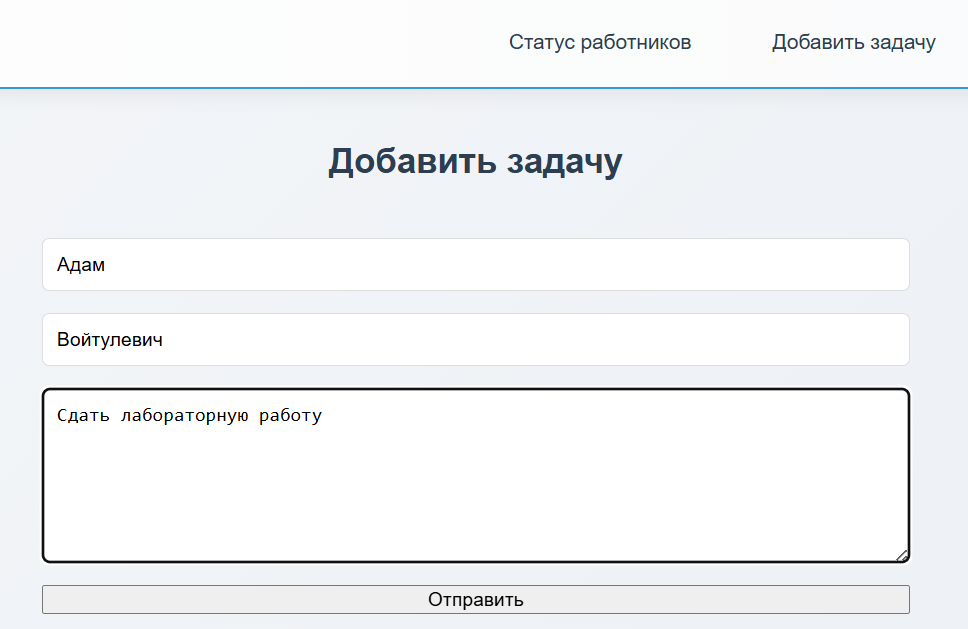


Рисунок 11 – Добавление задачи

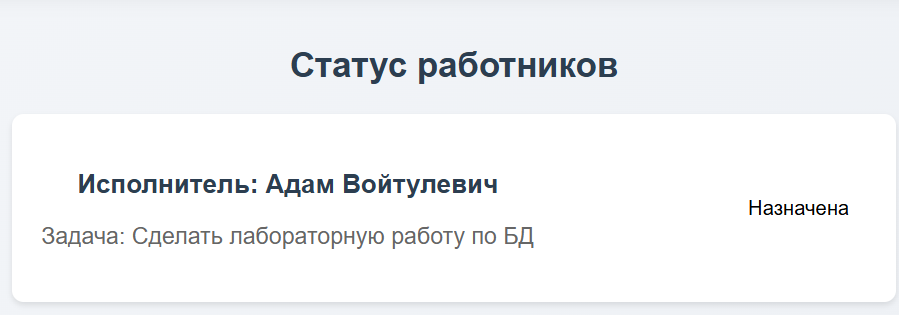


Рисунок 12 – Добавленная задача

Нажав на кнопку «Выйти», осуществляется выход с аккаунта пользователя, представлено на рисунке 13

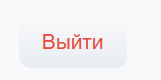


Рисунок 13 – Кнопка «Выйти»

Также при входе на сайт, можно войти как администратор, введя определенные данные, далее на главной странице появляется новый раздел «Админ панель» представлено на рисунке 14.

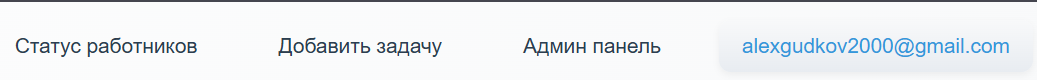


Рисунок 14 – Раздел «Админ панель»

В разделе «Админ панель» ниже есть форма для управления задачами, где администратор может просматривать все задачи, фильтровать их по статусу, сортировать задачи по названию (по убыванию или возрастанию), искать по фамилии работника, редактировать и удалять задачу. Представлено на рисунке 15.

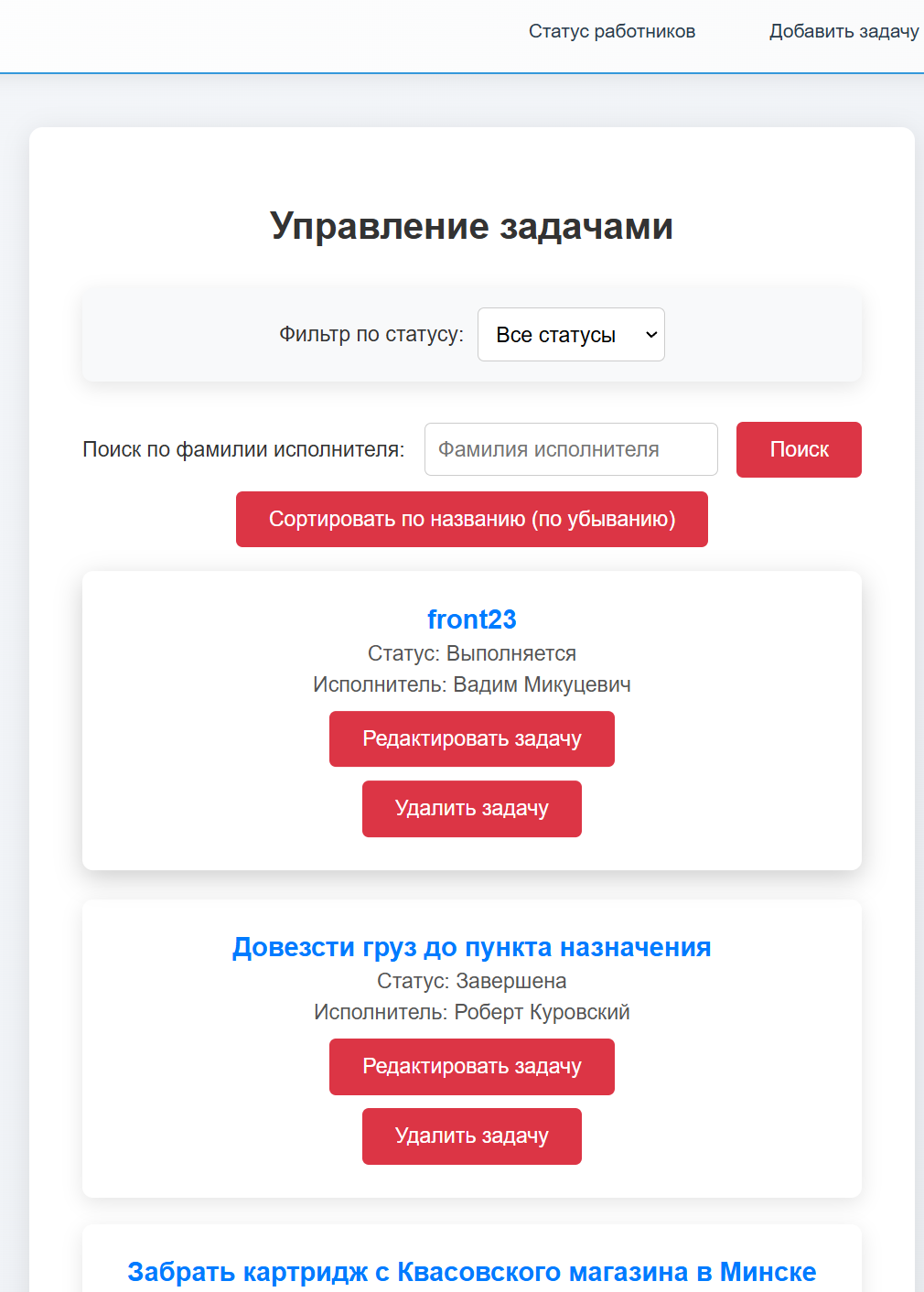


Рисунок 15 – Страница с формой «Админ панель»

При нажатии кнопки «Редактировать задачу», появляется форма с изменением названия задачи, представлено на рисунке 16.

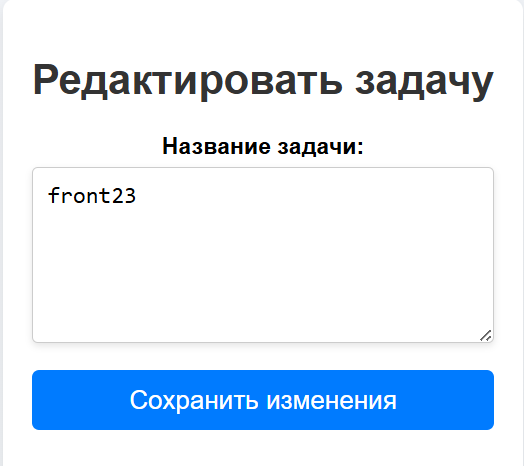


Рисунок 16 – Форма для редактирования задачи

При нажатии на кнопку «Удалить задачу», задача со всеми данными удаляется, представлена на рисунке 17.

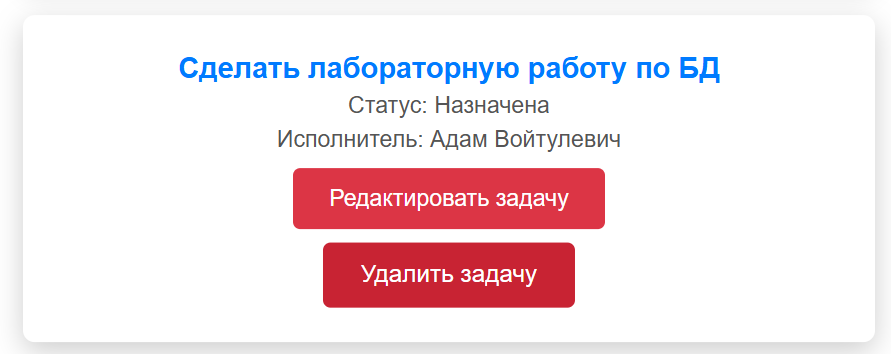


Рисунок 17 – Кнопка «Удалить задачу»

При выборе «Фильтр по статусу», можно отфильтровать задачи по статусу, представлено на рисунке 18.

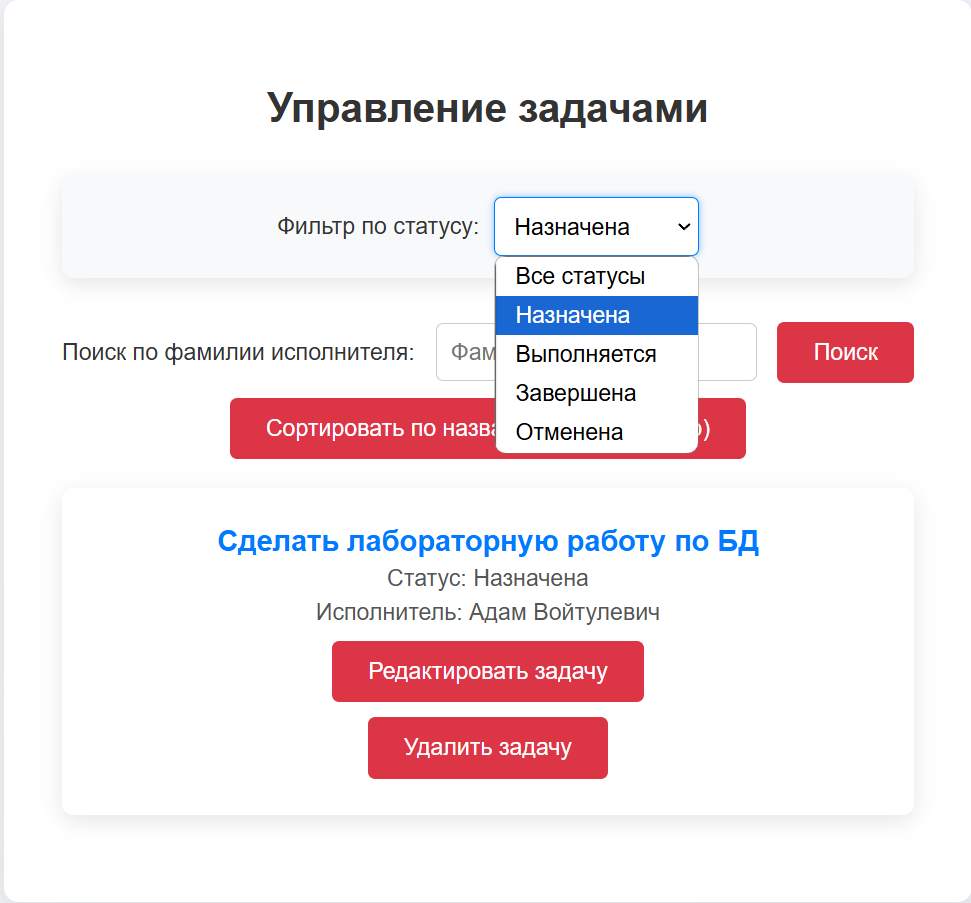


Рисунок 18 – Фильтр по статусу

При вводе фамилии в поле «Поиск по фамилии исполнителя», появляются только задачи определенного работника, представлена на рисунке 19.

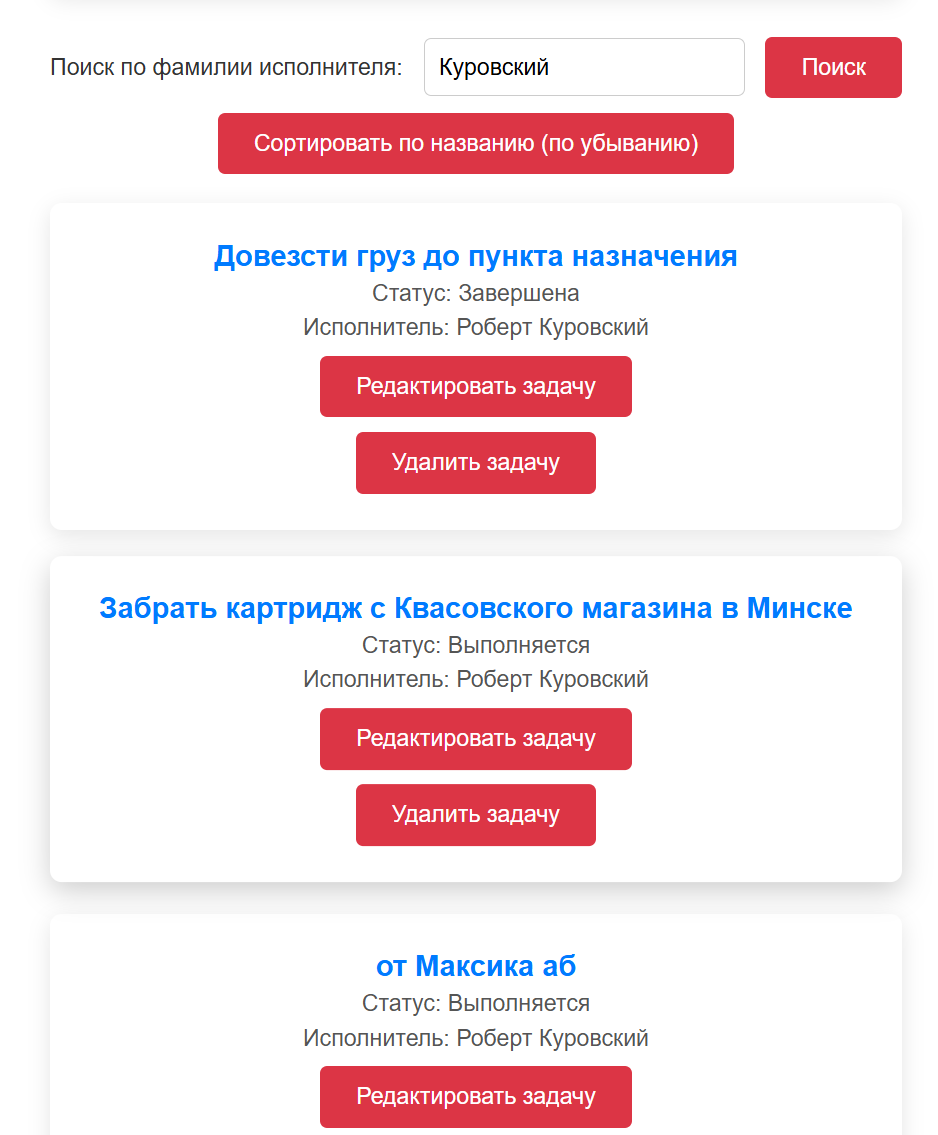


Рисунок 19 – Страница отпусков сотрудников

При нажатии на «Сортировать по названию (по возрастанию)» список задач фильтруется в алфавитном порядке, представлено на рисунке 20.

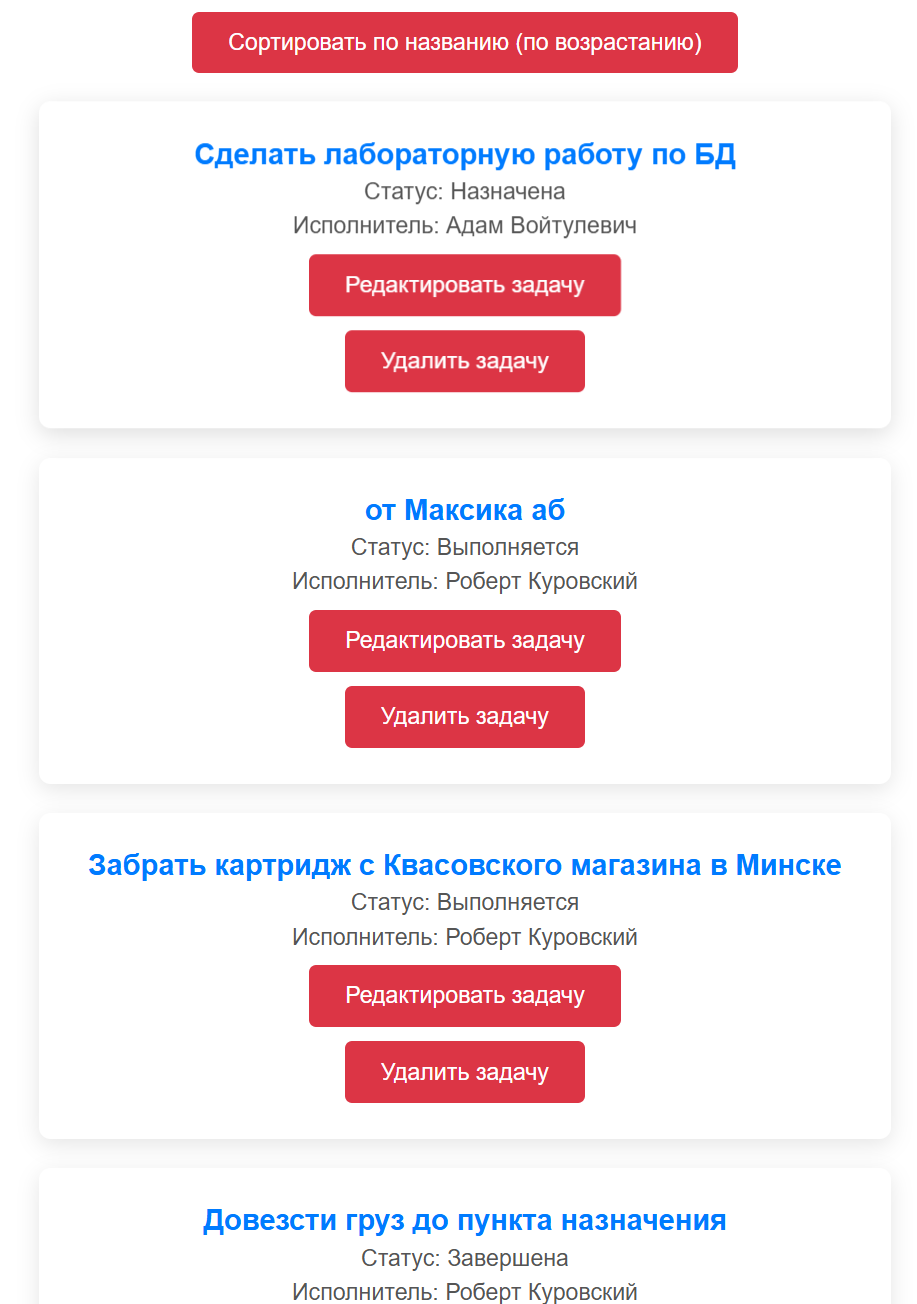


Рисунок 20 – Сортировка по названию (по возрастанию)

При нажатии на «Сортировать по названию (по возрастанию)» список задач фильтруется в алфавитном порядке, представлено на рисунке 21.

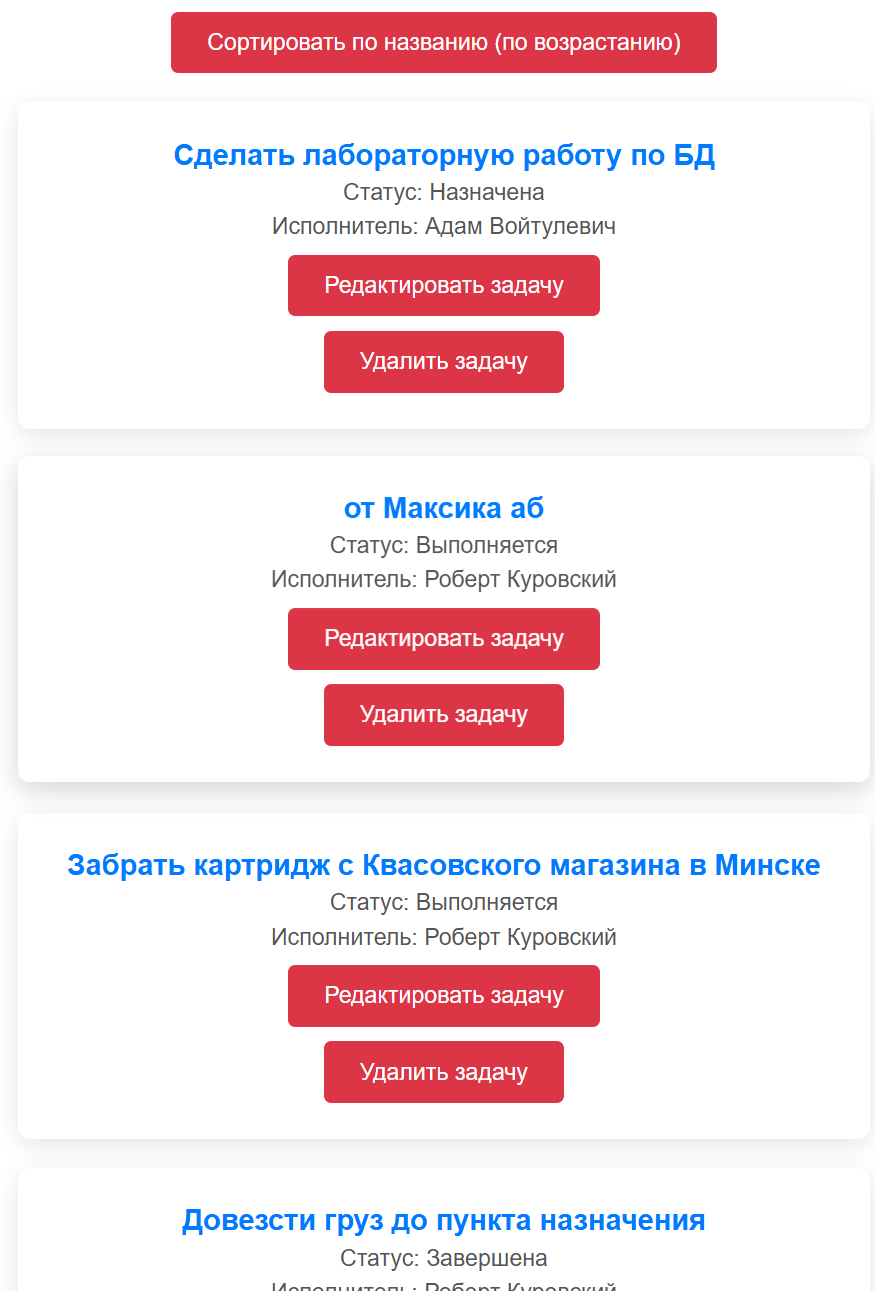


Рисунок 21 – Сортировка по названию (по убыванию)

**Заключение**

Целью курсового проекта было разработать программное обеспечение для автоматизированного рабочего места диспетчера производственного кооператива имени В.И. Кремко с целью оптимизации управления задачами.

В ходе разработки был реализован простой и понятный пользовательский интерфейс, обеспечивающий удобство работы с приложением.

При тестировании программного продукта все исключительные ситуации были обработаны. Проект работает без сбоев и корректно обрабатывает данные сотрудников.

Главную задачу курсового проектирования можно считать выполненной, так как разработанное приложение является логически завершённым и готовым к использованию в организации. В дальнейшем возможно внесение изменений и улучшений в функциональность. Все основные требования и пожелания, предъявляемые к проекту, были реализованы.

**Список использованных источников**

1. Крылов, А. (2022). Разработка веб-приложений с использованием PHP и MySQL. М.: БХВ-Петербург, 2022. – 300 с.
2. Конструктор диаграмм Draw.io [Электронный ресурс]. – Режим доступа: https://app.diagrams.net/. – Дата доступа 10.02.2025.
3. Руководство по MySQL. [Электронный ресурс]. – Режим доступа: https://metanit.com/sql/mysql/. – Дата доступа 11.02.2025.
4. Романова, И. (2021). Основы разработки веб-приложений с PHP и MySQL, 2021 – 320 с.
5. Форум программистов Cyberforum [Электронный ресурс]. – Режим доступа: <https://www.cyberforum.ru/>. – Дата доступа 19.02.2025.

<?php

session\_start();

require 'config.php';

if (!isset($\_SESSION['user']) || $\_SESSION['user']['role'] !== 'dispatcher') {

    header('Location: index.php');

    exit;

}

$user = $\_SESSION['user'];

$error = '';

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

    $executor\_name = $\_POST['executor\_name'];

    $executor\_surname = $\_POST['executor\_surname'];

    $task\_description = $\_POST['task\_description'];

    $stmt = $pdo->prepare("SELECT id, email FROM users WHERE name = ? AND surname = ? AND role = 'executor'");

    $stmt->execute([$executor\_name, $executor\_surname]);

    $executor = $stmt->fetch();

    if ($executor) {

        $stmt = $pdo->prepare("INSERT INTO tasks (title, description, executor\_id, dispatcher\_id, status\_id) VALUES (?, ?, ?, ?, 1)");

        if ($stmt->execute([$task\_description, $task\_description, $executor['id'], $user['id']])) {

            $subject = "Новая задача назначена";

            $message = "Вы получили новую задачу: $task\_description. Пожалуйста, проверьте вашу панель задач.";

            mail($executor['email'], $subject, $message);

            $notification\_stmt = $pdo->prepare("INSERT INTO notifications (user\_id, message) VALUES (?, ?)");

            $notification\_stmt->execute([$executor['id'], $message]);

            header('Location: home.php');

            exit;

        } else {

            $error = "Ошибка при добавлении задачи";

        }

    } else {

        $error = "Исполнитель не найден";

    }

}

$html = file\_get\_contents('../templates/add\_task\_template.html');

$html = str\_replace('{{error}}', htmlspecialchars($error), $html);

echo eval('?>' . preg\_replace('/\{\{(.\*?)\}\}/', '<?php echo $1; ?>', $html) . '<?php');

?>

<?php

session\_start();

require 'config.php';

if (!isset($\_SESSION['user']) || $\_SESSION['user']['role'] !== 'dispatcher') {

    header('Location: index.php');

    exit;

}

$user = $\_SESSION['user'];

$statusFilter = '';

$executorFilter = '';

$sortOrder = 'ASC'; *// По умолчанию сортировка по возрастанию*

if (isset($\_POST['status\_filter'])) {

    $statusFilter = $\_POST['status\_filter'];

}

if (isset($\_POST['executor\_filter'])) {

    $executorFilter = $\_POST['executor\_filter'];

}

if (isset($\_POST['sort\_order'])) {

    $sortOrder = $\_POST['sort\_order'] === 'ASC' ? 'ASC' : 'DESC'; *// Проверка на корректность*

}

$stmt = $pdo->prepare("

    SELECT t.\*, ts.name as status\_name, u.name as executor\_name, u.surname as executor\_surname

    FROM tasks t

    JOIN task\_status ts ON t.status\_id = ts.id

    LEFT JOIN users u ON t.executor\_id = u.id

    WHERE t.dispatcher\_id = ?

    AND (u.surname LIKE ?)

    AND (ts.id = ? OR ? = '')

    ORDER BY t.title $sortOrder

");

$stmt->execute([$user['id'], "%$executorFilter%", $statusFilter, $statusFilter]);

$statuses = $pdo->query("SELECT \* FROM task\_status")->fetchAll(PDO::FETCH\_ASSOC);

$tasks = $stmt->fetchAll(PDO::FETCH\_ASSOC);

$html = file\_get\_contents('../templates/admin\_template.html');

$html = str\_replace('{{tasks}}', json\_encode($tasks), $html);

$html = str\_replace('{{statuses}}', json\_encode($statuses), $html);

$html = str\_replace('{{executorFilter}}', htmlspecialchars($executorFilter), $html);

$html = str\_replace('{{statusFilter}}', htmlspecialchars($statusFilter), $html);

$html = str\_replace('{{sortOrder}}', $sortOrder, $html);

echo eval('?>' . preg\_replace('/\{\{(.\*?)\}\}/', '<?php echo $1; ?>', $html) . '<?php');

?>

<?php

session\_start();

require 'config.php';

if (!isset($\_SESSION['user']) || $\_SESSION['user']['role'] !== 'dispatcher') {

    header('Location: index.php');

    exit;

}

$user = $\_SESSION['user'];

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['task\_id'], $\_POST['task\_title'])) {

    $task\_id = $\_POST['task\_id'];

    $task\_title = $\_POST['task\_title'];

    $stmt = $pdo->prepare("UPDATE tasks SET title = ? WHERE id = ? AND dispatcher\_id = ?");

    $stmt->execute([$task\_title, $task\_id, $user['id']]);

    header('Location: admin.php');

    exit;

}

if (isset($\_GET['id'])) {

    $stmt = $pdo->prepare("SELECT \* FROM tasks WHERE id = ? AND dispatcher\_id = ?");

    $stmt->execute([$\_GET['id'], $user['id']]);

    $task = $stmt->fetch(PDO::FETCH\_ASSOC);

    if (!$task) {

        header('Location: admin.php');

        exit;

    }

} else {

    header('Location: admin.php');

    exit;

}

$html = file\_get\_contents('../templates/edit\_task\_template.html');

$html = str\_replace('{{task\_title}}', htmlspecialchars($task['title']), $html);

$html = str\_replace('{{task\_id}}', htmlspecialchars($task['id']), $html);

$html = str\_replace('{{user\_email}}', htmlspecialchars($user['email']), $html);

echo $html;

?>

<?php

session\_start();

if (!isset($\_SESSION['user'])) {

    header('Location: index.php');

    exit;

}

$user = $\_SESSION['user'];

$html = file\_get\_contents('../templates/home\_template.html');

$html = eval('?>' . preg\_replace('/\{\{(.\*?)\}\}/', '<?php echo $1; ?>', $html) . '<?php');

echo $html;

?>

<?php

require 'config.php';

$message = '';

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

    $email = $\_POST['email'];

    $password = $\_POST['password'];

*// Проверяем пользователя по email*

    $stmt = $pdo->prepare("SELECT \* FROM users WHERE email = ?");

    $stmt->execute([$email]);

    $user = $stmt->fetch();

*// Проверка пароля*

    if ($user && password\_verify($password, $user['password'])) {

        session\_start();

        $\_SESSION['user'] = $user;

        header('Location: home.php');

        exit;

    } else {

        $message = "Неверный email или пароль.";

    }

}

*// Передача сообщения об ошибке в шаблон*

$html = file\_get\_contents('../templates/login\_form.html');

$html = str\_replace('{{message}}', htmlspecialchars($message), $html);

echo $html;

?>

<?php

require 'config.php';

$message = '';

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

    $name = $\_POST['name'];

    $surname = $\_POST['surname'];

    $email = $\_POST['email'];

    $password = $\_POST['password'];

    $confirmPassword = $\_POST['password\_confirm'];

    $role = $\_POST['role'];

    $stmt = $pdo->prepare("SELECT \* FROM users WHERE email = ?");

    $stmt->execute([$email]);

    if ($stmt->rowCount() > 0) {

        $message = "Пользователь с таким email уже существует.";

    } else {

        if ($password !== $confirmPassword) {

            $message = "Пароли не совпадают.";

        } else {

            $hashedPassword = password\_hash($password, PASSWORD\_DEFAULT);

            $stmt = $pdo->prepare("INSERT INTO users (name, surname, email, password, role) VALUES (?, ?, ?, ?, ?)");

            if ($stmt->execute([$name, $surname, $email, $hashedPassword, $role])) {

                $userId = $pdo->lastInsertId();

                session\_start();

                $\_SESSION['user'] = [

                    'id' => $userId,

                    'name' => $name,

                    'surname' => $surname,

                    'email' => $email,

                    'role' => $role

                ];

                setcookie('user\_name', $name, time() + (86400 \* 30), "/"); *// 30 дней*

                setcookie('user\_surname', $surname, time() + (86400 \* 30), "/"); *// 30 дней*

*// Отладочное сообщение*

                error\_log("Пользователь зарегистрирован: ID = $userId");

                header('Location: home.php');

                exit;

            } else {

                $message = "Ошибка регистрации. Попробуйте снова.";

            }

        }

    }

}

*// Передача данных в шаблон*

$html = file\_get\_contents('../templates/register\_form.html');

$html = str\_replace('{{message}}', htmlspecialchars($message), $html);

echo $html;

?>

<?php

session\_start();

require 'config.php';

if (!isset($\_SESSION['user']) || $\_SESSION['user']['role'] !== 'executor') {

    header('Location: index.php');

    exit;

}

$user = $\_SESSION['user'];

*// Обновление статуса задачи, если пришел POST запрос*

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['task\_id'], $\_POST['status\_id'])) {

    $stmt = $pdo->prepare("UPDATE tasks SET status\_id = ? WHERE id = ? AND executor\_id = ?");

    $stmt->execute([$\_POST['status\_id'], $\_POST['task\_id'], $user['id']]);

    header('Location: tasks.php');

    exit;

}

*// Получаем задачи для текущего исполнителя*

$stmt = $pdo->prepare("

    SELECT t.\*, ts.name as status\_name,

           u.name as dispatcher\_name, u.surname as dispatcher\_surname

    FROM tasks t

    JOIN task\_status ts ON t.status\_id = ts.id

    JOIN users u ON t.dispatcher\_id = u.id

    WHERE t.executor\_id = ?

    ORDER BY t.created\_at DESC

");

$stmt->execute([$user['id']]);

$tasks = $stmt->fetchAll(PDO::FETCH\_ASSOC);

$stmt = $pdo->prepare("SELECT \* FROM task\_status");

$stmt->execute();

$statuses = $stmt->fetchAll(PDO::FETCH\_ASSOC);

$html = file\_get\_contents('../templates/tasks\_template.html');

echo eval('?>' . preg\_replace('/\{\{(.\*?)\}\}/', '<?php echo $1; ?>', $html) . '<?php');

?>