

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)**

Кафедра ПМ

Курсовая работа
защищена с оценкой

(подпись преподавателя, дата)

КУРСОВАЯ РАБОТА
по дисциплине «Проектирование программного обеспечения»
Вариант № 9
Тема: «Канбан-доска»

Выполнил студент группы ПМб-4-1
Фейзуллин К.М.
(Ф.И.О.)

Руководитель:

д.т.н., доцент, профессор каф. ПМ
Егорова А.А.
(звание, степень, Ф.И.О.)

МОСКВА – 2020

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра ПМ

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)

По дисциплине Проектирование программного обеспечения

Студента Фейзуллин Кирилл Маратович
(фамилия, имя, отчество)

Группы ПМб-4-1

1. Наименование темы «Канбан – доска»

2. Задание и основные характеристики:

Проектирование и разработка программного продукта, реализующего виртуальную канбан – доску, одного из инструментов управления в ходе разработки программного обеспечения. Работа комплексная, требуется выполнить:

- Реализовать блок для взаимодействия с карточками на форме;
- Реализовать блок для вывода карточек на форму, учитывая установленные фильтры;

3. Объем:

пояснительная записка на 16 листах,

приложения на 55 листах.

Руководитель: профессор Егорова А.А.

« » 2020 г.

Студент:

Фейзуллин К.М.

«20» октября 2020 г.

Содержание

1. Введение.....	6
2. Техническое задание.....	7
3. Описание системы в нотации UML	8
3.1. Диаграмма прецедентов использования	8
3.2. Диаграмма деятельности	10
3.3. Диаграмма последовательности.....	11
4. Создание базы данных.....	12
4.1. Интеллект-карта базы данных.....	12
4.2. Состав базы данных	12
4.3. Бизнес-правила базы данных Ошибка! Закладка не определена.	
4.4. Схема базы данных.....	12
4.5. Наполнение БД	Ошибка! Закладка не определена.
5. Прототип интерфейса системы.....	13
6. Программная реализация блока взаимодействия с базой данных.....	15
6.1. Структура программы	16
7. Приложения	20

Цель курсовой работы

Целью курсовой работы является освоение основных приемов:

- работы по проектированию информационных систем;
- работы по применению современных методологий структурного анализа и проектирования;
- освоение этапов разработки программного обеспечения;
- реализации программного обеспечения на примере системы,
- п
- р – оформления сопроводительной и пользовательской документации;
- е – разработки пользовательского интерфейса и созданию выходных форм.

Задача курсовой работы

а Задачей курсовой работы является: проектирование и разработка программного продукта, реализующего виртуальную канбан-доску в виде полностью работоспособного приложения.

а

ч

е

н

н

о

й

д

л

я

р

е

ш

е

н

и

Аннотация

В настоящей курсовой работе был спроектирован и разработан программный продукт, реализующий виртуальную канбан-доску, предназначенную для оптимизации процесса разработки программного обеспечения путем распределения и контроля задач в течение проектной деятельности. Исходные коды программы написаны на языке C# в среде разработки Microsoft Visual Studio Community 2019, с установленными компонентами «.NET Framework 4.8».

В ходе работы было разработано техническое задание, создана интеллект-карта по предметной области, выполнено описание системы в нотации UML, создан прототип интерфейса системы и база данных. А также осуществлено тестирование готового продукта и подготовлена необходимая сопроводительная документация.

1. Введение

Канбан - это способ управления проектами, который относится к методологии Agile. Основная его задача - визуализация каждого этапа реализации бизнес-плана.

Канбан помогает сделать рабочий процесс более прозрачным и эффективным, выявить и устранить потенциальные недостатки, а также равномерно распределить роли среди членов команды. Таким образом, шансы на завершение проекта с минимальными затратами и в срок повышаются, что интересовало руководителей во все времена.

Главной же выгодной особенностью программных канбан-продуктов является их независимость от местонахождения сотрудников организации – каждый работник, имея собственный ПК и подключение к Глобальной сети, может пользоваться этим инструментом. Это дает возможность организовывать эффективную удаленную работу, что особенно актуально в настоящее время.

2. Техническое задание

Техническое задание (ТЗ) было разработано в соответствии с ГОСТ 19.201-78. В данном разделе можно ознакомиться с более точной постановкой задачи, основанием и назначением для разработки, различными требованиями к программному продукту, а также составу и содержанию этапов выполнения данной работы. Полная копия ТЗ, включая титульный лист, приведена в Приложении 1.

3. Описание системы в нотации UML

UML-диаграммы были построены с помощью программного продукта Visio 2016. Выбор данного инструмента обусловлен знакомством с данной системой (в ней было удобно строить различные блок-схемы), полным необходимым набором UML компонентов, а также ее интуитивной понятностью.

3.1. Диаграмма прецедентов использования

Описание бизнес-процессов:

Программа предоставляет пользователю удобно организовать контроль рабочего процесса проекта путем карточек с задачами. Пользователи делятся на две категории: мастер и работник. Мастер создает проект, получает статус администратора, и создает учетные записи работников. Мастер создает задачи, имеет возможность менять их статусы и редактировать их содержание. Также мастер добавляет новых сотрудников (выдает им логин и пароль). Работник после авторизации имеет возможность менять статус своей задачи, а неавторизованный работник может только просматривать задачи.

Акторы:

1. Мастер
2. Работник

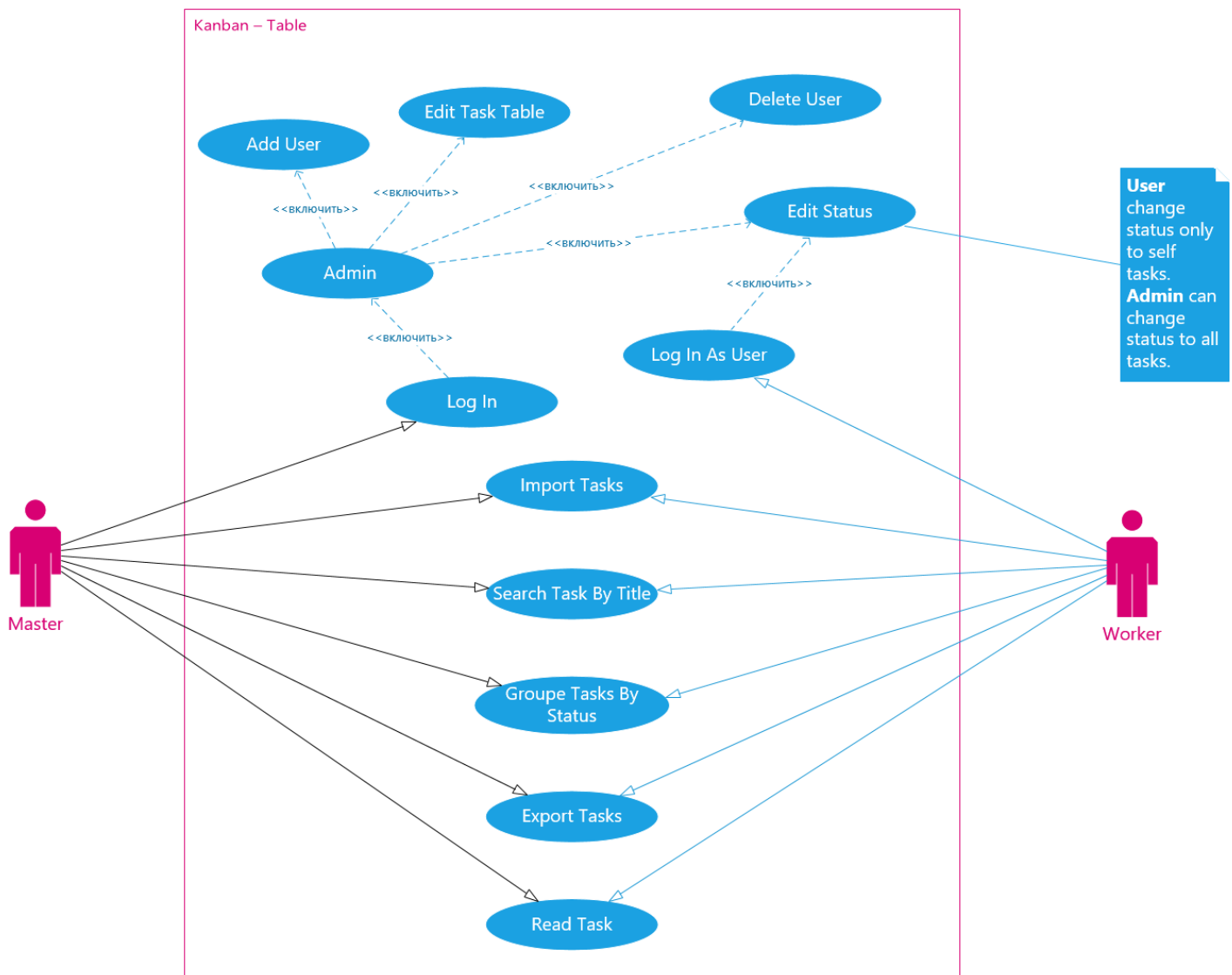


Рисунок 1 - диаграмма прецедентов использования приложения

3.2. Диаграммы деятельности

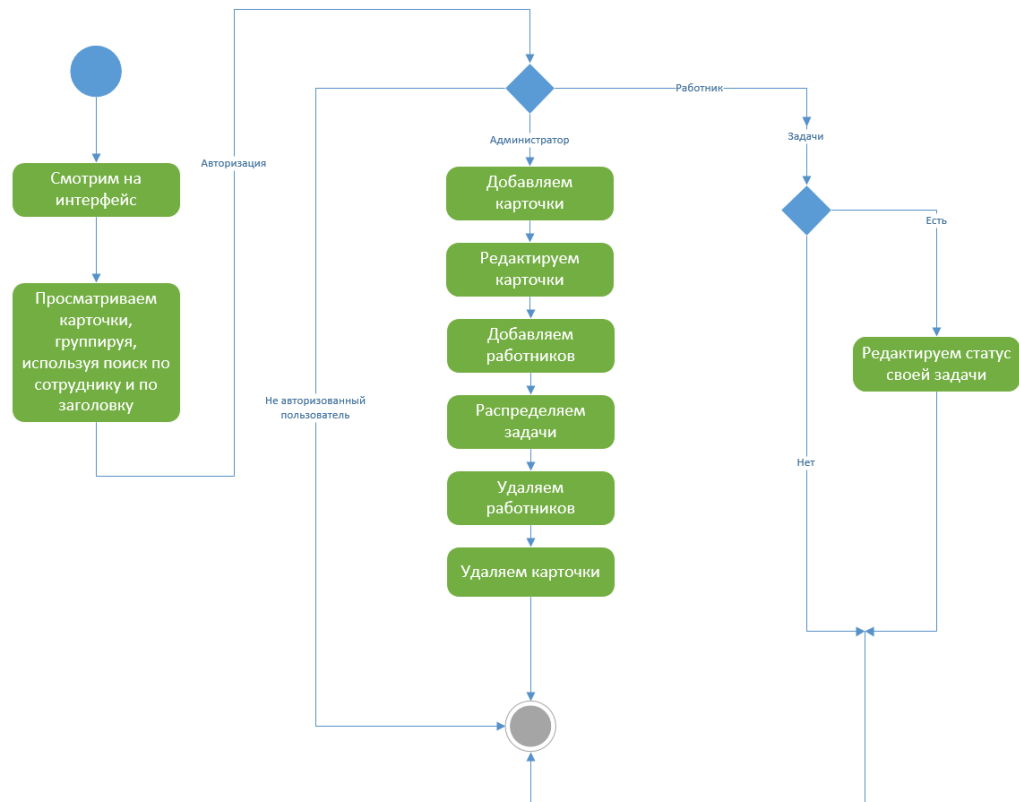


Рисунок 2 - диаграмма деятельности приложения

3.3. Диаграмма последовательности

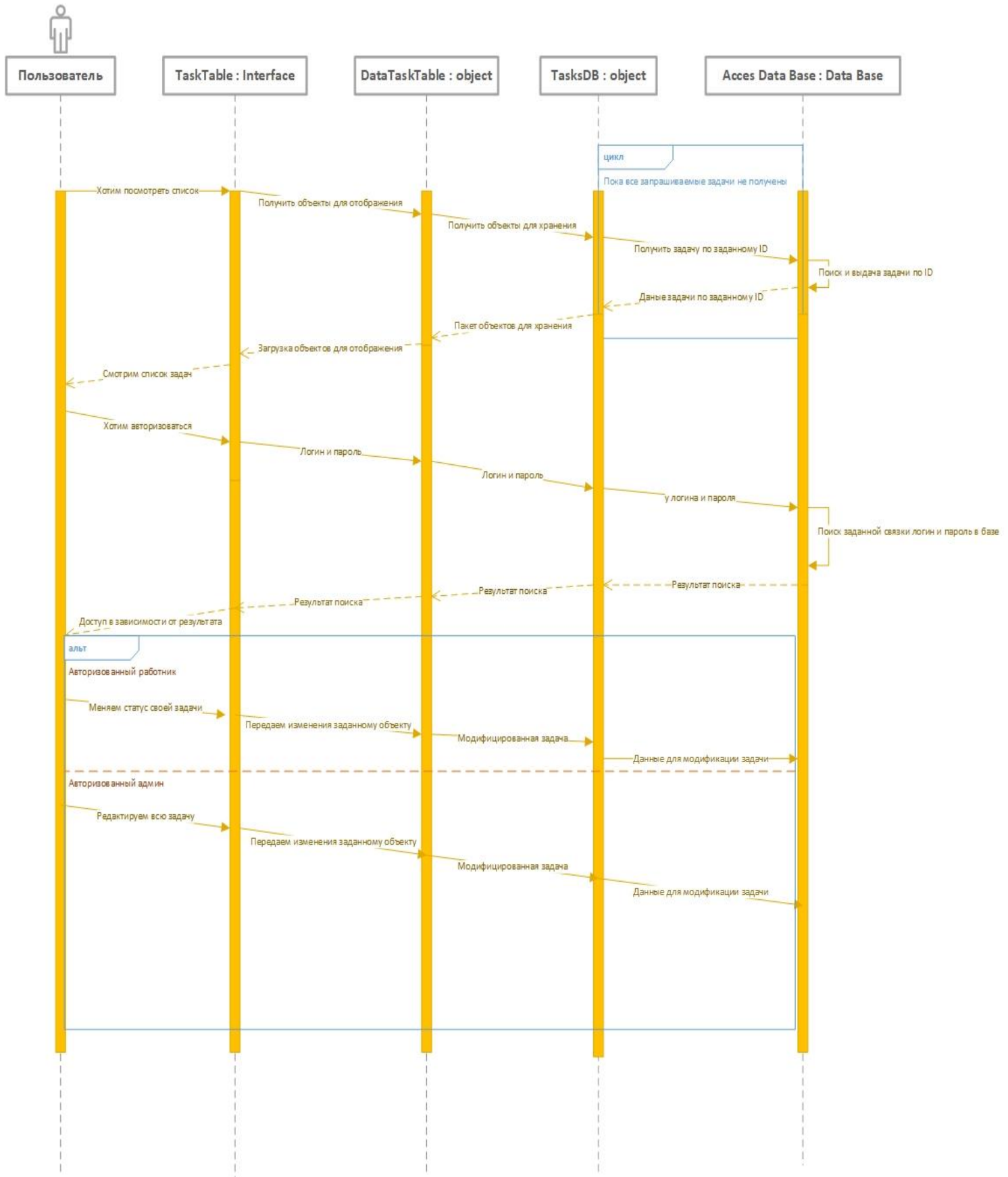


Рисунок 3 - диаграмма последовательности приложения

4. Создание модуля интерфейса

Для реализации интерфейса программы используется язык программирования C# в среде программирования Visual Studio Community 2019, с применением системных модулей Windows Forms.

4.1. Интеллект-карта интерфейса

Для выполнения этого элемента работы был выбран инструмент Coggle. Данный выбор обусловлен простотой, приятным дизайном этого приложения, а также возможностью бесплатного использования.

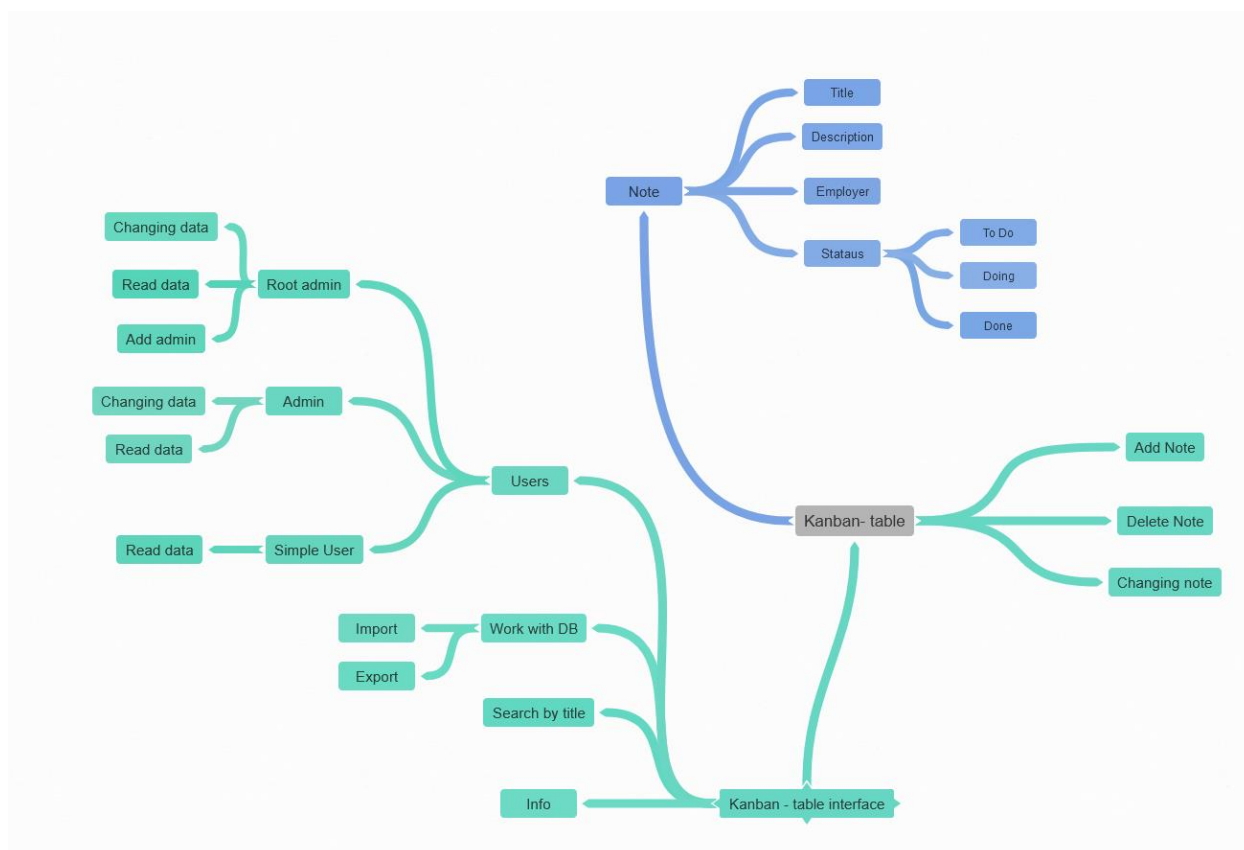


Рисунок 4 - интеллект-карта базы данных

4.2. Состав модуля интерфейса

Так как модуль интерфейса реализует помимо взаимодействия с карточками, распределение привилегий пользователя и фильтрацию по свойствам задач, такими как статут, заголовок и работник, то модуль включает в себя:

- Класс главной формы – реализует взаимодействие с главной формой приложения для загрузки и сохранения базы, вызова окна авторизации,

вызова окна добавления работника, вызова окна удаления работника.

Для редактирования задач на доске, фильтрации их по разным свойствам.

- Класс пользовательского элемента управления задачи (листочка) – так как каждая задача является автономным динамическим элементом, для связи отображаемой задачи с базой данных требуется доработать системный элемент, добавив к нему новый интерфейс.
- Класс формы авторизации (добавления и удаления работника) – реализует интерфейс авторизации работника, добавления работника и его удаления из базы. Так как все три действия завязаны на одних и тех же текстовых полях и в зависимости от сценария они видоизменяются.

5. Прототип интерфейса системы

Для создания интерфейса системы воспользуемся приложением NinjaMock. Это бесплатный онлайн-сервис, что является большим плюсом, так как не вынуждает устанавливать дополнительное ПО. Созданный прототип интерфейса изображен на отдельных рисунках, изображающих различные окна приложения.

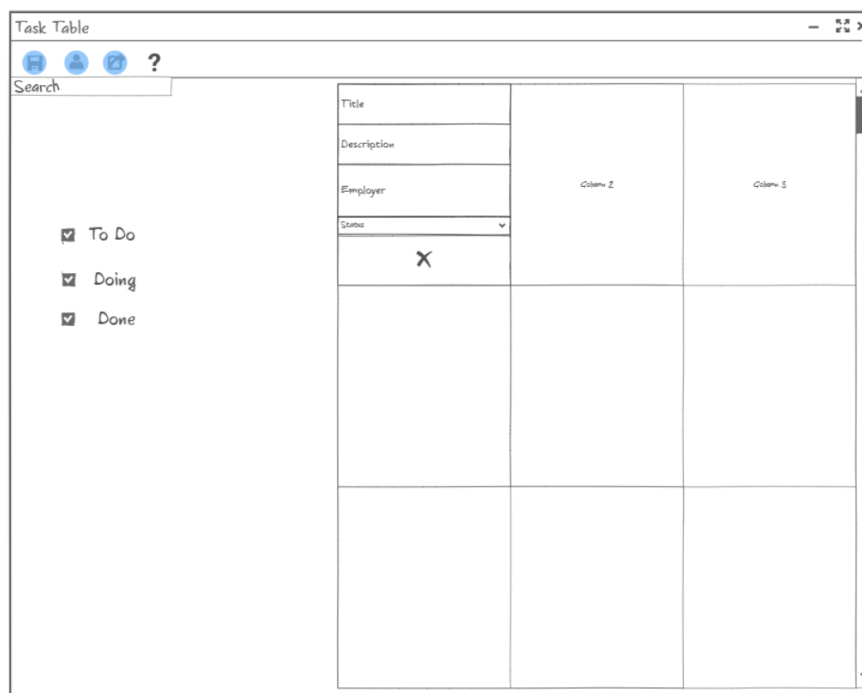


Рисунок 5 - прототип главного экрана



Рисунок 6 - прототип меню импорта/экспорта базы данных

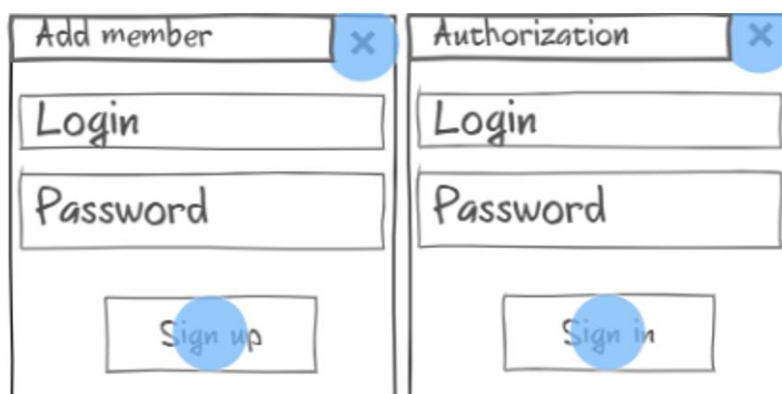


Рисунок 7 - прототип окон авторизации/добавления пользователя

6. Программная реализация блока интерфейса

Написание программного кода было осуществлено на языке C# в среде разработки Microsoft Visual Studio Community 2019, с установленными компонентами «.NET Framework 4.8». Данное решение позволило программировать на похожем по синтаксису, но чуть более высокоуровневом относительно C++ языке и иметь на выходе готовое приложение под операционную систему Windows.

Для реализации блока интерфейса был создан основной класс формы `TaskTableForm` и написаны методы для использования класса, содержащего в себе данные о задачах, которые были использованы в пространстве имен `TaskManager` для реализации логики и интерфейса приложения. С целью упрощения работы использовано пространство имен `System.Collections.Generic` которое содержит классы структур данных таких как список. Также используется пространство имен `System.IO`, для доступа к системным функциям копирования, определения директории. Системное пространство имен `System.Windows.Forms`, используется для реализации работы интерфейса взаимодействия, отображаемый на экране.

Для упрощения понимания написанного кода, была использована XML-документация – специальные теги XML, которые содержатся в комментариях и описывают методы класса `TaskDB`.

6.1. Структура программы

Структуру программы отобразим с помощью схем классов:

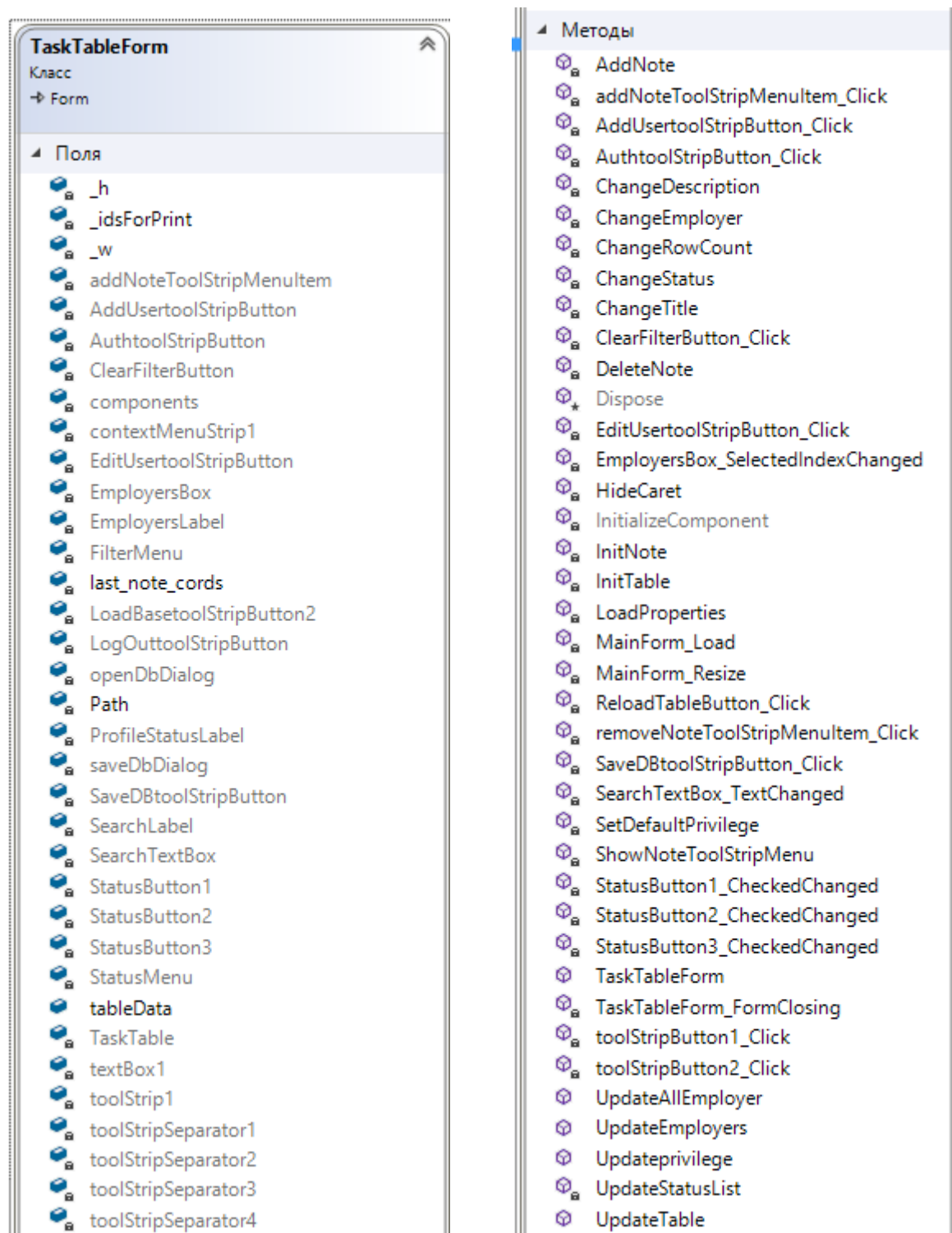


Рисунок 8 - схема класса TaskTableForm



Рисунок 9 - схема класса для авторизации UserForm

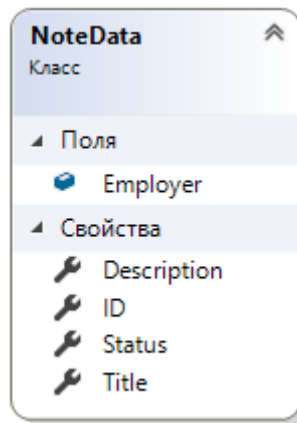


Рисунок 10 - схема класса для пользовательского элемента управления.

Код программы с подробными комментариями размещен в Приложении 2. Скриншоты работы готового приложения находятся в Приложении 3.

7. Руководство по эксплуатации

Руководства по эксплуатации для пользователя и администратора системы оформлены по правилам ГОСТ 19.101-77 и размещены соответственно в Приложении 4 и Приложении 5.

Техническое задание

**Канбан-доска для автоматизации процесса распределения и контроля
задач проекта создания прикладного программного обеспечения**

**Канбан-доска
ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

На 7 листах

СОГЛАСОВАНО

Должность: профессор, д.т.н.

Егорова Алла Альбертовна

« ____ » _____ 2020 г.

Аннотация

Настоящее Техническое задание (ТЗ) определяет назначение, общие и специальные требования к созданию канбан-доски, предназначенной для оптимизации процесса разработки программного обеспечения путем распределения и контроля задач в течение проектной деятельности.

Содержание

Постановка задачи.....	22
Основание для разработки	23
Назначение разработки.....	23
Технические требования к программному изделию	23
Требования к функциональным характеристикам	23
Требование к надежности	24
Условия эксплуатации.....	24
Климатические условия эксплуатации	24
Требования к видам обслуживания	24
Требования к численности и квалификации персонала	24
Требования к составу и параметрам технических средств.....	24
Требования к информационной и программной совместимости.....	24
Требование к исходным кодам и языка программирования.....	24
Требования к программным средствам, используемым программой.....	24
Требования к маркировке и упаковке	25
Требования к транспортировке и хранению	25
Требования к программной документации	25
Состав программной документации.....	25
Специальные требования к программной документации	25
Технико-экономические показатели	25
Состав и содержание этапов работ по созданию канбан-доски.....	26

Постановка задачи

Проектирование и разработка программного продукта, реализующего виртуальную канбан – доску, одного из инструментов управления в ходе разработки программного обеспечения.

Канбан-доску можно рассматривать как вариацию на тему традиционных канбан-карточек (записок с задачами). Вместо сигнальных карточек, которые обычно обозначают потребность или пропускную способность, вместе с доской используются магниты, пластиковые фишки, цветные шайбы или стикеры для представления рабочих элементов и процессов. Каждый из этих объектов представляет собой этап производственного процесса и движется по доске, по мере прогресса. Такое движение соответствует движению процесса производства. Доска, как правило, разделена на три логические секции: «ожидание», «работа в процессе» и «завершенная работа». Сотрудники перемещают заметки в ту секцию доски, которая соответствует статусу задачи.

Канбан-доска:

- Предназначена для:
 - ✓ рядовых сотрудников в качестве визуализации их задач,
 - ✓ лидера команды как инструмент управления ходом разработки программного обеспечения;
- Используется на всех этапах спринта (анализ, проектирование, программирование, тестирование и т.д.)
- Обладает следующими возможностями:
 - ✓ Добавление и удаление карточек;
 - ✓ Редактирование карточек на доске;
 - ✓ Поиск карточек по заголовку задачи;
 - ✓ Импорт / экспорт карточек;

✓ Разделение карточек по статусу;

- Программа должна визуально отображать набор карточек в доступном и удобном для восприятия и взаимодействия в приложении.

Основание для разработки

Основанием для разработки данного программного продукта является выполнение курсовой работы по дисциплине «Проектирование программного обеспечения (ППО)».

Назначение разработки

Произведенное ПО будет использоваться для визуализации этапов разработки проектов у команд, отделов в целом, что позволит самостоятельно распределять задачи в команде разработки и четко отслеживать ход выполнения рабочего процесса.

Технические требования к программному изделию

Требования к функциональным характеристикам

Программный продукт должен выполнять базовый функционал:

1. добавление записей на доску;
2. удаление записей с доски;
3. редактирование содержания записей;
4. упорядочивание записей по статусу задачи и по работнику, за которым задачи закреплены;
5. Поиск по заголовку записи;
6. Импорт и экспорт данных в формате mdb-файла базы данных.
7. Разделение в возможностях пользователей. Пользователи, обладающие статусом администратора, будут обладать всем возможным функционалом. Обычным авторизованным пользователям будет открыт доступ к редактированию статуса задачи, за которой он закреплен. Неавторизованные пользователи смогут только просматривать задачи.

Исходными данными выступает mdb-файл, содержащий данные по каждой записи и сотруднику.

Требование к надежности

Пользователю, работающему с программой, должен быть предоставлен непрерывный доступ к записям. Программа не должна непредвиденно прерывать свою работу.

Условия эксплуатации

Климатические условия эксплуатации

Требования к климатическим условиям эксплуатации не предъявляются.

Требования к видам обслуживания

Требования к обслуживанию отсутствуют.

Требования к численности и квалификации персонала

Для управления системой достаточно одного человека, способного запустить программу. Требуемая квалификация пользователя – уверенный пользователь персонального компьютера.

Требования к составу и параметрам технических средств

Для корректной работы приложения требуется операционная система «Windows 7» и выше.

Требования к информационной и программной совместимости

Требование к исходным кодам и языка программирования

Исходные коды программы должны быть написаны на языке C# в среде разработки Microsoft Visual Studio Community 2019 и выше, с установленными компонентами «.NET Framework 4.8» или «.NET Core 3.1».

Требования к программным средствам, используемым программой

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы не ниже Windows 7.

Требования к маркировке и упаковке

Программа поставляется в виде EXE-файла и не требует установки.

Требования к маркировке и упаковке не предоставляется.

Требования к транспортировке и хранению

Приложение хранится на Интернет-ресурсе Github: ссылка.

Специальные требования к транспортировке и хранению не предъявляется.

Требования к программной документации

Состав программной документации

- «Канбан-доска». Техническое задание (ГОСТ 19.201-78 и ГОСТ 34.602-89);
- «Канбан-доска». Пояснительная записка (ГОСТ 19.404-79);
- «Канбан-доска». Текст программы (ГОСТ 19.401-78);

Специальные требования к программной документации

Вся документация должна быть оформлена в соответствии с вышеуказанными ГОСТами, к каждому виду документа (смотреть предыдущий пункт).

Технико-экономические показатели

С учетом того, что будет необходимо осваивать новый язык программирования (C#) и особенности взаимодействия с базами данных, исполнение проекта займет 101 человеко-час, из которых на каждую задачу отведено по 4 часа и взят некоторый запас с учетом возможных отклонений от запланированного темпа работы.

Уровень экономической эффективности от внедрения автоматизированной системы зависит от умения работников проводить организованную работу, так как инструмент является лишь вспомогательным средством для уменьшения временных затрат на распределение и проверку задач в течение проекта.

Состав и содержание этапов работ по созданию канбан-доски

Таблица 1 - Перечень стадий и этапов разработки

№	Этап работ	Документы, предъявляемые к результатам
1	Ознакомление с идеологией и принципом работы канбан-доски	
2	Написание ТЗ	Техническое задание по ГОСТ 19.201-78
3	Описание системы электронной канбан-доски в UML*	
4	Реализация базового интерфейса канбан – доски**	Текст программы по ГОСТ 19.401-78
5	Добавление записей на доску	Текст программы по ГОСТ 19.401-78
6	Удаление записи с доски	Текст программы по ГОСТ 19.401-78
7	Добавление пустой записи на доску	Текст программы по ГОСТ 19.401-78
8	Удаление всех записей с доски	Текст программы по ГОСТ 19.401-78
9	Редактирование записи на доске	Текст программы по ГОСТ 19.401-78
10	Разделение функциональных возможностей для пользователя и админа	Текст программы по ГОСТ 19.401-78
12	Тестирование системы	
13	Написание документации***	ГОСТ 19.101-77

* - Создание UML-диаграмм: прецедентов использования, следования и активности.

** - Реализация добавления данных по записям и работникам, редактирования этих данных, а также удаления этих данных с доски.

*** - Написание руководства пользователя системы и руководства администратора системы.

Листинг класса TaskTableForm, реализующий интерфейс главной формы

```
public partial class TaskTableForm : Form
{
    private int _w;
    private int _h;
    [DllImport("user32.dll")]
    static extern bool HideCaret(IntPtr hWnd);
    private int[] last_note_cords = { 0, 0 };
    private string Path = Properties.Settings.Default.PathToDB;
    public TaskTableData tableData = null;
    private List<int> _idsForPrint = new List<int>();

    /// <summary>
    /// Конструктор главной формы
    /// </summary>
    public TaskTableForm()
    {
        InitializeComponent(); //Инициализируем меню и доску для записей, но без
самых записей.
        //Запоминаем размеры главно для дальнейшего использования
        _w = this.Width;
        _h = this.Height;
        TaskTable.ContextMenuStrip = contextMenuStrip1; // Добавляем для доски с
записями контекстное меню на ПКМ.
        TaskTable.ControlRemoved += new ControlEventHandler(ChangeRowCount);
    }

    /// <summary>
    /// Метод, вызываемый системой при показе главной формы, при вызове метода Show()
    /// Вызываем загрузку свойств системы, свойств привелегий пользователя и
загружаем данник из базы,
    /// если она указана верно
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void MainForm_Load(object sender, EventArgs e)
    {
        LoadProperties();
        Updateprivilege();
        if (!File.Exists(Properties.Settings.Default.PathToDB))
        {
            MessageBox.Show("Заданной базы данных не существует");
        }
        else
        {
            InitTable();
        }
    }

    /// <summary>
    /// Загружаем и устанавливаем настройки приложения
    /// </summary>
    private void LoadProperties()
    {
        Properties.Settings.Default.Reload(); //Загружаем системные настройки из
системного файла
        Properties.Settings.Default.Admin = false;
        Properties.Settings.Default.User = false;
    }
}
```

```

        Properties.Settings.Default.UserID = -1;
    }

    /// <summary>
    /// Загрузка задач на доску и вызов обслуживающих методов
    /// </summary>
    private void InitTable()
    {
        tableData = new TaskTableData(Properties.Settings.Default.PathToDB); //
        Создаем экземпляр класса для работы с данными из базы по заданному пути
        _idsForPrint = tableData.forPrint; //Список id задач для отображения на доске
        UpdateEmployers();
        UpdateAllEmployer();
        UpdateTable();
    }

    //Создаем саму запись как объект, добавляем текстовые поля и события для
    взаимодействия
    /// <summary>
    /// Метод для создания листочка задачи, возвращает созданный листочек
    /// </summary>
    /// <param name="note"></param>
    /// <returns></returns>
    private System.Windows.Forms.TableLayoutPanel InitNote(NoteData note)
    {
        var NewNote = new Note() { Margin = new Padding(10), ID = note.ID };
        NewNote.BackColor = System.Drawing.SystemColors.Window;
        NewNote.CellBorderStyle =
        System.Windows.Forms.TableLayoutPanelCellBorderStyle.Single;
        NewNote.ColumnCount = 1;
        NewNote.ColumnStyles.Add(new
        System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 50F));
        NewNote.Location = new System.Drawing.Point(3, 3);
        NewNote.RowCount = 5;
        NewNote.RowStyles.Add(new
        System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 18.18182F));
        NewNote.RowStyles.Add(new
        System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 81.81818F));
        NewNote.RowStyles.Add(new
        System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 28F));
        NewNote.RowStyles.Add(new
        System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 33F));
        NewNote.RowStyles.Add(new
        System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 35F));
        NewNote.RowStyles.Add(new
        System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 20F));
        NewNote.Size = new System.Drawing.Size(330, 320);
        NewNote.TabIndex = 0;
        //6 Здесь заполняются текстовые поля предоставленными данными и добавляются
        на Запись
        var textbox = new System.Windows.Forms.TextBox()
        {
            BackColor = Color.White,
            Multiline = true,
            Text = note.Title,
            Width = 260,
            Size = new System.Drawing.Size(330, 30),
            ReadOnly = !Properties.Settings.Default.Admin
        };
        textbox.TextChanged += (sender, args) => ChangeTitle(note.ID, (sender as
        TextBox).Text);
        HideCaret(textbox.Handle);
        NewNote.Controls.Add(textbox);
    }

```

```

textbox = new System.Windows.Forms.TextBox()
{
    BackColor = Color.White,
    Multiline = true,
    Text = note.Description,
    WordWrap = true,
    ScrollBars = ScrollBars.Vertical,
    Size = new System.Drawing.Size(330, 250),
    ReadOnly = !Properties.Settings.Default.Admin
};
textbox.TextChanged += (sender, args) => ChangeDescription(note.ID, (sender
as TextBox).Text);
HideCaret(textbox.Handle);
NewNote.Controls.Add(textbox);

var box = new System.Windows.Forms.ComboBox()
{
    BackColor = Color.White,
    Text = note.Employer.Name,
    Width = 260,
    Size = new System.Drawing.Size(330, 20),
    Enabled = Properties.Settings.Default.Admin
};
foreach (var employer in tableData.allEmployers)
{
    box.Items.Add(employer);
}
box.TextChanged += (sender, args) => ChangeEmployer(note.ID, (sender as
ComboBox).SelectedItem);
NewNote.Controls.Add(box);

box = new System.Windows.Forms.ComboBox()
{
    BackColor = Color.White,
    Text = note.Status,
    Width = 260,
    Size = new System.Drawing.Size(330, 20),
    Items = { "To Do", "Doing", "Done" },
    Enabled = (note.Employer.ID == Properties.Settings.Default.UserID) ||
Properties.Settings.Default.Admin
};
box.TextChanged += (sender, args) => ChangeStatus(note.ID, (sender as
ComboBox).Text);
NewNote.Controls.Add(box);

System.Drawing.Color color;
switch (note.Status)
{
    case "Doing":
        color = System.Drawing.Color.Orange;
        break;
    case "To Do":
        color = System.Drawing.Color.Silver;
        break;
    case "Done":
        color = System.Drawing.Color.ForestGreen;
        break;
    default:
        color = System.Drawing.Color.Red;
        break;
}

NewNote.Controls.Add(new System.Windows.Forms.TextBox()

```



```

        {
            BackColor = color,
            Dock = System.Windows.Forms.DockStyle.Fill,
            Location = new System.Drawing.Point(4, 286),
            Multiline = true,
            ReadOnly = true,
            Size = new System.Drawing.Size(330, 30),
            Text = "* * *",
            TextAlign = HorizontalAlignment.Center,
            TabIndex = 5,
            Cursor = System.Windows.Forms.Cursors.Hand,
            Enabled = Properties.Settings.Default.Admin
        });
        // #7 Здесь добавляем функциональное меню для элемента записи
        NewNote.Controls[4].ContextMenuStrip = new NoteContextMenu(NewNote,
tableData, this);
        NewNote.Controls[4].Click += new System.EventHandler(ShowNoteToolStripMenu);
        return NewNote;
    }

    /// <summary>
    /// Метод для добавления листочка на доску
    /// </summary>
    /// <param name="note"></param>
    private void AddNote(NoteData note)
    {
        TaskTable.Controls.Add(InitNote(note));
    }

    /// <summary>
    /// Метод для контроля размером доски с листочками
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private static void ChangeRowCount(object sender, EventArgs e)
    {
        System.Windows.Forms.TableLayoutPanelPanel table =
(System.Windows.Forms.TableLayoutPanelPanel)sender;
        table.RowCount = (int)table.Controls.Count / 3;
    }

    /// <summary>
    /// Метод для вызова во время события удаления листочка с доски
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="args"></param>
    /// <param name="form"></param>
    private static void removeNoteToolStripMenuItem_Click(object sender, EventArgs
args, TaskTableForm form)
    {
        var menu = (MenuItemForDeleteNote)sender;
        form.DeleteNote(menu.Note.ID);
    }

    /// <summary>
    /// Вызов всплывающего меню на доске
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private static void ShowNoteToolStripMenu(object sender, EventArgs e)
    {
        System.Windows.Forms.TextBox box = (System.Windows.Forms.TextBox)sender;
        box.ContextMenuStrip.Show(Control.MousePosition);
    }

```

```

    }

    /// <summary>
    /// Добавление листочка на доску и в базу
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void addNoteToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //Добавление записки в базу
        tableData.Add(); //Добавляем пустой листочек в базу
        _idsForPrint = tableData.forPrint; //Обновляем список id для отображения
        UpdateTable();
        UpdateStatusList();
        UpdateEmployers();

    }

    /* ===== CLASSES
    =====*/
    /// <summary>
    /// Меню для взаимодействия с листочком на доске
    /// </summary>
    private class NoteContextMenu : System.Windows.Forms.ContextMenuStrip
    {
        //Инициализируем унаследованный системный компонент с добавлением в него
        наших элементов меню
        public NoteContextMenu(Note note, TaskTableData tableData, TaskTableForm
        form)
        {
            this.Items.Add(new MenuItemForDeleteNote(note, tableData, form));
            this.Items[0].Text = "Remove Note";
            this.Size = new System.Drawing.Size(180, 22);
        }
    }

    /// <summary>
    /// Элемент меню для удаления листочка с доски
    /// </summary>
    private class MenuItemForDeleteNote : ToolStripMenuItem
    {
        //Инициализируем наш элемент меню, унаследованный системный компонент
        public MenuItemForDeleteNote(Note note, TaskTableData tableData,
        TaskTableForm form)
        {
            //Добавляем в него событие удаления записи
            this.Click += (sender, args) =>
            {
                removeNoteToolStripMenuItem_Click(sender, args, form);
                form.UpdateTable();
                tableData.UpdateEmployers();
                tableData.UpdateStatusList();
            };
            //Сохраняем саму запись для удаления
            Note = note;
        }
        public Note Note;
    }

    /// <summary>
    /// Модификация стандартного элемента для хранения ID записи
    /// </summary>

```

```

private class Note : TableLayoutPanel
{
    public int ID = 0;
}
/* ===== CLASSES
===== */

/// <summary>
/// Метод для контроля расположения компонентов на форме при изменении ее
размеров
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void MainForm_Resize(object sender, EventArgs e)
{
    int d_w = this.Width - _w;
    int d_h = this.Height - _h;
    _w = this.Width;
    _h = this.Height;
    TaskTable.Width += d_w;
    TaskTable.ColumnCount = (int)(TaskTable.Width / 350);
    TaskTable.Height += d_h;
    TaskTable.RowCount = (int)(TaskTable.Height / 330);
    TaskTable.AutoScrollMargin = new Size(10, TaskTable.Height);
    // TaskTable.AutoScroll = true;
}

/// <summary>
/// Метод события при нажатии на вывод задач со статусом "Нужно сделать"
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void StatusButton1_CheckedChanged(object sender, EventArgs e)
{
    //Переводим другие элементы фильтра в состояние по умолчанию
    EmployersBox.Text = "";
    SearchTextBox.Text = "";
    StatusButton2.Checked = false;
    StatusButton3.Checked = false;
    _idsForPrint = tableData.TasksByStatus("To Do");
    UpdateTable();
}

/// <summary>
/// Метод события при нажатии на вывод задач со статусом "В работе"
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void StatusButton2_CheckedChanged(object sender, EventArgs e)
{
    //Переводим другие элементы фильтра в состояние по умолчанию
    EmployersBox.Text = "";
    SearchTextBox.Text = "";
    StatusButton1.Checked = false;
    StatusButton3.Checked = false;
    _idsForPrint = tableData.TasksByStatus("Doing");
    UpdateTable();
}

/// <summary>
/// Метод события при нажатии на вывод задач со статусом "Сделано"
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

```

```

private void StatusButton3_CheckedChanged(object sender, EventArgs e)
{
    //Переводим другие элементы фильтра в состояние по умолчанию
    EmployersBox.Text = "";
    SearchTextBox.Text = "";
    StatusButton1.Checked = false;
    StatusButton2.Checked = false;
    _idsForPrint = tableData.TasksByStatus("Done");
    UpdateTable();
}

/// <summary>
/// Обновление отображаемых задач на форме
/// </summary>
public void UpdateTable()
{
    TaskTable.Controls.Clear(); //Очищаем наполнение доски
    //Добавляем объекты для отображения на доску
    foreach (var id in _idsForPrint)
    {
        AddNote(tableData[id]);
    }
}

/// <summary>
/// Метод, вызываемый при выборе работника, для которого нужно отобразить задачи
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void EmployersBox_SelectedIndexChanged(object sender, EventArgs e)
{
    //Переводим другие элементы фильтра в состояние по умолчанию
    StatusButton1.Checked = false;
    StatusButton2.Checked = false;
    StatusButton3.Checked = false;
    SearchTextBox.Text = "";
    _idsForPrint = tableData.GetTasksIdByEmployerId((EmployersBox.SelectedItem as
Employer).ID, 1);
    UpdateTable();
}

/// <summary>
/// Метод для очистки установок фильтра
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void ClearFilterButton_Click(object sender, EventArgs e)
{
    //Переводим все элементы фильтра в состояние по умолчанию
    StatusButton1.Checked = false;
    StatusButton2.Checked = false;
    StatusButton3.Checked = false;
    EmployersBox.Text = "";
    SearchTextBox.Text = "";
    _idsForPrint = tableData.GetAllIds();
    UpdateTable();
}

/// <summary>
/// Метод вызываемый при использовании окна поиска
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void SearchTextBox_TextChanged(object sender, EventArgs e)

```

```

{
    //Проверка на пустой поиск
    if((SearchTextBox.Text.Length < 2) && (!SearchTextBox.Text.StartsWith(" ")))
    {
        //Переводим другие элементы фильтра в состояние по умолчанию
        StatusButton1.Checked = false;
        StatusButton2.Checked = false;
        StatusButton3.Checked = false;
        EmployersBox.Text = "";
        _idsForPrint = tableData.GetTasksIdByTitle(SearchTextBox.Text, 1);
        UpdateTable();
    }
}

/// <summary>
/// Метод, вызываемый при изменении заголовка задачи
/// </summary>
/// <param name="id"></param>
/// <param name="newStr"></param>
private void ChangeTitle(int id, string newStr)
{
    tableData[id].Title = newStr;
    tableData.UpdateNote(id); //Обновляем данные о задаче в объекте
}

/// <summary>
/// Метод, вызываемый при изменении описания задачи
/// </summary>
/// <param name="id"></param>
/// <param name="newStr"></param>
private void ChangeDescription(int id, string newStr)
{
    tableData[id].Description = newStr;
    tableData.UpdateNote(id); //Обновляем данные о задаче в объекте
}

/// <summary>
/// Метод, вызываемый при изменении работника, за которым закреплена задача
/// </summary>
/// <param name="id"></param>
/// <param name="new_employer"></param>
private void ChangeEmployer(int id, object new_employer)
{
    tableData[id].Employer = new_employer as Employer;
    tableData.UpdateNote(id); //Обновляем данные о задаче в объекте
    UpdateEmployers();
    UpdateTable();
}

/// <summary>
/// Метод, вызываемый при изменении статуса задачи
/// </summary>
/// <param name="id"></param>
/// <param name="newStr"></param>
private void ChangeStatus(int id, string newStr)
{
    tableData[id].Status = newStr;
    tableData.UpdateNote(id); //Обновляем данные о задаче в объекте
    UpdateStatusList();
    UpdateTable();
}

/// <summary>
/// Метод для удаления листочка с доски по ID

```

```

    /// </summary>
    /// <param name="id">id листочка</param>
    private void DeleteNote(int id)
    {
        tableData.DeleteNote(id); //Удаляем листочек из объекта по заданному ID
        _idsForPrint = tableData.forPrint; //Обновляем список id для отображения
    }

    /// <summary>
    /// Метод для обновления списка работников для фильтра
    /// </summary>
    public void UpdateEmployers()
    {
        tableData.UpdateEmployers(); //Обновляем список работников, за которыми
закреплены задачи в объекте
        EmployersBox.Items.Clear(); //Очищаем прошлый список работников, за которыми
закреплены задачи в объекте
        foreach (var employer in tableData.employers)
        {
            EmployersBox.Items.Add(employer); //Добавляем работников в список
        }
    }

    /// <summary>
    /// Обновление списка всех работников
    /// </summary>
    public void UpdateAllEmployer()
    {
        tableData.UpdateAllEmployers(); //Обновление списка всех работников в объекте
    }

    /// <summary>
    /// Обновление списков задач по статусу
    /// </summary>
    private void UpdateStatusList()
    {
        tableData.UpdateStatusList(); // Обновление списков задач по статусу в объекте
    }

    /// <summary>
    /// Обновление отображаемых задач на доске
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void ReloadTableButton_Click(object sender, EventArgs e)
    {
        UpdateTable();
    }

    /// <summary>
    /// Смена действующей базы данных и ее загрузка, если та существует
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void toolStripButton2_Click(object sender, EventArgs e)
    {
        openFileDialog.InitialDirectory = "c:\\\\";
        openFileDialog.Filter = "access mdb files (*.mdb) | *.mdb";
        openFileDialog.RestoreDirectory = true;

        if (openDbDialog.ShowDialog() == DialogResult.OK) //Если диалоговое окно
закрыто успешно
        {

```

```

        //Получаем путь к файлу, который указали в диалоговом окне
        Properties.Settings.Default.PathToDB = openDbDialog.FileName;
        if (!File.Exists(Properties.Settings.Default.PathToDB)) //Проверка на
существование указанной базы
        {
            MessageBox.Show("Заданной базы данных не существует");
            return;
        }
        if(tableData != null)
        {
            tableData.Dispose();
            tableData = null;
        }
        tableData = new TaskTableData(Properties.Settings.Default.PathToDB);
        SetDefaultPrivilege();
        Updateprivilege();
        InitTable();
    }
}

/// <summary>
/// Происходит при закрытии формы. Сохраняет настройки приложения и очищает
память
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void TaskTableForm_FormClosing(object sender, FormClosingEventArgs e)
{
    if(tableData != null)
    {
        tableData.Dispose();
        tableData = null;
    }
    SetDefaultPrivilege();
    Updateprivilege();
    Properties.Settings.Default.Save(); //Сохраняем системные настройки перед
закрытием
}

/// <summary>
/// Вызов окна для авторизации
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void AuthtoolStripButton_Click(object sender, EventArgs e)
{
    var authform = new UserForm(this);
    authform.Show();
}

/// <summary>
/// Вызов окна добавления пользователя
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void AddUsertoolStripButton_Click(object sender, EventArgs e)
{
    var authform = new UserForm(this, true);
    authform.Show();
}

/// <summary>
/// Вызов окна для редактирования профиля пользователя
/// </summary>

```



```

    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void EditUserToolStripButton_Click(object sender, EventArgs e)
    {
        var authform = new UserForm(this, false, true);
        authform.Show();
    }

    /// <summary>
    /// Обновление действующих привелегий пользователя
    /// </summary>
    public void Updateprivilege()
    {
        AddUserToolStripButton.Enabled = Properties.Settings.Default.Admin;
        EditUserToolStripButton.Enabled = Properties.Settings.Default.Admin;
        LogOutToolStripButton.Enabled = Properties.Settings.Default.Admin ||
Properties.Settings.Default.User;
        addNoteToolStripMenuItem.Enabled = Properties.Settings.Default.Admin;
        ProfileStatusLabel.Text = "Авторизованный пользователь:\n" +
Properties.Settings.Default.UserLogin;
    }

    /// <summary>
    /// Выход из авторизованного профиля
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void toolStripButton1_Click(object sender, EventArgs e)
    {
        SetDefaultPrivilege();
        Updateprivilege();
        UpdateTable();
    }

    /// <summary>
    /// Установление возможностей пользователя по умолчанию
    /// </summary>
    private static void SetDefaultPrivilege()
    {
        Properties.Settings.Default.Admin = false;
        Properties.Settings.Default.User = false;
        Properties.Settings.Default.UserID = -1;
        Properties.Settings.Default.UserLogin = "Гость";
    }

    /// <summary>
    /// Вызов диалогового окна для сохранения / копирования действующей базы данных
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void SaveDBtoolStripButton_Click(object sender, EventArgs e)
    {
        saveDbDialog.InitialDirectory = "c:\\";
        saveDbDialog.Filter = " access mdb files (*.mdb) | *.mdb";
        saveDbDialog.RestoreDirectory = true;

        if (saveDbDialog.ShowDialog() == DialogResult.OK)
        {
            File.Copy(Properties.Settings.Default.PathToDB, saveDbDialog.FileName,
true);
        }
    }
}

```

Листинг класса UserForm, реализующий интерфейс формы авторизации

```
public partial class UserForm : Form
{
    public string Login { get; set; }
    public string Password { get; set; }
    public bool IsClosed { get; set; } = true;

    private bool _newMember;
    private bool _editMember;

    private TaskTableForm parent;

    private int _w;
    private int _h;

    /// <summary>
    /// Задаем начальные состояния формы авторизации
    /// </summary>
    /// <param name="parent"></param>
    /// <param name="_new"></param>
    /// <param name="edit"></param>
    public UserForm(TaskTableForm parent, bool _new = false, bool edit = false)
    {
        InitializeComponent();
        _newMember = _new;
        this.parent = parent;
        _editMember = edit;
    }

    /// <summary>
    /// Изменения интерфейса в случае сценария вызова окна для добавления работника
    /// </summary>
    private void NewMember()
    {
        GoButton.Text = "Добавить работника";
        this.Text = "Добавление работника";
    }

    /// <summary>
    /// Событие нажатия на кнопку формы авторизации
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void GoButton_Click(object sender, EventArgs e)
    {
        if((LoginBox.Text == string.Empty) || (PasswordBox.Text == string.Empty))
        //Если какие-то поля пропущены, выходим
        {
            MessageBox.Show("Введены не все данные, повторите ввод");
            return;
        }
        if (_newMember) //Если сценарий вызова окна для добавления нового работника
        {
            if(NameBox.Text == string.Empty) //Если поле пропущено, выходим
            {
                MessageBox.Show("Введены не все данные, повторите ввод");
                return;
            }
        }
    }
}
```

```

        var db = new TaskDB(Properties.Settings.Default.PathToDB);
        db.Login(NameBox.Text, LoginBox.Text, PasswordBox.Text); //Добавляем
нового работника в базу
        parent.UpdateAllEmployer();
        parent.UpdateTable();
        parent.UpdateEmployers();
    }

    if(_editMember) //Если сценарий вызова окна для редактирования пользователя
    {
        var db = new TaskDB(Properties.Settings.Default.PathToDB);
        //Редактирование пользователя, выбранного в UserBox
    }

    //Обычная авторизация
    if(!_editMember && !_newMember)
    {
        var db = new TaskDB(Properties.Settings.Default.PathToDB);
        bool? res = db.Verification(LoginBox.Text, PasswordBox.Text);
        if(res != null) //Если работник не найден в базе, то res = null
        {
            Properties.Settings.Default.Admin = (bool)res; //Если авторизованный
пользователь - Администратор, то res = true, если обычный работник, то false
            Properties.Settings.Default.User = true; //Говорим, что пользователь
авторизован
            Properties.Settings.Default.UserID =
(int)db.GetIdByLogin(LoginBox.Text); //Запоминаем id пользователя
            Properties.Settings.Default.UserLogin = LoginBox.Text + ((bool)res ?
"(Админ)" : "(Работник)"); //Строка для статуса пользователя
            parent.Updateprivilege();
        }
        parent.UpdateTable();
    }
    IsClosed = false;
    this.Close();
}

/// <summary>
/// Загрузка формы авторизации в зависимости от сценария использования
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void UserForm_Load(object sender, EventArgs e)
{
    NameBox.Enabled = _newMember;
    NameLabel.Enabled = _newMember;
    NameLabel.Visible = _newMember;
    NameBox.Visible = _newMember;
    UsersBox.Enabled = _editMember;
    UsersBox.Visible = _editMember;
    GoButton.Enabled = !_editMember;
    GoButton.Visible = !_editMember;
    _w = this.Width;
    _h = this.Height;
    if(_editMember) //Если сценарий вызова окна для редактирования пользователя
    {
        UpdateEmployers();
        this.Text = "Редактирование профиля";
        //Заменяем кнопку на форме на новый элемент с процедурой для удаления
пользователя
        var btn = new Button { Location = GoButton.Location, Text = "Удалить",
Size = GoButton.Size };
        btn.Click += (_sender, args) =>
        {

```

```

        var _db = new TaskDB(Properties.Settings.Default.PathToDB);
        _db.DeleteEmployer((UsersBox.SelectedItem as Employer).ID);
        parent.UpdateAllEmployer();
        parent.UpdateEmployers();
        parent.UpdateTable();
        UpdateEmployers();

    };
    this.Controls.Add(btn);
    /*
     * Добавить редактирование пользователя
     */
}
if (_newMember) // Изменения интерфейса в случае сценария вызова окна для
добавления работника
{
    NewMember();
}
}

/// <summary>
/// Обновление списка всех пользователей для редактирования
/// </summary>
private void UpdateEmployers()
{
    UsersBox.Items.Clear();
    UsersBox.Text = "";
    var db = new TaskDB(Properties.Settings.Default.PathToDB);
    var employers = db.GetEmployers();
    foreach (var employer in employers)
    {
        UsersBox.Items.Add(employer);
    }
}

/// <summary>
/// Не даем форме изменять свои размеры
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void UserForm_Resize(object sender, EventArgs e)
{
    this.Width = _w;
    this.Height = _h;
}

/// <summary>
/// Не даем форме изменять свои размеры
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void UserForm_ResizeBegin(object sender, EventArgs e)
{
    this.Width = _w;
    this.Height = _h;
}

/// <summary>
/// Не даем форме изменять свои размеры
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void UserForm_ResizeEnd(object sender, EventArgs e)

```

```
{  
    this.Width = _w;  
    this.Height = _h;  
}
```

ПРИЛОЖЕНИЕ 3

Скриншоты работы приложения

Task Table

Фильтры

Поиск по заголовку:

Сотрудник:

Статус задачи

☐ Нужно сделать

☐ В работе

☐ Выполнено

Авторизованный пользователь:

Гость

Очистить фильтр

Заголовок после update	Логика приложения	Интерфейс приложения
Описание после update	Продумать логику приложения	Продумать интерфейс приложения
Баранов	Фейзуллин	Андреев
Doing	Done	Doing

Рисунок П.3.1 - главная форма

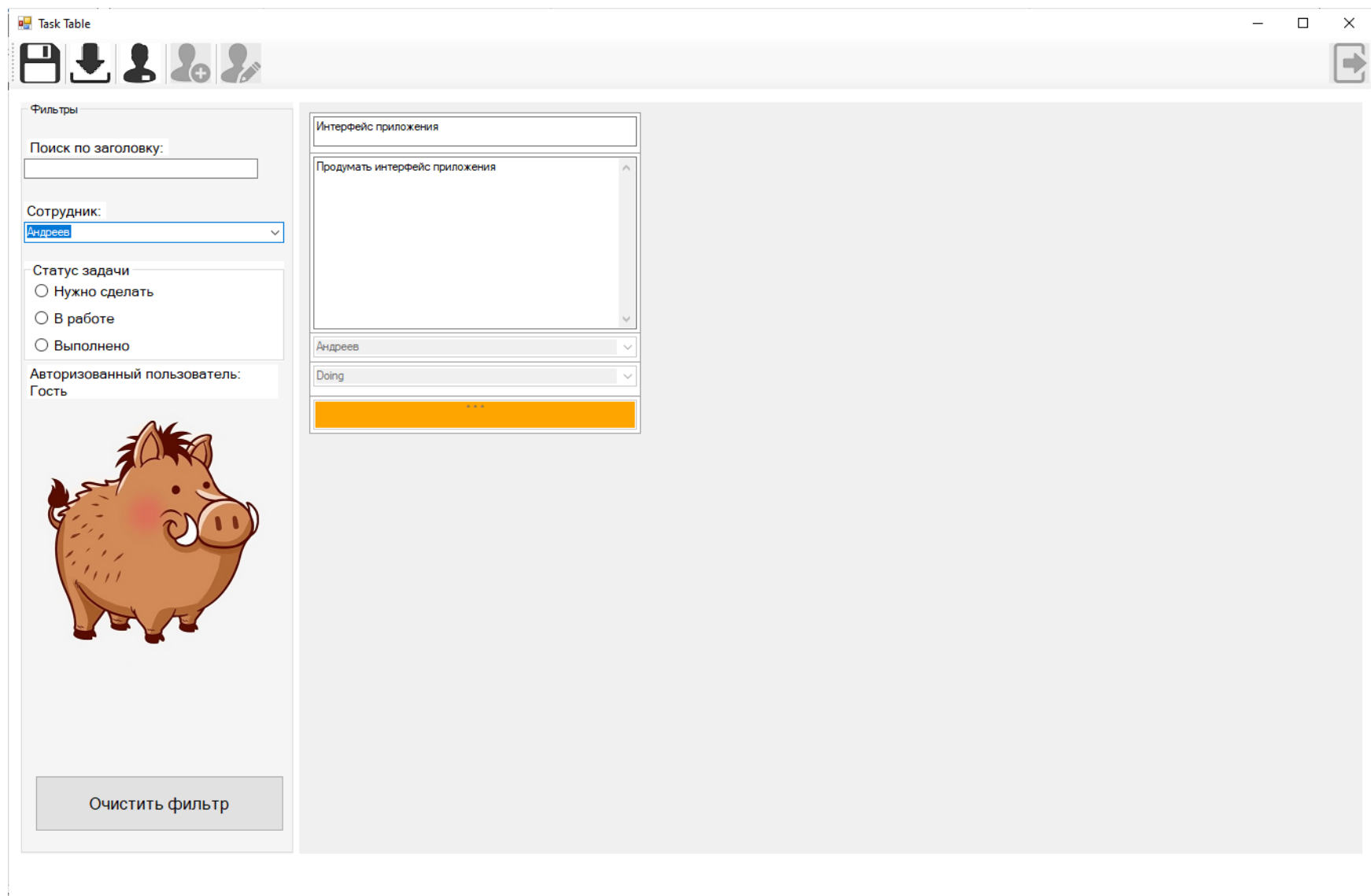


Рисунок П.3.2 – поиск по фамилии работника

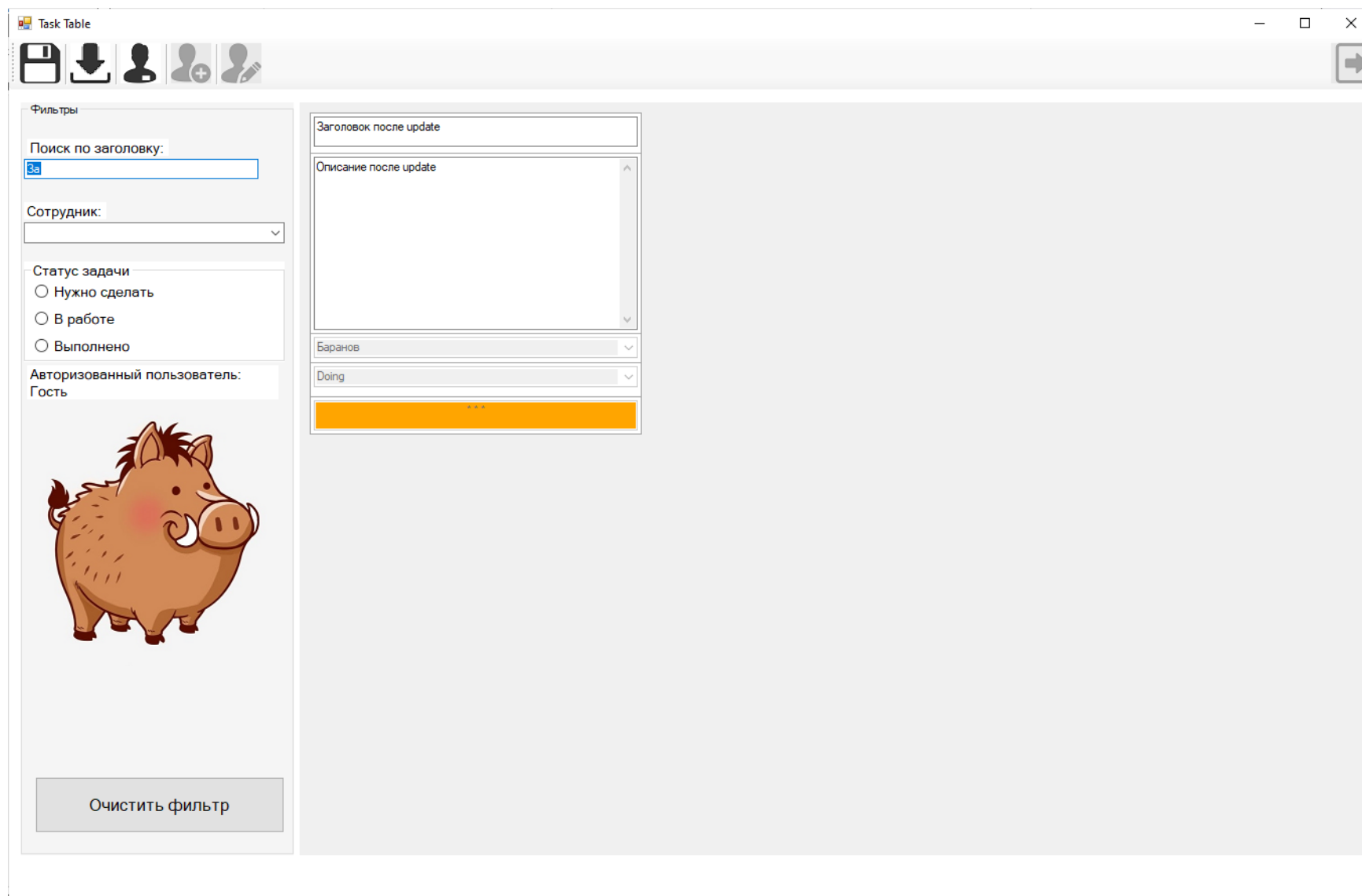


Рисунок П.3.3 – поиск по заголовку задачи

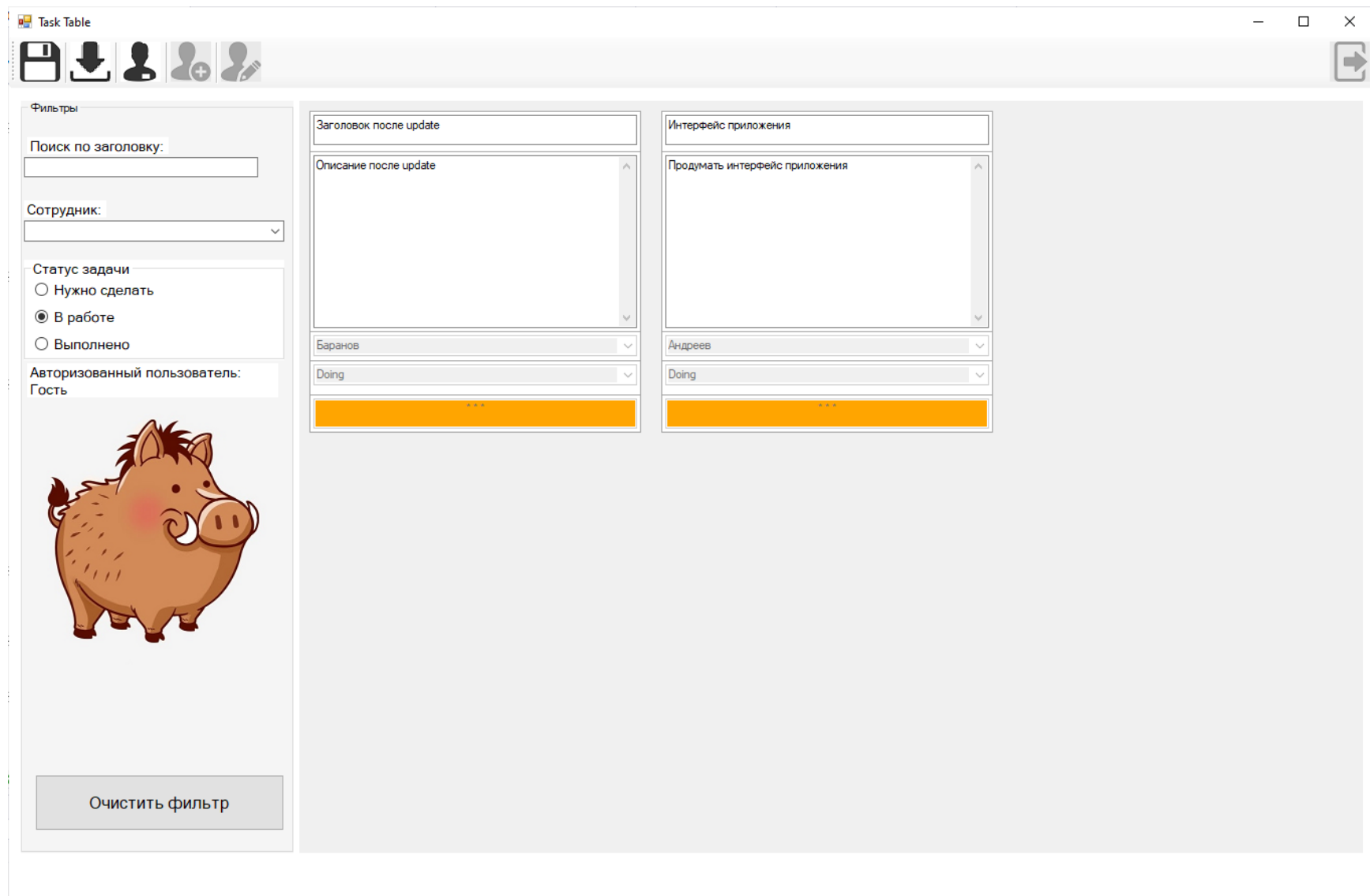


Рисунок П.3.4 – сортировка задач по статусу

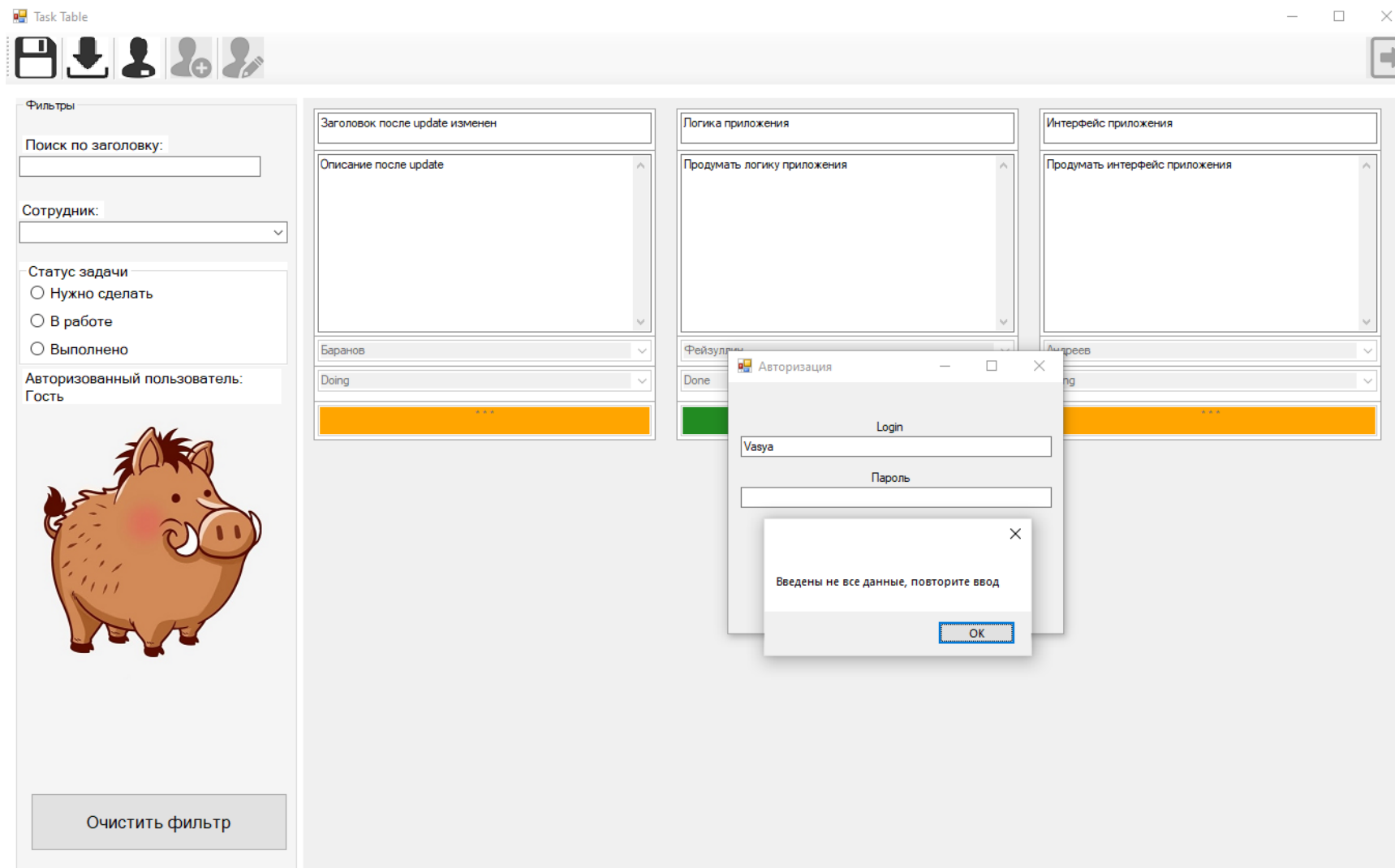


Рисунок П.3.5 – авторизация пользователя

The image displays two side-by-side screenshots of a web application interface for managing workers. Both windows have a standard Windows-style title bar with a minimize button, a maximize button, and a close button.

The left window, titled "Добавление работника" (Add worker), contains the following elements:

- A "Login" label above a text input field containing "NewWorker".
- A "Пароль" (Password) label above a text input field containing "12345".
- A "ФИО" (Full Name) label above a text input field containing "Worker".
- A blue button labeled "Добавить работника" (Add worker) at the bottom.

The right window, titled "Редактирование профиля" (Edit profile), contains the following elements:

- A dropdown menu at the top with "Worker" selected.
- A "Login" label above an empty text input field.
- A "Пароль" (Password) label above an empty text input field.
- A blue button labeled "Удалить" (Delete) at the bottom.

Рисунок П.3.6 – добавление и редактирование профиля работника (доступно администратору)

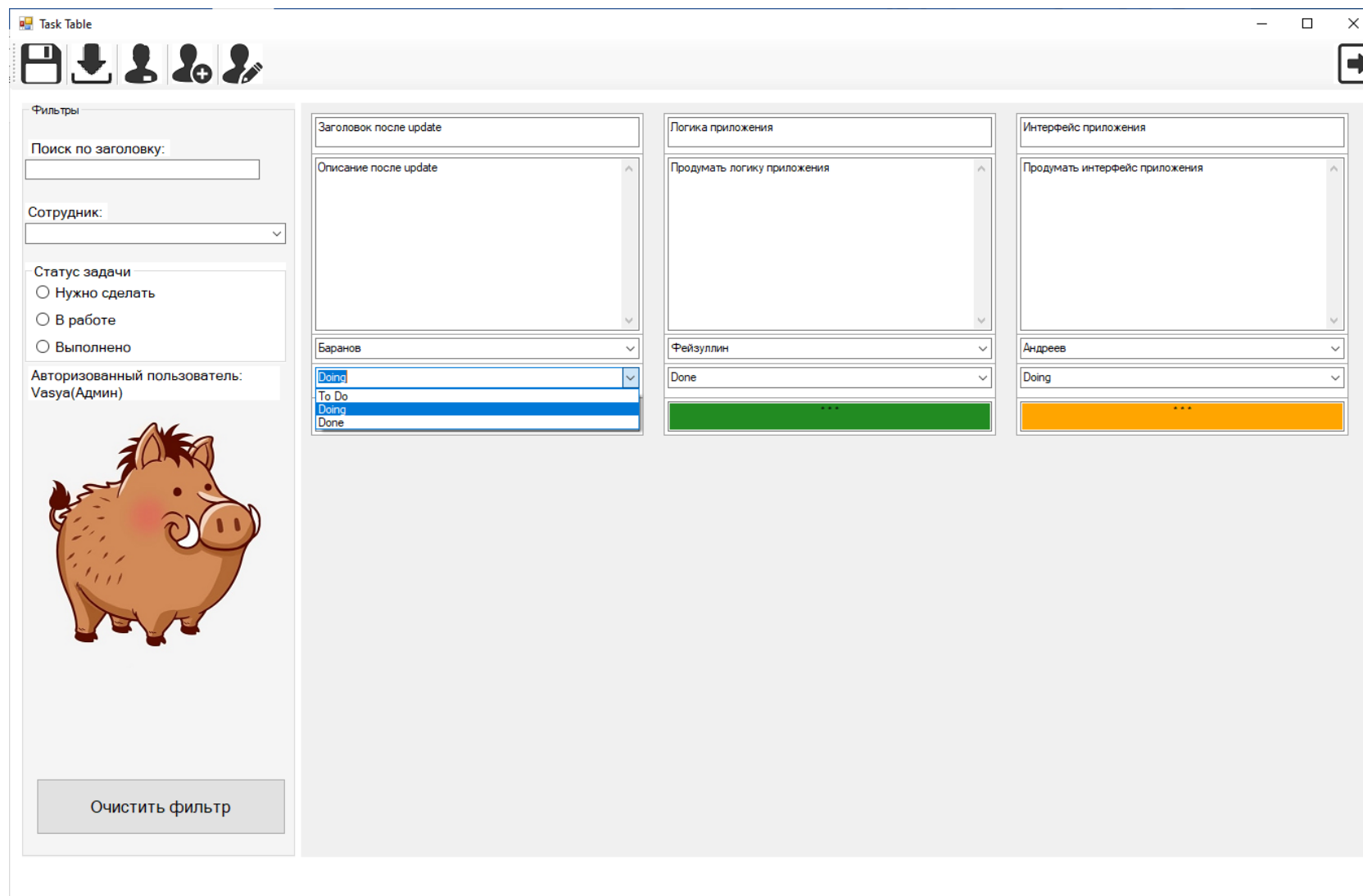


Рисунок П.3.7 – изменение статуса задачи (доступно администратору или сотруднику, работающему над данной задачей)

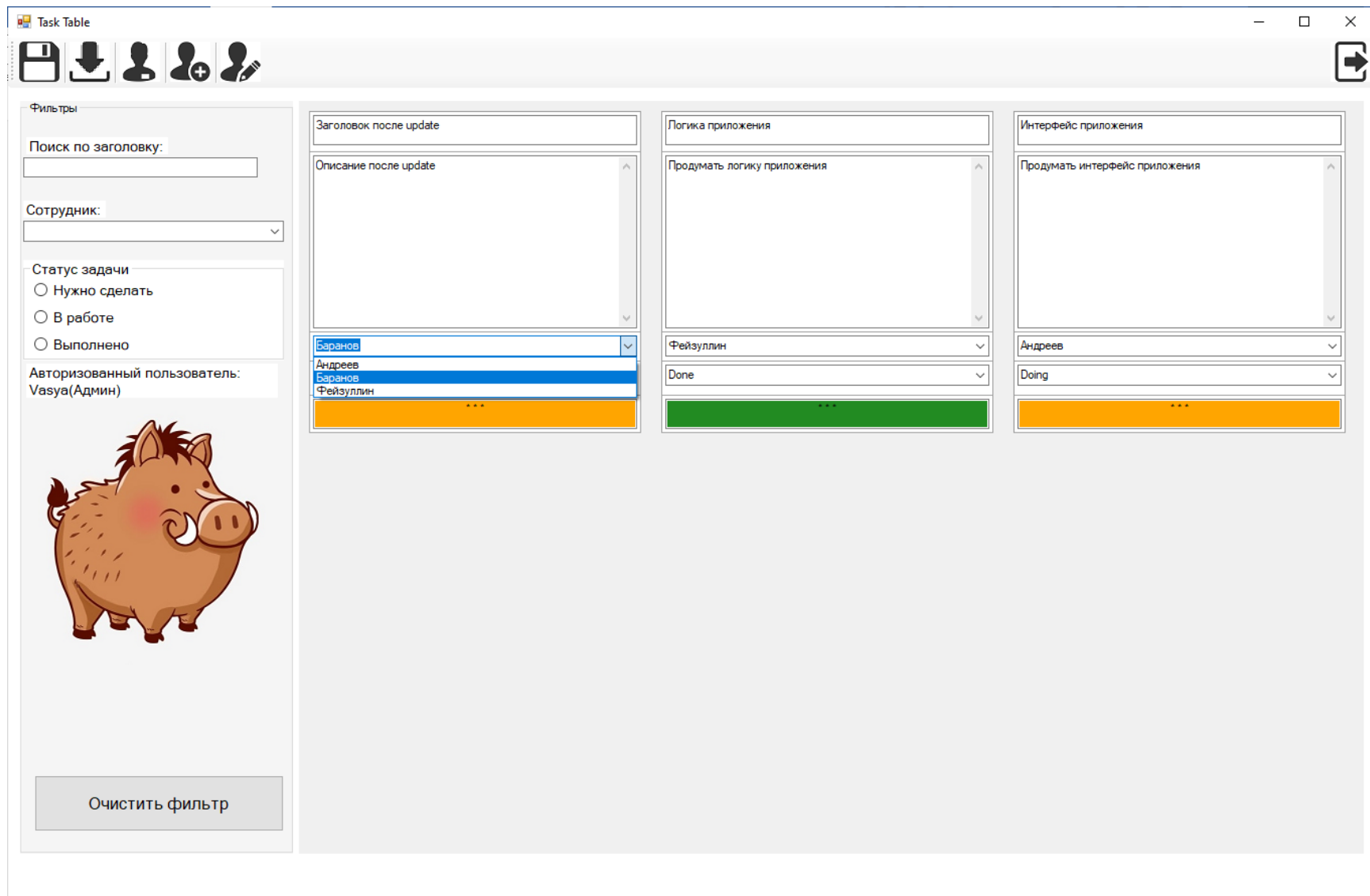


Рисунок П.3.8 – назначение работника на задачу (доступно администратору)

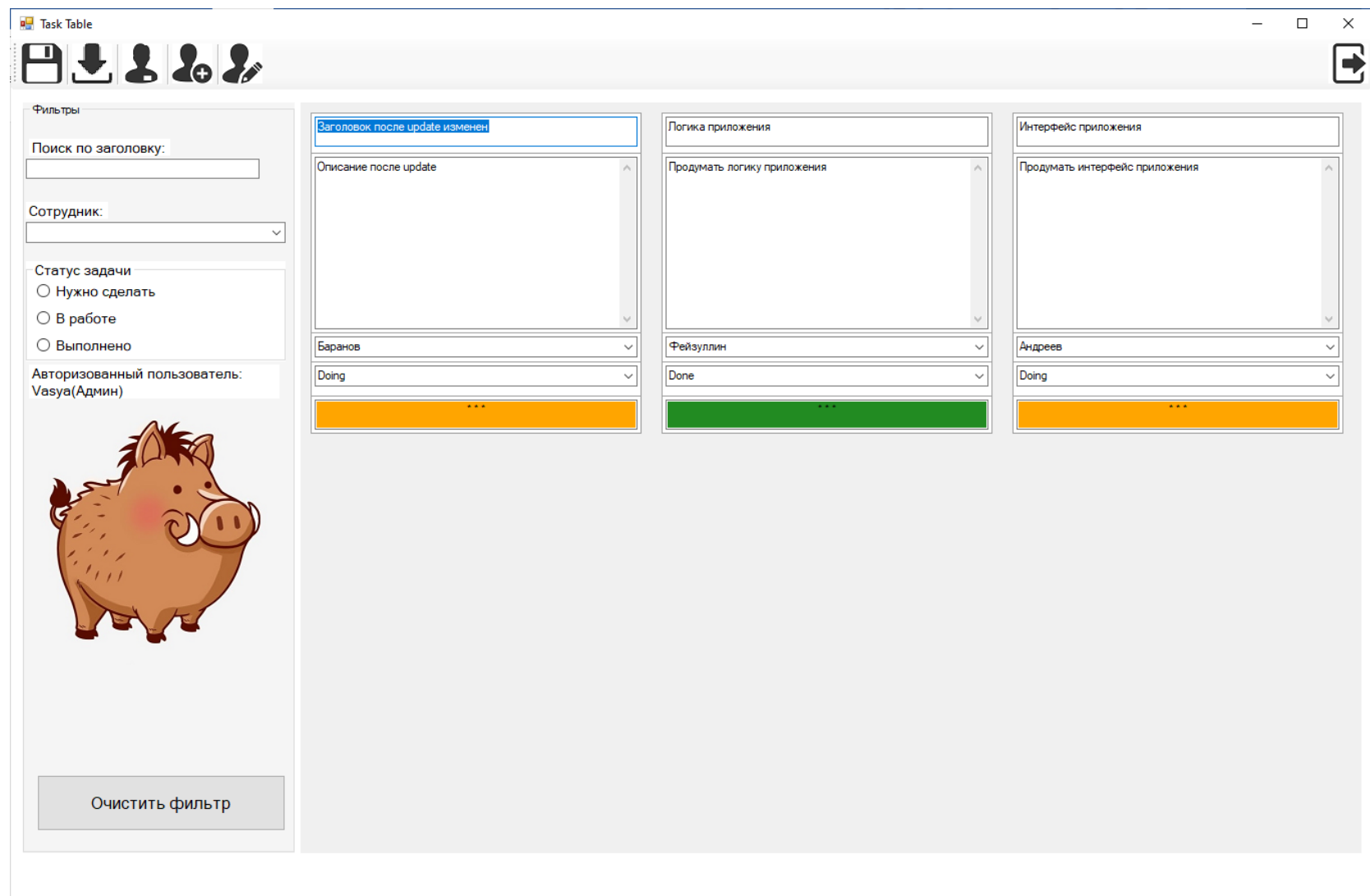


Рисунок П.3.9 – изменение заголовка задачи (доступно администратору)

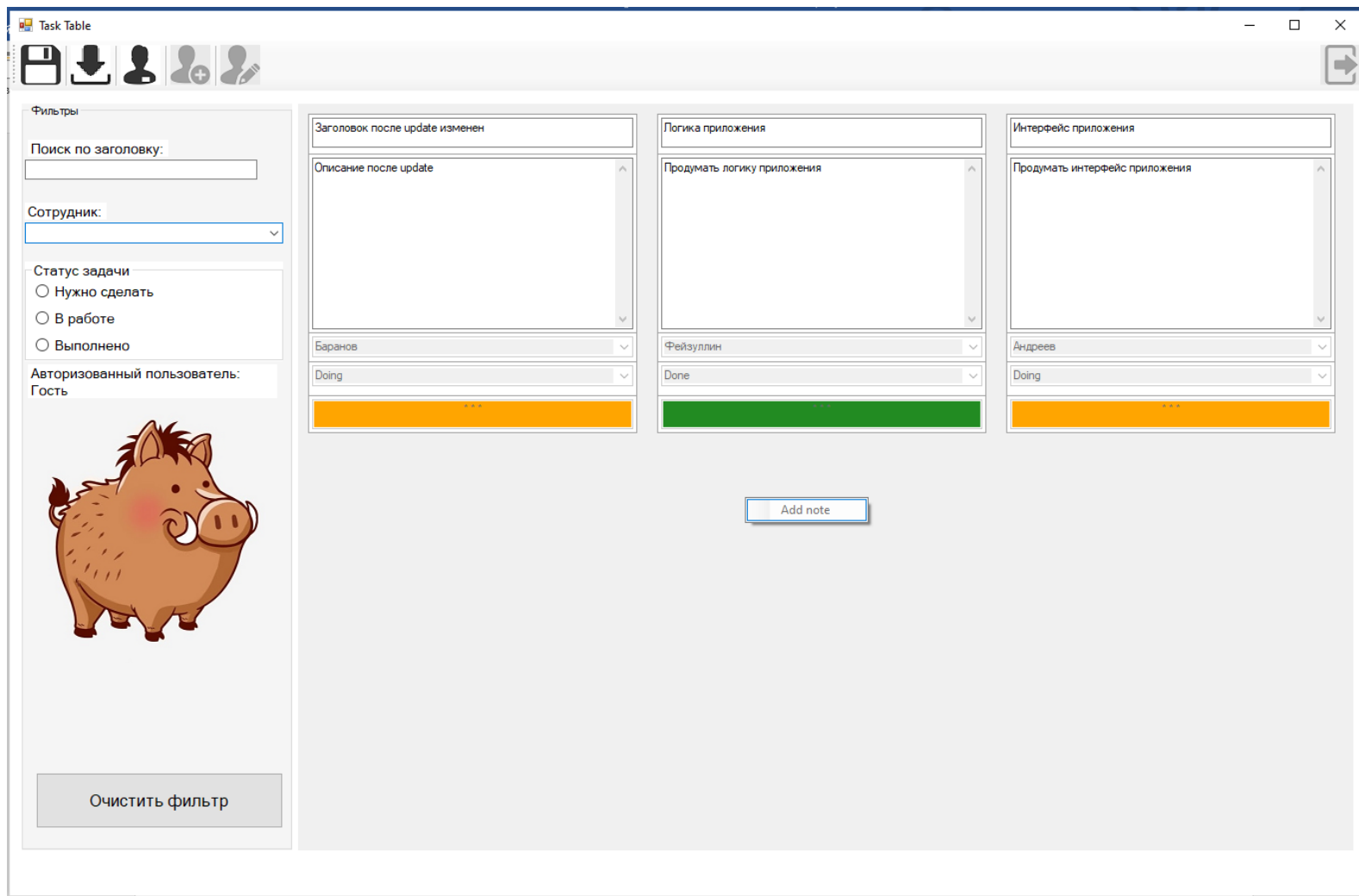


Рисунок П.3.10 – добавление новой задачи (недоступно неавторизованному пользователю)

Task Table

Icons: Save, Download, User, Add User, Edit User, Share

Фильтры

Поиск по заголовку:

Сотрудник:

Статус задачи

- ☐ Нужно сделать
- ☐ В работе
- ☐ Выполнено

Авторизованный пользователь: Vasya(Админ)

Очистить фильтр

Заголовок после update изменен	Логика приложения	Интерфейс приложения
Описание после update	Продумать логику приложения	Продумать интерфейс приложения
Баранов	Фейзуллин	Андреев
Doing	Done	Doing
***	***	***

Add note

Рисунок П.3.11 – добавление новой задачи (доступно администратору)

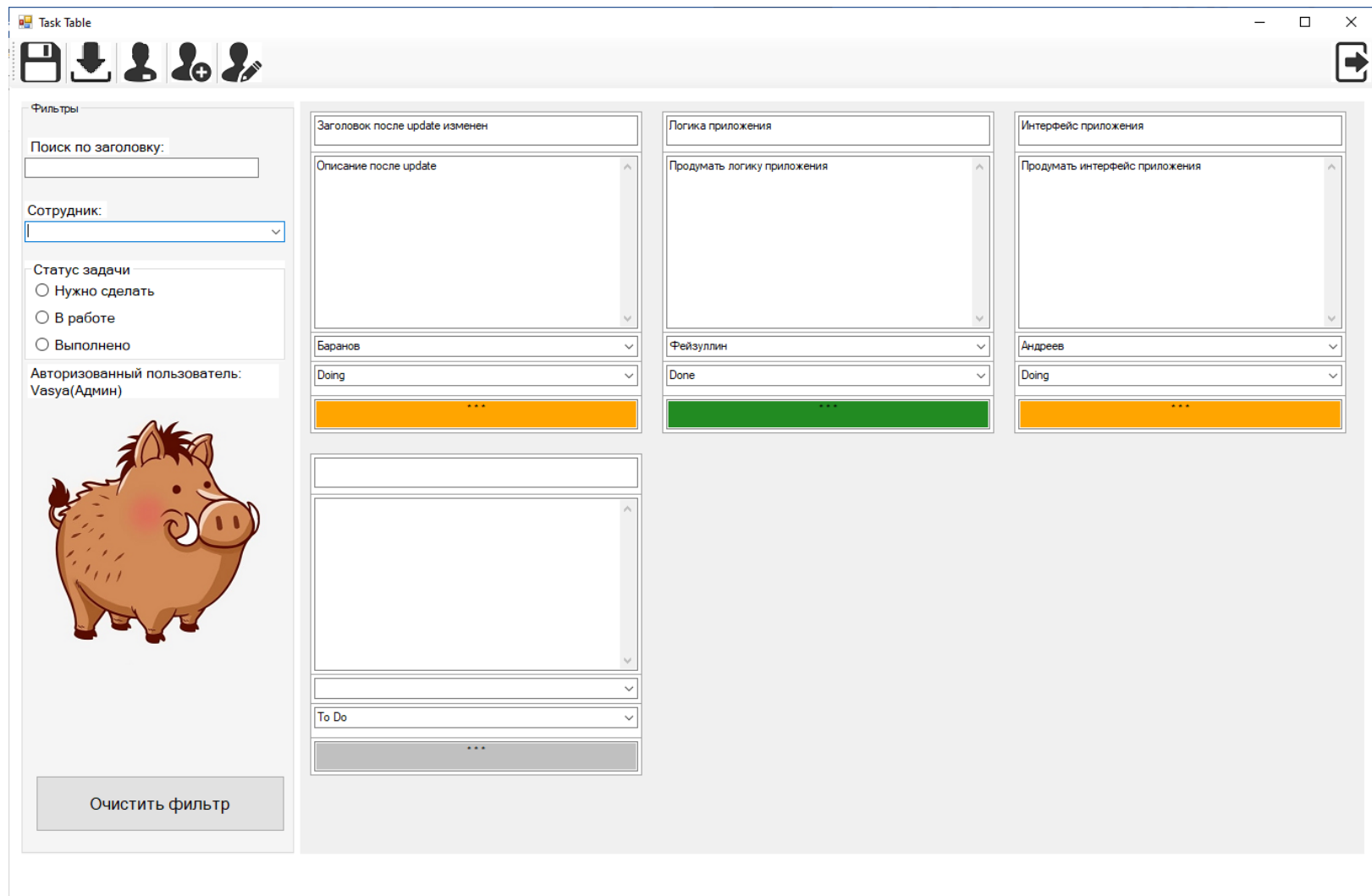


Рисунок П.3.12 – после добавления новой задачи

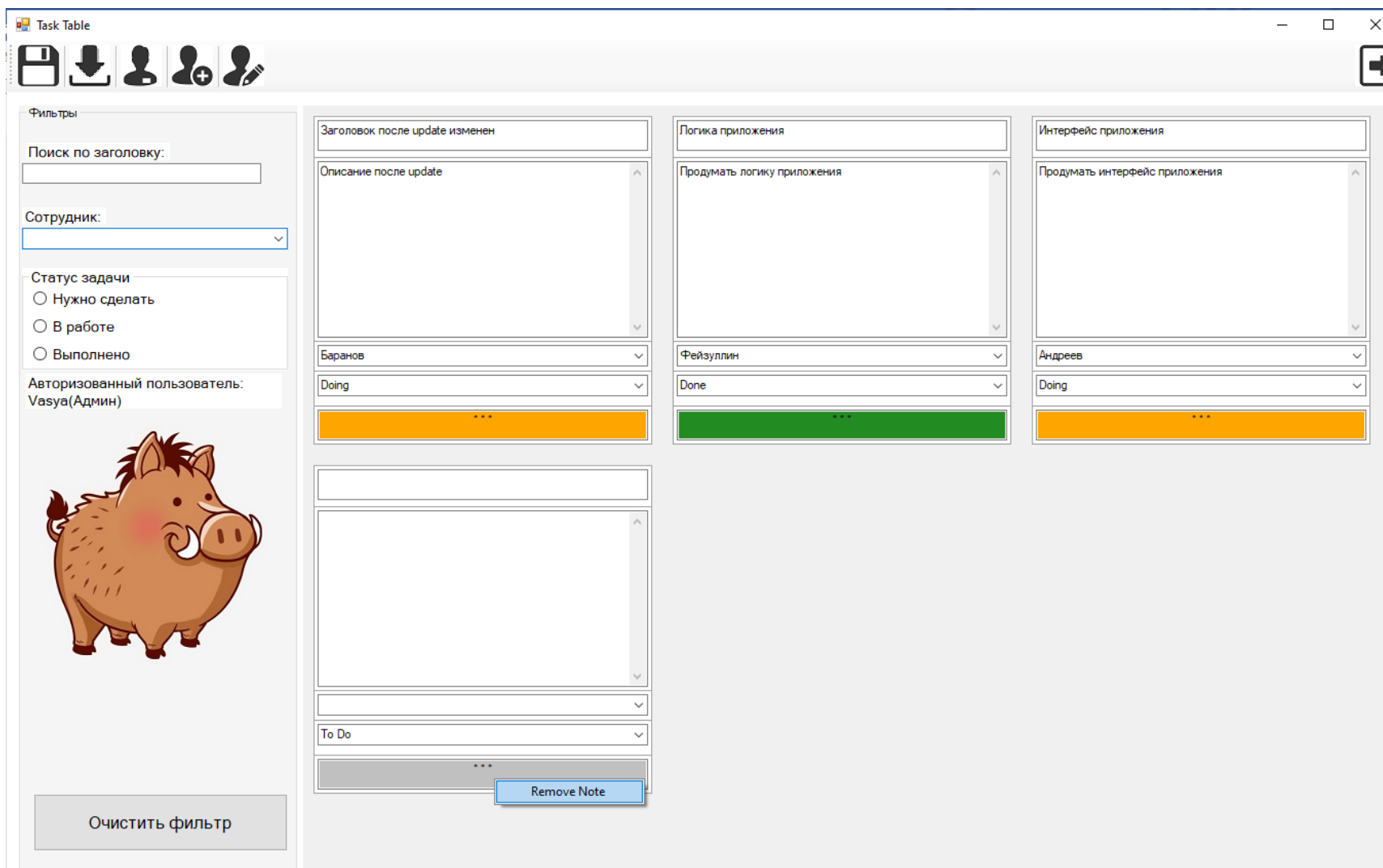


Рисунок П.3.13 – удаление задачи (доступно администратору)

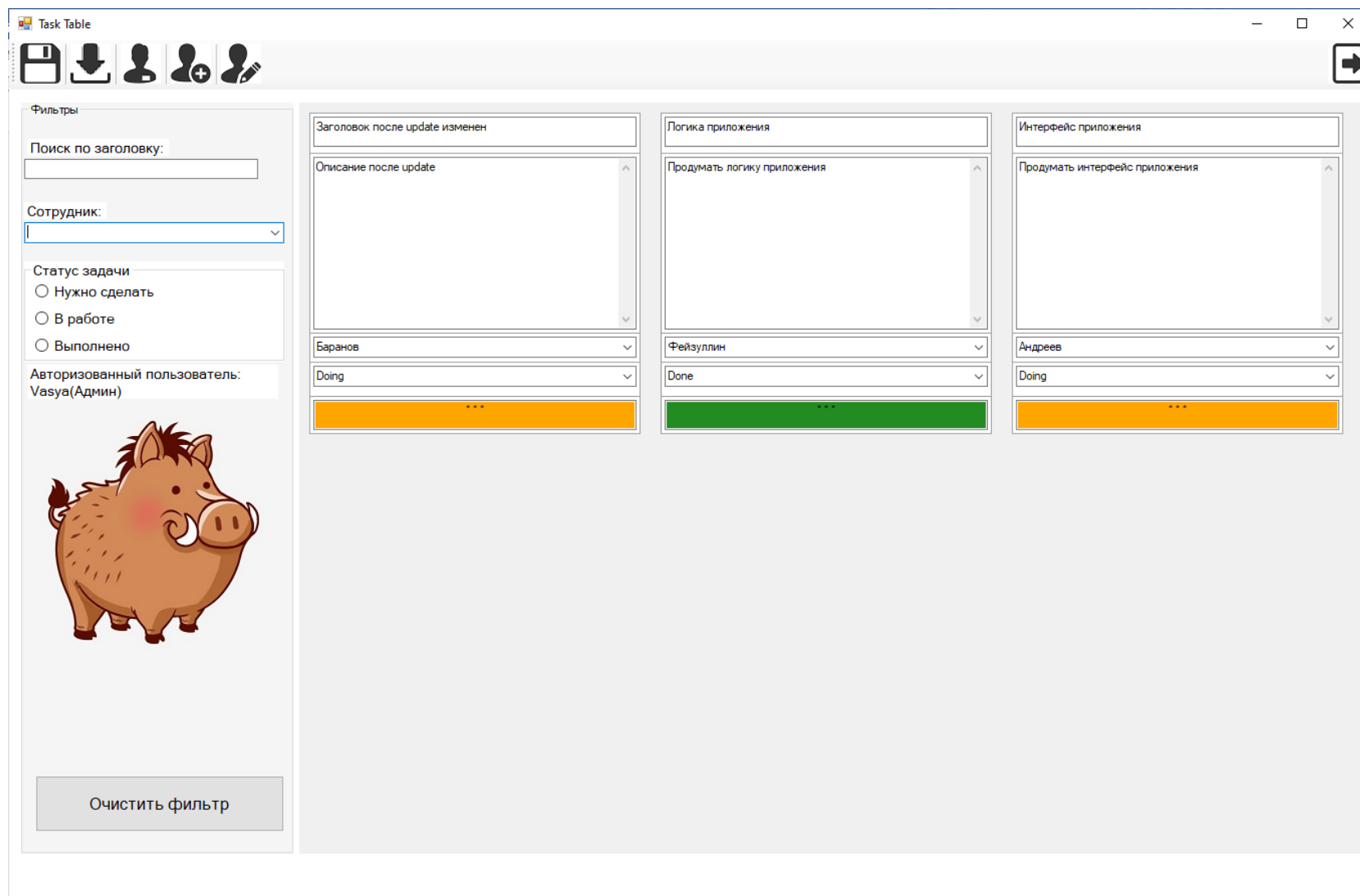


Рисунок П.3.14 – после удаления задачи

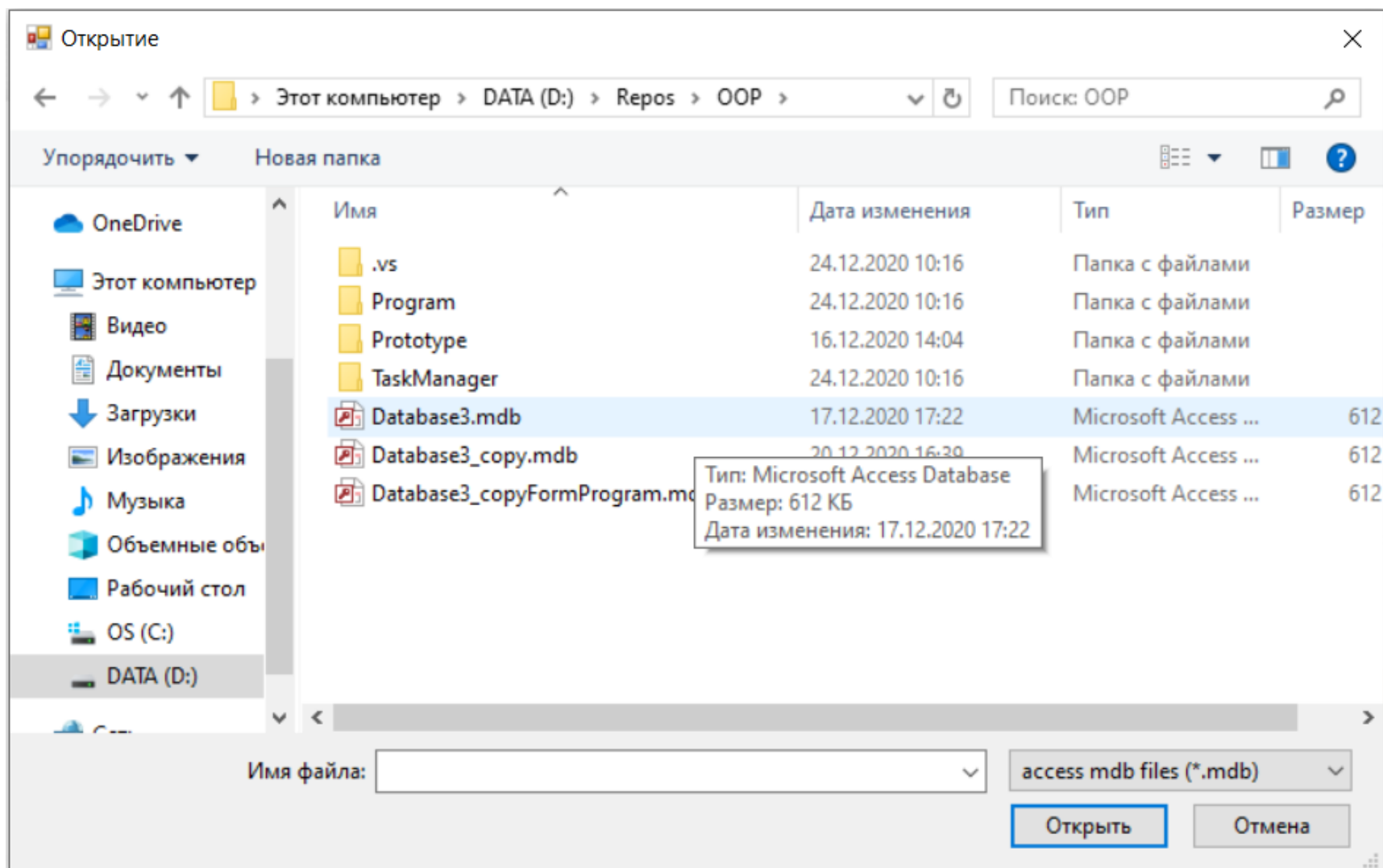


Рисунок П.3.15 – смена базы данных

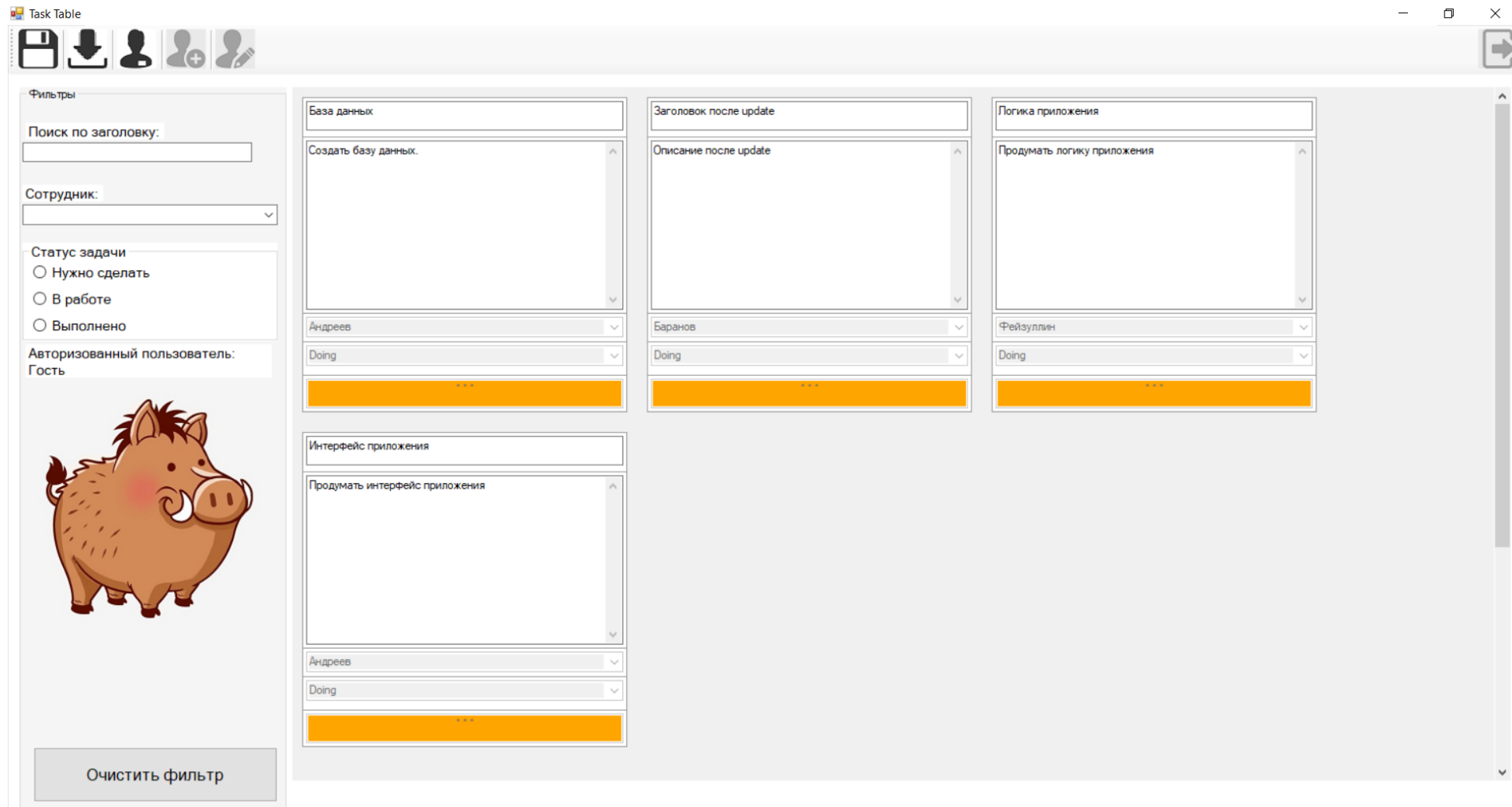


Рисунок П.3.16 – после смены базы данных

Руководство пользователя системы

**Руководство
по эксплуатации программного обеспечения
«Канбан-доска» для пользователя**

Содержание

1. Введение	61
1.1. Назначение	61
1.2. Работа ПО	61
2. Описание ПО	61
2.1. Панель инструментов	62
2.2. Окно вывода карточек	64
2.3. Окно фильтрации	67

1. Введение

В настоящем Руководстве по эксплуатации дано описание и работа с программным обеспечением «Канбан-доска».

1.1 Назначение

По предназначено для выполнения следующих функций:

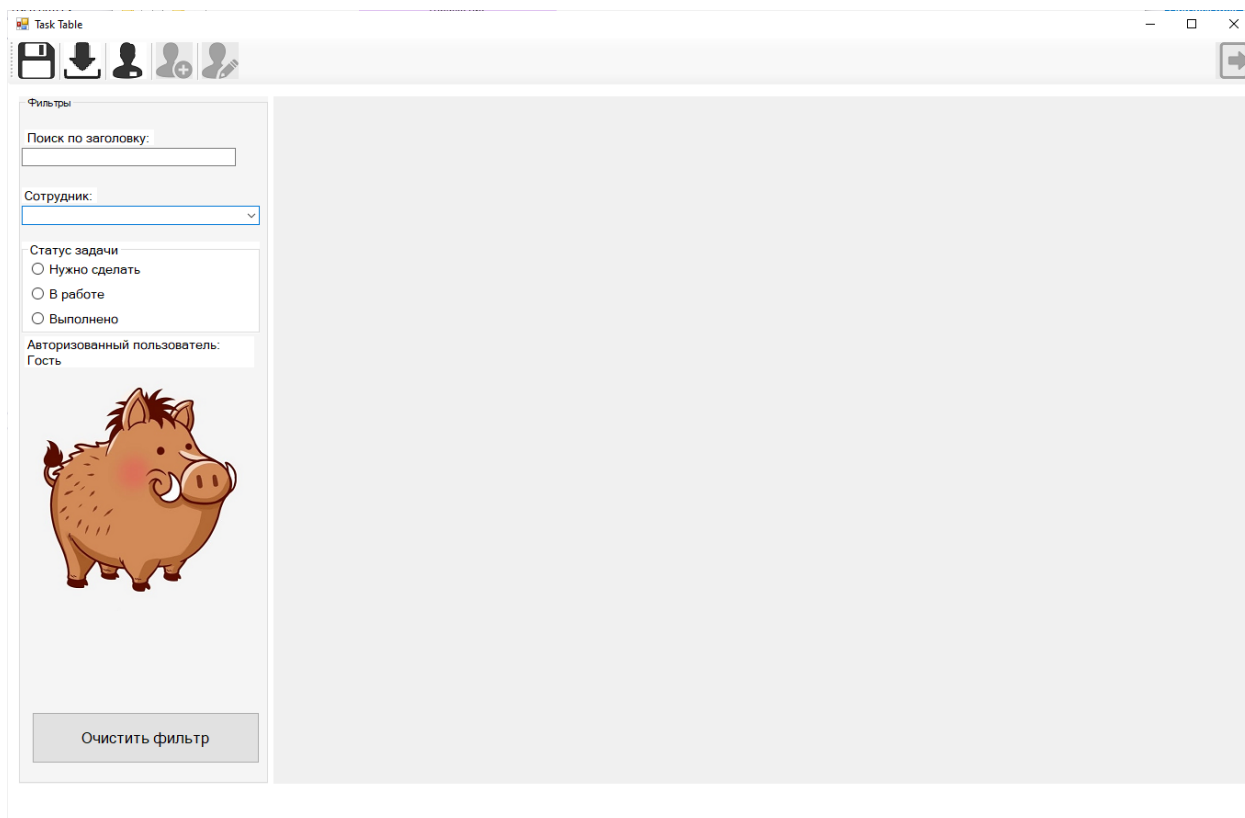
- Визуализация рабочего процесса.
- Мониторинг задач.
- Добавление, удаление, изменение задач.
- Назначение работника на выполнение задачи.

1.2 Работа ПО

По функционирует при наличии установленного пакета Microsoft office (версии 2016 и старше) и под управлением операционных систем Windows (версии Windows 7 и старше).

2. Описание ПО

Внешний вид ПО («основное окно») показан на рисунке:



ПО состоит из следующих частей:

2.1 Панель инструментов



Состоит из 6 кнопок, реализующих следующий функционал:

- 1) Создать или сохранить базу данных – позволяет создать БД из данных, находящихся в данный момент в программе, или перезаписать уже существующую БД



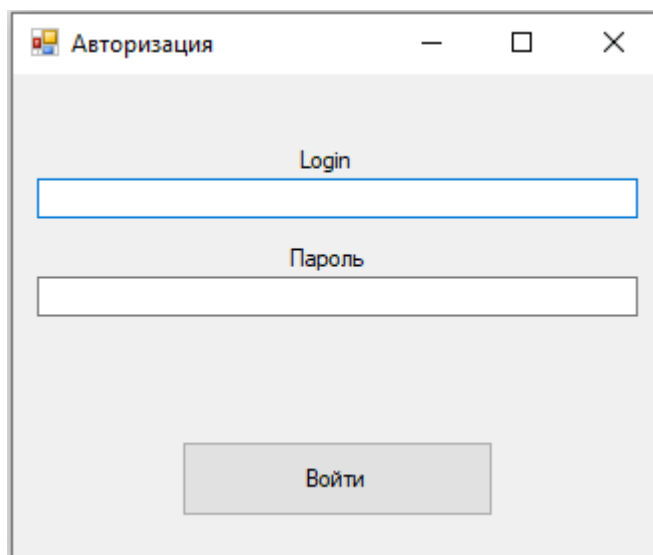
- 2) Загрузить базу данных – позволяет загрузить данные из БД и отобразить их на карточках



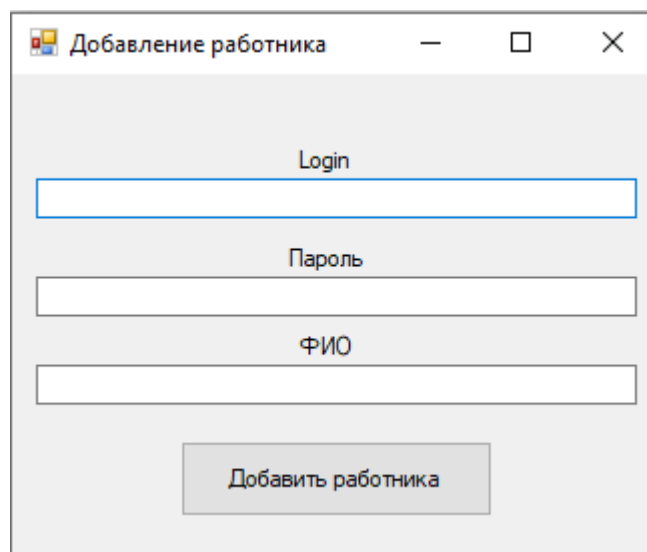
- 3) Авторизоваться – позволяет авторизоваться в системе, не авторизованные пользователи могут только просматривать карточки, они имеют статус гость. Авторизованные пользователи имеют расширенные возможности по взаимодействию с программой, они могут иметь два статуса:

- Администратор
- Работник



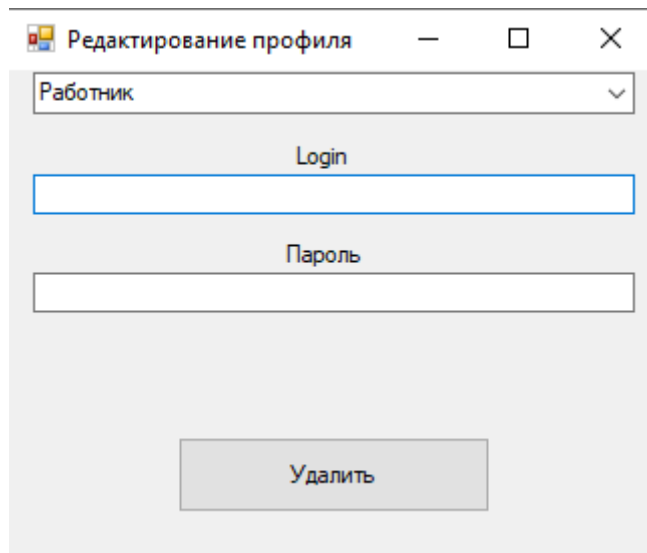


- 4) Добавить работника – позволяет зарегистрировать в системе нового работника, создать ему логин и пароль. Данная кнопка доступна только для администратора.



- 5) Редактировать пользователя – позволяет удалить выбранного пользователя, удалять данные можно только для пользователей со статусом работник. Данная кнопка доступна только для администратора.





Профиле редактирование

Работник

Login

Пароль

Удалить

- 6) Выйти из учетной записи – позволяет выйти из данного профиля, и перейти к статусу гость. Данная кнопка доступна только для авторизованных пользователей.



2.2 Окно вывода карточек

Окно вывода карточек позволяет просматривать задачи и их статусы, если пользователь авторизован, то он сможет редактировать карточки.

Заголовок после update	Логика приложения	Интерфейс приложения
Описание после update	Продумать логику приложения	Продумать интерфейс приложения
Баранов	Фейзуллин	Андреев
Doing	Done	Doing
...

Карточка имеет следующие поля:

- Заголовок задачи – в этом поле отображается название задачи. Это поле могут редактировать администраторы.

Логика приложения

- Описание задачи – в этом поле отображается сама задача, ее описание. Это поле могут редактировать администраторы.

Продумать логику приложения

- **Работник** – в этом поле отображается имя работника, который выполняет, будет выполнять или выполнил данную задачу. Это поле может редактировать администратор.

 A horizontal dropdown menu with a light gray background. The text 'Фейзуллин' is displayed in a standard font. A small downward-pointing arrow is visible on the right side of the menu.

- **Статус** – в этом поле отображается статус задачи, его могут изменять администратор и работник, за которым закреплена данная задача.

Статус может принимать три вариации:

- 1) To do – необходимо сделать

 A horizontal dropdown menu with a light gray background. The text 'To Do' is displayed in a standard font. A small downward-pointing arrow is visible on the right side of the menu.

- 2) Doing – находится в процессе

 A horizontal dropdown menu with a light gray background. The text 'Doing' is displayed in a standard font. A small downward-pointing arrow is visible on the right side of the menu.

- 3) Done – сделано

 A horizontal dropdown menu with a light gray background. The text 'Done' is displayed in a standard font. A small downward-pointing arrow is visible on the right side of the menu.

- **Индикатор** – это поле отображает цветом статус задачи, для администратора это поле также является кнопкой удаления карточки.

- 1) To do

 A horizontal rectangular bar with a gray background. In the center, there are three small red dots. The bar has a thin border.

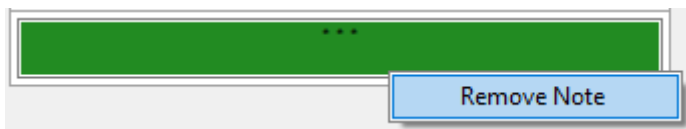
- 2) Doing

 A horizontal rectangular bar with an orange background. In the center, there are three small red dots. The bar has a thin border.

- 3) Done

 A horizontal rectangular bar with a green background. In the center, there are three small red dots. The bar has a thin border.

При нажатии на индикатор в случае наличия статуса администратора появляется всплывающее меню Remove Note позволяющая удалить карточку.



2.3 Окно фильтрации

Окно фильтрации позволяет искать задачи по некоторым признакам, так же здесь расположена информация о статусе авторизации.

Фильтры

Поиск по заголовку:

Сотрудник:

Статус задачи


☐ Нужно сделать

☐ В работе

☐ Выполнено

Авторизованный пользователь:

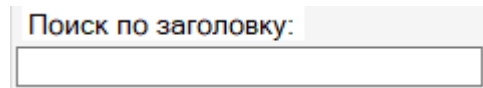
Гость



Очистить фильтр

Окно фильтрации имеет следующие поля:

- Поиск по заголовку – позволяет отсортировать карточки с совпадающим заголовком с введённым в данное поле текстом, и вывести их в «окно вывода карточек».



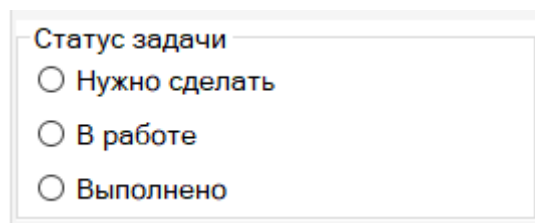
Поиск по заголовку:

- Сотрудник – позволяет отсортировать карточки по выбранному в данном поле работнику, и вывести их в «окно вывода карточек».



Сотрудник:

- Статус задачи – позволяет отсортировать карточки по выбранному в данном поле статусу, и вывести их в «окно вывода карточек»



Статус задачи

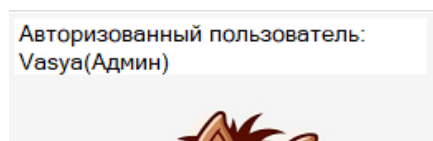
☐ Нужно сделать

☐ В работе

☐ Выполнено

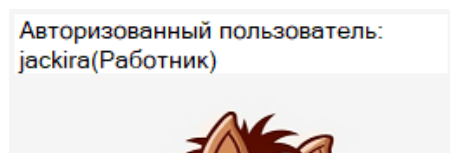
- Статус авторизации – в этом поле отображается статус авторизации в данный момент, их может быть три вариации:

1) Администратор



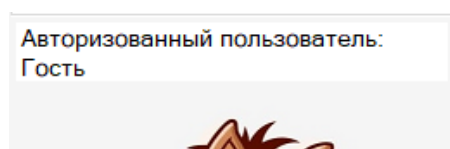
Авторизованный пользователь:
Vasya(Админ)

2) Работник



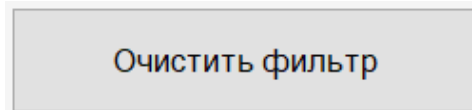
Авторизованный пользователь:
jaskira(Работник)

3) Гость



Авторизованный пользователь:
Гость

- Очистить фильтр – данная кнопка приводит поля «поиск по заголовку», «сотрудник», «статус задачи» в стандартное состояние, очищая их. При этом «окно вывода карточек» отобразит все карточки, находящиеся в системе.



ПРИЛОЖЕНИЕ 5

Руководство администратора системы

Руководство по эксплуатации программного обеспечения «Канбан-доска» для администратора системы

Содержание

1. Введение	72
1.1. Назначение	72
1.2. Работа ПО	72
2. Начало работы	72
3. Работа в приложении	73

1. Введение

В настоящем Руководстве по эксплуатации дано описание и работа с программным обеспечением «Канбан-доска» для администратора системы

1.1 Назначение

По предназначено для выполнения следующих функций:

- Визуализация рабочего процесса.
- Мониторинг задач.
- Добавление, удаление, изменение задач.
- Назначение работника на выполнение задачи.

1.2 Работа ПО

По функционирует при наличии установленного пакета Microsoft office(версии 2016 и старше), и под управлением операционных систем.

2. Начало работы

Для начала работы в системе приложения как администратор, вам понадобится чистая база данных, поставляемая вместе с приложением.

База данных включает в себя готовую структуру данных, где администратору нужно лишь изменить свои данные в таблице «Сотрудники»

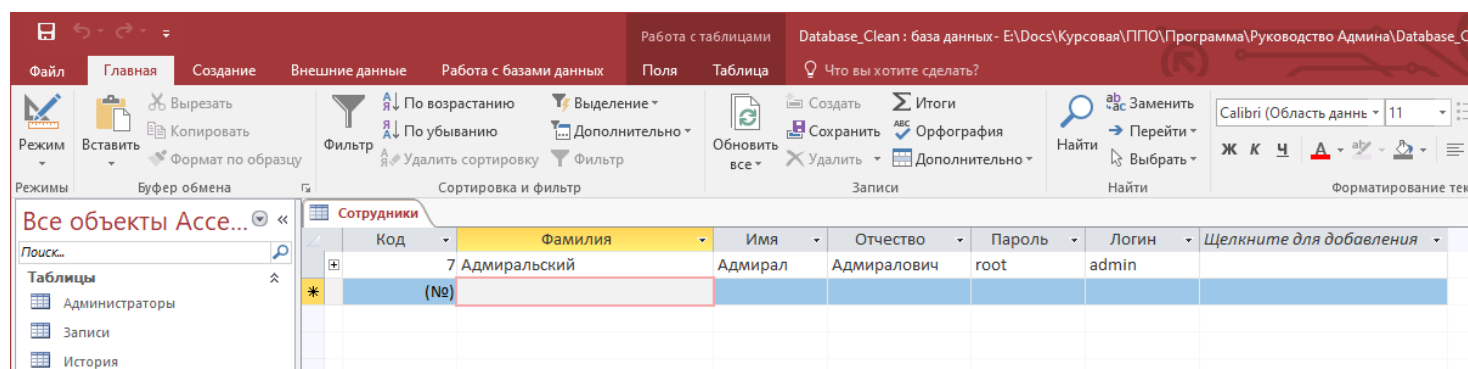


Рисунок 11

для входа и отображения, если он того хочет. Данная таблица представлена на рисунке 1.

На этом работа с базой данных непосредственно напрямую окончена. Можно перейти в приложение и начать работу.

3. Работа в приложении

Открываем приложение (рисунок 2).

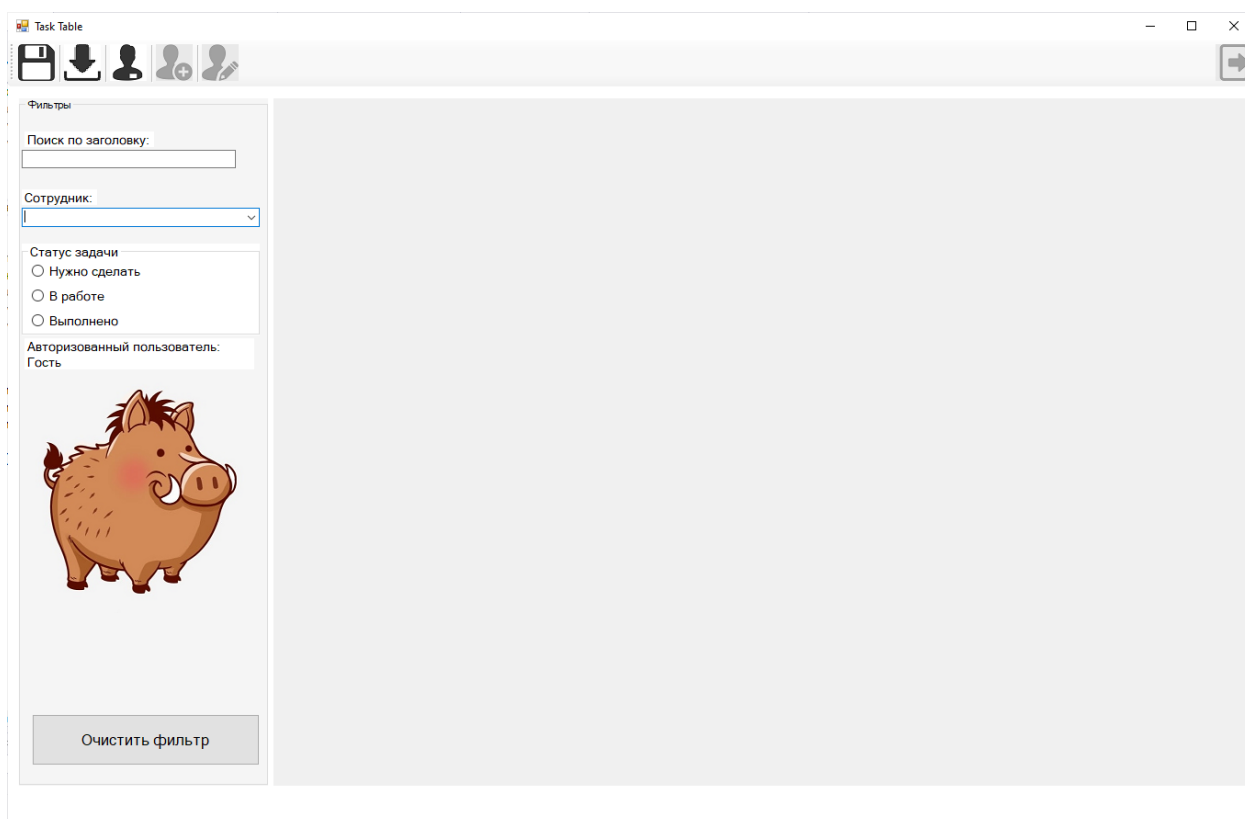


Рисунок 12

Нажимаем на кнопку загрузки базы (рисунок 3).

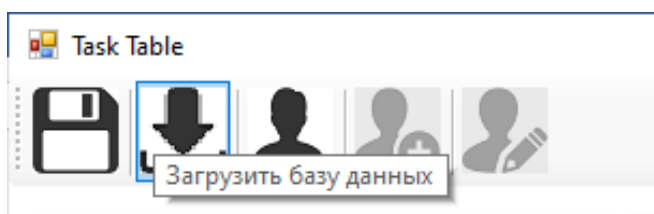


Рисунок 13

Выбираем нашу пустую базу данных (рисунок 4).

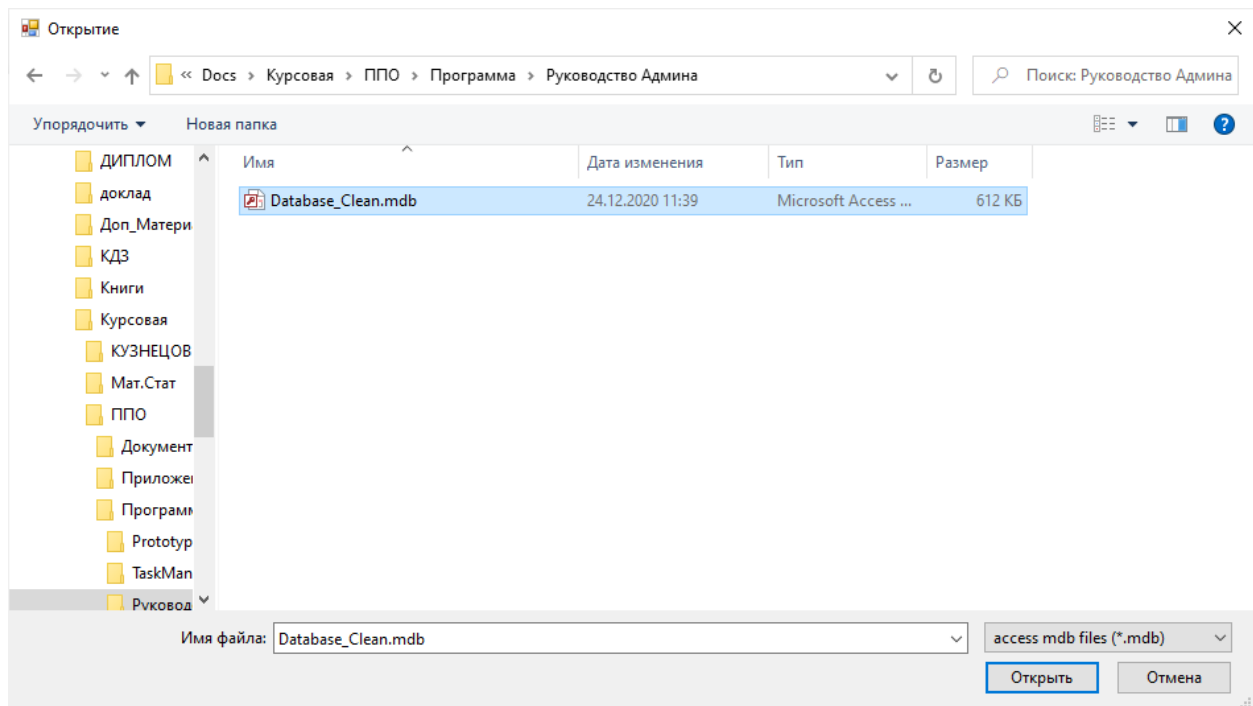


Рисунок 14

Далее авторизуемся, вызвав окно входа, нажав на кнопку авторизации (рисунок 5).

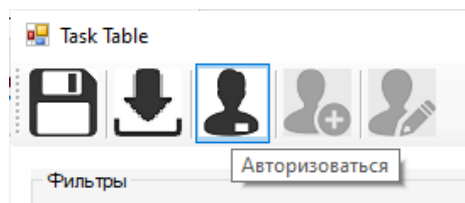


Рисунок 15

Вводим свои данные в окне (рисунок 6).

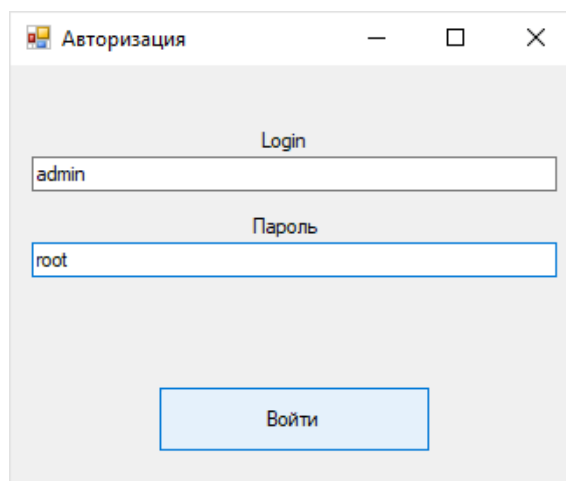


Рисунок 6

Теперь мы авторизовались как администратор, о чем нам сообщит окно статуса пользователя (рисунок 7).

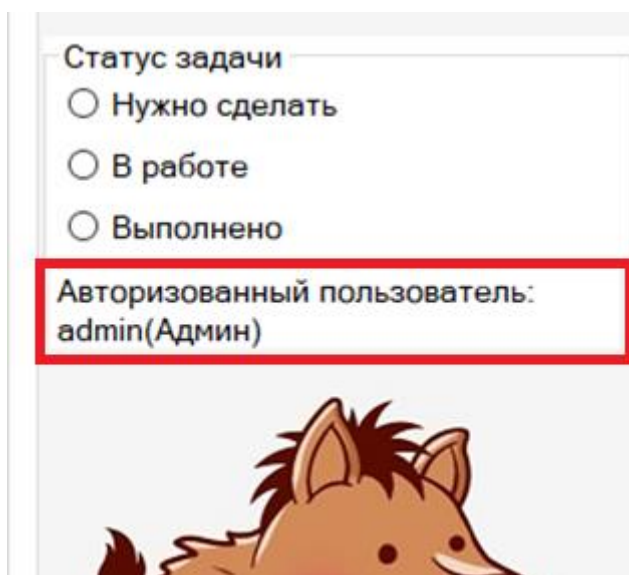


Рисунок 16

Дальнейшая работа в системе изложена в руководстве пользователя.