



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

**«МИРЭА – Российский технологический**

**университет» РТУ МИРЭА**

**Колледж программирования и кибербезопасности**

Специальность 09.02.07 Информационные системы и программирование

**Практическая работа №10**

**«МДК.01.01. Разработка программных модулей»**

Выполнил студент

группы ЩПКО-01-22 (ПКС-31)

\_\_\_\_\_ Шакиров А.А.

подпись

ФИО студента

Преподаватель

\_\_\_\_\_ Стоколос.М.Д.

подпись

ФИО преподаватель

Москва

2024

## ОГЛАВЛЕНИЕ

«МИРЭА – Российский технологический университет» РТУ МИРЭА.....	1
Практическая работа №10.....	1
«МДК.01.01. Разработка программных модулей».....	1
ВВЕДЕНИЕ.....	3
1. Анализ существующих разработок.....	4
2. Анализ предметной области выбранной тематики.....	5
2.1. Описание предметной области.....	5
2.2. Ключевые задачи предметной области.....	7
2.3. Основные бизнес-процессы предметной области:.....	7
2.3. Бизнес-процесс для автоматизации.....	8
3. Обзор инструментальных средств.....	9
3.1. Язык программирования: C#.....	9
3.2. Фреймворк: WPF (Windows Presentation Foundation).....	10
3.3. Инструментальное средство разработки: Visual Studio.....	11
3.4. Система управления базами данных PostgreSQL.....	11
4. Описание функционала.....	12
4.1. Окно входа.....	12
ЗАКЛЮЧЕНИЕ.....	18
Список использованных источников.....	19

## **ВВЕДЕНИЕ**

Студент группы ПКС-31 Шакиров Артём Андреевич проходил практику в организации – Колледж программирования и кибербезопасности по адресу: г. Москва, 1-й Щипковский переулок, д 23.

Целью практической работы практики является получение первичных профессиональных умений и навыков, закрепление и углубление теоретической подготовки по направлению «Разработать приложение с использованием платформы WPF».

Для достижения поставленной цели были определены следующие задачи:

- Последовательный пользовательский интерфейс, позволяющий перемещаться между существующими окнами в приложении (в том числе обратно, например, с помощью кнопки «Назад»);
- соответствующий заголовок на каждом окне приложения;
- допустимо использование не более одной команды в строке;
- использовать комментарии для пояснения неочевидных фрагментов кода.

## 1. Анализ существующих разработок

Для анализа существующих разработок были выбраны следующие приложения:

Приложение "Обработка заявок на ремонт оборудования" предназначено для учета заявок: ФИО, Поломка, Состояние. Оно позволяет контролировать заявки на ремонт.

Достоинства данного приложения:

- Удобство использования: Простота интерфейса и навигации, которые делают приложение доступным для пользователей.
- Производительность: Быстрая загрузка и высокая скорость работы без сбоев.

А также представлены недостатки:

- Безопасность данных: Возможны риски утечки личных данных пользователей.

Google Keep — это приложение для создания заметок и хранения информации, которое предлагает пользователям множество возможностей. Вот его достоинства и недостатки:

**Достоинства:**

- **Простота использования:** Интуитивно понятный интерфейс позволяет быстро создавать и организовывать заметки.
- **Многофункциональность:** Поддерживает текстовые заметки, списки, изображения и голосовые записи, что делает его универсальным инструментом для хранения информации.
- **Синхронизация:** Заметки автоматически синхронизируются между устройствами, что позволяет получать доступ к ним в любое время и в любом месте.
- **Совместная работа:** Возможность делиться заметками с другими

пользователями и совместно редактировать их.

- **Напоминания:** Функция установки напоминаний помогает не забыть важные дела и задачи.

### **Недостатки:**

- **Ограниченные функции форматирования:** В сравнении с другими приложениями для заметок, Google Keep предлагает ограниченные возможности форматирования текста.
- **Зависимость от интернета:** Хотя приложение работает в оффлайн-режиме, для синхронизации и доступа к некоторым функциям требуется интернет-соединение.
- **Отсутствие структурированности:** Заметки могут быстро накапливаться, и без системы папок или категорий их может быть сложно организовать.
- **Ограниченное количество функций:** Для более сложных задач может потребоваться использование других приложений, так как Google Keep не поддерживает некоторые продвинутые функции.
- **Проблемы с поиском:** Поиск по заметкам может быть не всегда эффективным, особенно если у вас много записей.

## **2. Анализ предметной области выбранной тематики**

### **2.1. Описание предметной области**

Предметная область для разработки программного продукта:

Основная цель учёта заявок на ремонт оборудования - эффективное и оперативное осуществление ремонтных работ с минимизацией простоев и удовлетворением запросов клиентов или сотрудников. Эта предметная область широко используется в различных сферах деятельности,

таких как сервисные услуги, производство, информационные технологии и другие.

3Предметная область учёта заявок на ремонт оборудования касается процесса подачи, обработки и учёта заявок на ремонт различного оборудования.

В данной области включены следующие основные составляющие:

1. Заявка на ремонт: это информация, предоставленная клиентом или сотрудником о неисправности оборудования, которое требует ремонта. Заявка может содержать данные о типе оборудования, его серийном номере, описании проблемы и другой важной информации.

2. Регистрация заявки: этот процесс включает приём и регистрацию заявки в системе учёта. Важными аспектами регистрации являются присвоение уникального идентификатора заявке, сохранение информации о заявке и её приоритете.

3. Обработка заявки: процесс, включающий состояние и назначение исполнителя (ремонтного специалиста) для задачи. В процессе обработки может потребоваться дополнительная информация или уточнение деталей проблемы у клиента или сотрудника

4. Исполнение заявки: фактическое выполнение ремонта оборудования. В этом этапе назначенный исполнитель ремонтирует оборудование, вносит необходимые изменения или заменяет неисправные компоненты.

Важно отметить, что на этом этапе могут возникать необходимость заказа запчастей или координации работ с другими специалистами.

5. Отчётность и информирование: важной составляющей учёта заявок на 4 ремонт является фиксация и отчёт о выполненной работе. После

завершения ремонта, исполнитель должен предоставить отчёт о

проделанной работе, включая информацию о затраченных ресурсах (время, материалы, стоимость), причине неисправности и оказанной помощи.

6. Мониторинг и анализ: этот этап предполагает контроль и анализ процесса учёта заявок на ремонт. Важно отслеживать и анализировать время обработки заявок, качество выполненных работ, расходы и прочие параметры, которые могут помочь в оптимизации и улучшении процесса.

## **2.2. Ключевые задачи предметной области**

Ключевыми задачами предметной области "Обработка заявок на ремонт оборудования" являются:

- Автоматизация процессов: Упрощение и автоматизация учета заявок, что минимизирует ручной ввод данных, снижает вероятность ошибок и повышает скорость доступа к информации.
- Прозрачность и контроль: Обеспечение полной видимости и контроля над заявками, а также их статусом (в ожидании в работе готово).

## **2.3. Основные бизнес-процессы предметной области:**

В рамках предметной области "Обработка заявок на ремонт оборудования" можно выделить не сколько ключевых бизнес-процессов, требующих автоматизации:

Регистрация заявки: этот процесс включает приём и регистрацию заявки в системе учёта. Важными аспектами регистрации являются присвоение уникального идентификатора заявке, сохранение информации о заявке и её состоянии.

### **2. Ведение учета заявок**

- Пользователь вводит информацию о заявках, включая ФИО ПОЧТУ

ТЕЛЕФОН ОПИСАНИЕ МОДЕЛЬ и статус (в ожидании, в работе, готово).

- Система автоматически добавляет их к базе данных.

### **2.3. Бизнес-процесс для автоматизации**

Для автоматизации бизнес-процесса обработки заявок на ремонт оборудования был выбран данный процесс, так как он является критически важным для эффективного управления техническим обслуживанием и должен быть автоматизирован по следующим причинам:

- 1. Скорость и точность обработки данных:** Автоматизация процесса обработки заявок на ремонт оборудования позволяет быстро и точно обрабатывать информацию о неисправностях. Система может моментально проверять наличие заявки, обновлять статус ремонта и автоматически сохранять изменения, что значительно снижает вероятность ошибок, связанных с ручным вводом данных.
- 2. Улучшение качества управления техническим обслуживанием:** Благодаря автоматизации пользователи получают мгновенный доступ к информации о своих заявках, включая их статусы, приоритеты и комментарии. Это создает более профессиональный подход к управлению процессом ремонта и улучшает общее впечатление от взаимодействия с сервисом.
- 3. Сокращение издержек:** Системы автоматизации помогают избежать затрат, связанных с физическим хранением документов и ведением бумажных записей. Они упрощают процесс управления заявками на ремонт, делая его более эффективным, что позволяет организациям избежать излишних расходов на организацию и поддержку процессов обслуживания.
- 4. Централизованное управление информацией:** Вся информация о



заявках, их статусах, сроках выполнения и ответственных лицах хранится в одном месте, что упрощает поиск и анализ данных. Это гарантирует безопасность и конфиденциальность информации о ремонте и предпочтениях пользователей.

**5. Гибкость и масштабируемость:** Современные автоматизированные системы легко адаптируются к изменениям в потребностях организации. При увеличении объема заявок или изменении процессов система может быстро реагировать на новые запросы, а также расширять функционал, добавляя новые категории и возможности без значительных вложений.

Автоматизация бизнес-процесса обработки заявок на ремонт оборудования не только упрощает управление техническим обслуживанием, но и способствует повышению эффективности работы команды, улучшению качества обслуживания и сокращению времени простоя оборудования.

### **3. Обзор инструментальных средств**

Обзор инструментальных средств для создания приложения по учету заявок на ремонт необходимо учитывать множество факторов, включая функциональность, производительность, удобство разработки, поддержку сообщества и совместимость с другими компонентами. Выбор языка программирования, среды разработки и базы данных должен основываться на потребностях пользователей, таких как простота использования, возможность интеграции с другими сервисами и масштабируемость решения. Для разработки данного приложения были выбраны следующие программы:

#### **3.1. Язык программирования: C#**

C# является одним из самых популярных языков программирования благодаря своей простоте, мощным возможностям объектно-ориентированного

программирования и поддержке современных технологий, таких как .NET Framework и .NET Core. Основные преимущества выбора C#:

- Производительность: C# обладает хорошей производительностью благодаря компиляции в промежуточный код (IL), который затем компилируется в машинный код. Это позволяет быстро выполнять задачи без значительных потерь производительности.
- Поддержка платформы .NET: Использование .NET предоставляет широкий спектр библиотек и фреймворков, упрощающих разработку сложных приложений.
- Объектно-ориентированное программирование: Легкость работы с классами, интерфейсами и полиморфизмом делает C# идеальным выбором для больших проектов.
- Сообщество поддержки: Большое количество документации, примеров кода и учебных материалов делают обучение и работу с языком легкой и доступной.

### **3.2. Фреймворк: WPF (Windows Presentation Foundation)**

WPF представляет собой платформу для разработки пользовательских интерфейсов Windows. Основные причины выбора этой технологии:

- Современный пользовательский интерфейс: Возможность создавать современные и привлекательные пользовательские интерфейсы с использованием XAML и различных элементов управления.
- Производительность: Поддержка графического ускорения и возможностей анимации, что особенно важно для приложений с интенсивным взаимодействием с пользователем.
- Интерактивность: Простая интеграция с различными технологиями, такими как сенсорные экраны и мультимедийные элементы.
- Масштабируемость: Хорошая поддержка для крупных и сложных

проектов, благодаря гибкости и расширяемости.

### **3.3. Инструментальное средство разработки: Visual Studio**

Visual Studio – это мощная интегрированная среда разработки от Microsoft, которая поддерживает разработку на многих языках, включая C#. Основные аргументы в пользу ее выбора:

- Расширенная функциональность: включает множество встроенных инструментов для написания, тестирования и отладки кода, проектирования архитектуры, работы с базами данных и многого другого.
- Интеграция с экосистемой .NET: Полная поддержка всех возможностей .NET, включая новые версии и библиотеки.
- Интеграция с системами контроля версий: Удобство работы с репозиториями и контроль версий проекта.
- Большое сообщество пользователей: Множество обучающих материалов, форумов и решений проблем от других разработчиков.

### **3.4. Система управления базами данных PostgreSQL**

PostgreSQL – это мощная реляционная система управления базами данных, часто выбираемая для веб-приложений и сервисов. Ее использование обосновано следующими причинами:

- Скорость и надежность: PostgreSQL известен своей высокой производительностью и надежностью, что особенно важно для систем, обрабатывающих большие объемы данных.
- Открытый исходный код: Открытый исходный код PostgreSQL позволяет свободно использовать и модифицировать базу данных под конкретные нужды.
- Распространение: PostgreSQL широко распространен и поддерживается множеством хостингов и провайдеров облачных сервисов, что облегчает

развертывание приложений.

- Масштабируемость: PostgreSQL хорошо подходит для проектов любого размера, от малых стартапов до крупных корпоративных решений, благодаря своей способности обрабатывать большие объемы данных и сложные запросы.
- API и интеграция: PostgreSQL предлагает богатый набор API и интеграционных возможностей с различными языками программирования и платформами, что упрощает разработку и интеграцию с другими системами.

## 4. Описание функционала

### 4.1. Окно входа

На данном экране пользователи могут войти в систему, используя свои учетные данные (логин и пароль). Если введенные данные совпадают с теми, что находятся в базе данных, открывается окно базы данных заявок.

Поля ввода: Логин и пароль.

Кнопка “Войти”: проверяет данные о пользователе что они существуют в базе данных.

Окно Авторизации изображено на рис.1.

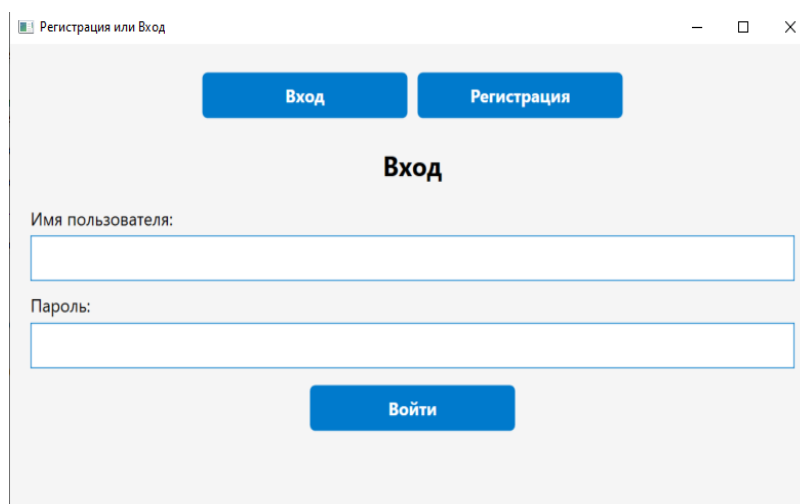


Рис.1.Окно входа

### 4.2. Окно регистрации

Здесь пользователи могут создать новый аккаунт. Для этого необходимо ввести логин, пароль и повторить пароль для подтверждения.

Окно регистрации и входа изображен на рис.4, рис.5,6.

Рисунок 4 Окно регистрации

Поля ввода: Логин, Пароль

Кнопка “Зарегистрироваться”: Проверяет, не существует ли уже пользователь с таким логином в базе данных. Если все данные корректны, создает новый аккаунт в базе.

Возврат на окно входа: Переход на окно входа в случае необходимости.

#### **4.3.Окно карточки заявки**

Это основное окно, в котором пользователи могут добавлять новые заявки в базу данных.

Окно базы данных заявок и код изображен на рис.7, рис.8.

Карточка заявки

ФИО:

Модель:

Телефон:

Статус заказа:

Почта:

Готово

Удалить

Описание:

Тип неисправности:

Рисунок 7 Окно карточки

```

1 using System.Text.RegularExpressions;
2 using System.Windows;
3 using System.Windows.Controls;
4 using System.Windows.Input;
5 using System.Linq;
6
7 namespace OrderManagementApplication
8 {
9     public partial class CardWindow : Window
10     {
11         public delegate void RequestSavedHandler(RepairRequest request);
12         public event RequestSavedHandler RequestSaved;
13
14         public delegate void RequestUpdatedHandler(RepairRequest request);
15         public event RequestUpdatedHandler RequestUpdated;
16
17         public delegate void RequestDeletedHandler(RepairRequest request);
18         public event RequestDeletedHandler RequestDeleted;
19
20         private RepairRequest _request;
21         private bool _isPhoneTextChanging; // флаг для отключения изменения текста
22
23         public CardWindow(RepairRequest request)
24         {
25             InitializeComponent();
26             _request = request;
27
28             if (_request != null)
29             {
30                 FullNameTextBox.Text = _request.FullName;
31                 PhoneTextBox.Text = _request.Phone;
32                 EmailTextBox.Text = _request.Email;
33                 DescriptionTextBox.Text = _request.Description;
34                 FaultTypeTextBox.Text = _request.FaultType;
35                 ModelTextBox.Text = _request.Model;
36
37                 // Установка состояния заказа
38                 if (!string.IsNullOrEmpty(_request.OrderStatus))
39                 {
40                     OrderStatusComboBox.SelectedItem = OrderStatusComboBox.Items
41                         .Cast<ComboBoxItem>()
42                         .FirstOrDefault(item => (string)item.Content == _request.OrderStatus);
43                 }
44             }
45         }
46
47         private void PhoneTextBox_PreviewTextInput(object sender, TextCompositionEventArgs e)
48         {
49             e.Handled = !IsTextAllowed(e.Text);
50         }
51
52         private void PhoneTextBox_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)
53         {
54             if (!_isPhoneTextChanging) return; // Если изменение текста уже происходит, выходим
55
56             var textBox = sender as System.Windows.Controls.TextBox;
57             string input = textBox.Text.Replace("<br>", "").Replace("<\\>", "").Replace("<\\>", "");
58
59             if (input.Length > 10)
60             {
61                 input = input.Substring(0, 10);
62             }
63
64             if (input.Length > 0)
65             {
66                 _isPhoneTextChanging = true; // Установка флага, чтобы предотвратить повторное изменение
67                 string formatted = string.Empty;
68
69                 if (input.Length >= 3)
70                 {
71                     formatted += "(" + input.Substring(0, 3) + ") ";
72                     if (input.Length >= 6)
73                     {
74                         formatted += input.Substring(3, 3) + "-";
75                         if (input.Length >= 10)
76                         {
77                             formatted += input.Substring(6, 4);
78                         }
79                         else
80                         {
81                             formatted += input.Substring(6);
82                         }
83                     }
84                     else
85                     {
86                         formatted += input.Substring(3);
87                     }
88                 }
89                 else
90                 {
91                     formatted = input; // Если меньше 3 символов, просто выводим их
92                 }
93
94                 textBox.Text = formatted;
95                 textBox.SelectionStart = textBox.Text.Length; // Установка курсора в конец
96                 _isPhoneTextChanging = false; // Сброс флага
97             }
98         }
99
100         private void EmailTextBox_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)
101         {
102             var textBox = sender as System.Windows.Controls.TextBox;
103             string email = textBox.Text;
104         }
105     }
106 }

```

Рисунок 8 Часть кода окна карточки

Поля ввода: ФИО, телефон описание модель, состояние, тип неисправности

Кнопка "Готово": Добавляет введенные данные в базу

## 4.4. Каталог

Можно смотреть редактировать и удалять заявки

Окно изображено на рис.9, рис.10.

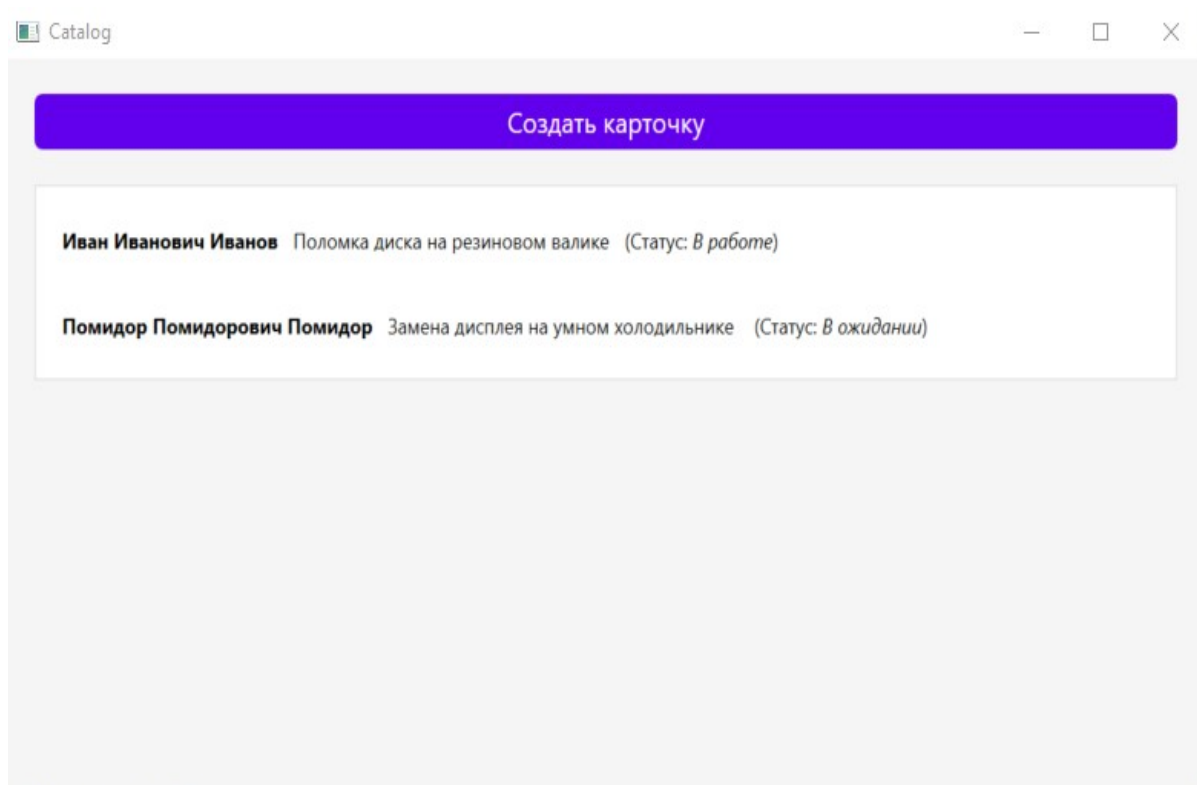


Рисунок 9 Каталог



```

1 using System.Collections.Generic;
2 using System.IO;
3 using System.Windows;
4 using Newtonsoft.Json;
5
6 namespace OrderManagementApplication
7 {
8     public partial class Catalog : Window
9     {
10         private Dictionary<string, List<RepairRequest>> userRequests = new Dictionary<string, List<RepairRequest>>();
11         private string currentUser; // имя текущего пользователя
12
13         public Catalog(string username)
14         {
15             InitializeComponent();
16             currentUser = username; // Загружаем имя текущего пользователя
17             LoadRequests(); // Загружаем запросы для текущего пользователя
18             UpdateRequestsList();
19         }
20
21         private void CreateCardButton_Click(object sender, RoutedEventArgs e)
22         {
23             var cardWindow = new CardWindow(null);
24             cardWindow.RequestSaved += CardWindow_RequestSaved;
25             cardWindow.ShowDialog();
26         }
27
28         private void CardWindow_RequestSaved(RepairRequest request)
29         {
30             if (!userRequests.ContainsKey(currentUser))
31             {
32                 userRequests[currentUser] = new List<RepairRequest>();
33             }
34             userRequests[currentUser].Add(request);
35             UpdateRequestsList();
36             SaveRequests(); // Сохраняем запросы после добавления
37         }
38
39         private void RequestsListBox_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)
40         {
41             if (RequestsListBox.SelectedItem is RepairRequest selectedRequest)
42             {
43                 var cardWindow = new CardWindow(selectedRequest);
44                 cardWindow.RequestUpdated += CardWindow_RequestUpdated;
45                 cardWindow.RequestDeleted += CardWindow_RequestDeleted; // Обработка на удаление запроса
46                 cardWindow.ShowDialog();
47             }
48         }
49
50         private void CardWindow_RequestUpdated(RepairRequest updatedRequest)
51         {
52             if (userRequests.ContainsKey(currentUser))
53             {
54                 var index = userRequests[currentUser].IndexOf(updatedRequest);
55                 if (index >= 0)
56                 {
57                     userRequests[currentUser][index] = updatedRequest;
58                     UpdateRequestsList();
59                     SaveRequests(); // Сохраняем запросы после обновления
60                 }
61             }
62         }
63
64         private void CardWindow_RequestDeleted(RepairRequest request)
65         {
66             if (userRequests.ContainsKey(currentUser))
67             {
68                 userRequests[currentUser].Remove(request);
69                 UpdateRequestsList();
70                 SaveRequests(); // Сохраняем запросы после удаления
71             }
72         }
73
74         private void UpdateRequestsList()
75         {
76             RequestsListBox.Items.Clear();
77             if (userRequests.ContainsKey(currentUser))
78             {
79                 foreach (var request in userRequests[currentUser])
80                 {
81                     RequestsListBox.Items.Add(request);
82                 }
83             }
84         }
85
86         private void SaveRequests()
87         {
88             var json = JsonConvert.SerializeObject(userRequests, Formatting.Indented);
89             File.WriteAllText("repairRequests.json", json);
90         }
91
92         private void LoadRequests()
93         {
94             if (File.Exists("repairRequests.json"))
95             {
96                 var json = File.ReadAllText("repairRequests.json");
97                 userRequests = JsonConvert.DeserializeObject<Dictionary<string, List<RepairRequest>>>(json) ?? new Dictionary<string, List<RepairRequest>>();
98             }
99         }
100     }
101 }

```

Рисунок 10 Код окна каталога

## **ЗАКЛЮЧЕНИЕ**

За время прохождения учебной практики выполнены все задания, предусмотренные программой.

В процессе работы выполнены следующие задачи:

Была изучена предметная область. В результате было получено глубокое понимание предметной области и её основных концепций.

Техническое задание было проанализировано, и на его основе была составлена краткая спецификация разрабатываемого модуля. Были выделены входные и выходные данные, что позволило определить требования к модулю и его функциональность.

**В результате цель учебной практики по разработке программного модуля для учёта заявок на ремонт оборудования достигнута.**

## Приложения

```

1  using System.Collections.Generic;
2  using System.IO;
3  using System.Linq;
4  using System.Windows;
5  using Newtonsoft.Json;
6
7  namespace OrderManagementApplication
8  {
9      Создать 4
10     public partial class registr_or_login : Window
11     {
12         private const string usersFilePath = "users.json"; // Путь к файлу для хранения пользователей
13         private List<User> users = new List<User>();
14
15         Создать 1
16         public registr_or_login()
17         {
18             InitializeComponent();
19             LoadUsers(); // Загружаем пользователей при инициализации
20         }
21
22         Создать 1
23         private void ShowLoginPanel(object sender, RoutedEventArgs e)
24         {
25             LoginPanel.Visibility = Visibility.Visible;
26             RegisterPanel.Visibility = Visibility.Collapsed;
27         }
28
29         Создать 1
30         private void ShowRegisterPanel(object sender, RoutedEventArgs e)
31         {
32             LoginPanel.Visibility = Visibility.Collapsed;
33             RegisterPanel.Visibility = Visibility.Visible;
34         }
35
36         Создать 1
37         private void LoginButton_Click(object sender, RoutedEventArgs e)
38         {
39             string username = LoginUsernameTextBox.Text;
40             string password = LoginPasswordBox.Password;
41
42             // Проверим, существует ли пользователь с таким именем и паролем
43             var user = users.FirstOrDefault(u => u.Username == username && u.Password == password);
44             if (user != null)
45             {
46                 MessageBox.Show($"{username}\nИмя пользователя: {username}", "Информация о входе");
47                 Catalog cataloge = new Catalog(username); // Передаем имя пользователя
48                 cataloge.Show();
49                 this.Close();
50             }
51             else
52             {
53                 MessageBox.Show("Имя пользователя или пароль.", "Ошибка входа");
54             }
55         }
56
57         Создать 1
58         private void RegisterButton_Click(object sender, RoutedEventArgs e)
59         {
60             string username = RegisterUsernameTextBox.Text;
61             string password = RegisterPasswordBox.Password;
62
63             // Проверим, существует ли уже пользователь с таким именем
64             if (users.Any(u => u.Username == username))
65             {
66                 MessageBox.Show("Пользователь с таким именем уже существует.", "Ошибка регистрации");
67                 return;
68             }
69
70             // Создаем нового пользователя и добавляем его в список
71             var newUser = new User { Username = username, Password = password };
72             users.Add(newUser);
73             SaveUsers(); // Сохраняем пользователей после регистрации
74             MessageBox.Show($"Регистрация:\nИмя пользователя: {username}", "Информация о регистрации");
75         }
76
77         Создать 1
78         private void SaveUsers()
79         {
80             var json = JsonConvert.SerializeObject(users, Formatting.Indented);
81             File.WriteAllText(usersFilePath, json);
82         }
83
84         Создать 1
85         private void LoadUsers()
86         {
87             if (File.Exists(usersFilePath))
88             {
89                 try
90                 {
91                     var json = File.ReadAllText(usersFilePath);
92                     users = JsonConvert.DeserializeObject<List<User>>(json) ?? new List<User>();
93                 }
94                 catch (JsonException ex)
95                 {
96                     MessageBox.Show($"Ошибка при загрузке пользователей: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
97                 }
98                 catch (IOException ex)
99                 {
100                     MessageBox.Show($"Ошибка при чтении файла: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
101                 }
102             }
103         }
104     }
105
106     Создать 5
107     public class User
108     {
109         Создать 3
110         public string Username { get; set; }
111         Создать 2
112         public string Password { get; set; }
113     }
114 }

```

Рисунок 2: Код окна авторизации

### Список использованных источников

1. Гагарина, Л. Г. Разработка и эксплуатация автоматизированных информационных систем: учебное пособие / Л. Г. Гагарина. — Москва: ФОРУМ: ИНФРА-М, 2021. — 384 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0735-1. — Текст: электронный. - URL: <https://znanium.com/catalog/product/1214882>
2. Гагарина, Л. Г. Технология разработки программного обеспечения: учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул; под ред. Л.Г. Гагариной. — Москва: ФОРУМ: ИНФРА-М, 2021. — 400 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0812-9. - Текст: электронный. - URL: <https://znanium.com/catalog/product/1189951>
3. Гниденко, И. Г. Технология разработки программного обеспечения: учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва: Издательство Юрайт, 2021. — 235 с. — (Профессиональное образование). — ISBN 978-5-534-05047-9. — Текст: электронный // Образовательная платформа Юрайт
4. Федорова Г.Н. Осуществление интеграции программных модулей, учебник для студентов учреждений среднего профессионального образования, 4-е изд., перераб. Издательство: Академия. , 2021. — 272 с - URL: <https://www.academia-library.ru/catalogue/4891/345766/>
5. Лаврищева Е. М. Программная инженерия и технологии программирования сложных систем [Электронный ресурс]:учебник для вузов. - Москва: Юрайт, 2022. - 432 с – Режим доступа: <https://urait.ru/bcode/491029>
6. Рудаков А. В. Технология разработки программных продуктов. Учебное пособие Издательство: Академия <https://academia-library.ru/reader/?id=401005&demo=Y>