МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных Технологий

Кафедра Корпоративных Информационных Систем

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине Разработка программный приложений

Тема курсового проекта (работы) Разработка информационной системы «Сервисный центр».

Студент группы ИКБО-08-18 Корчиков Михаил Дмитриевич _____

(учебная группа, фамилия, имя, отчество студента) *(подпись студента)*

Руководитель курсового проекта (работы) ст. преп. Мирзоян Д.И._____

(подпись руководителя)

Работа представлена к защите «____»_____20___г.

Допущен к защите «____»_____20___г.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных Технологий

Кафедра Корпоративных Информационных Систем

Утверждаю

Заведующий кафедрой КИС

_____ Андрианова Е.Г.

«____»_____20___г.

ЗАДАНИЕ

на выполнение курсового проекта (работы) по дисциплине

«Разработка программных приложений»

Студент Корчиков Михаил Дмитриевич Группа ИКБО-08-18

Тема Разработка информационной системы «Сервисный центр».

Исходные данные: описание языка и среды разработки на языке C#, стандарт оформления программного кода на языке C#

Перечень вопросов, подлежащих разработке, и обязательного графического материала:

1. ER модель предметной области
2. Состав, атрибуты классов и их типы
3. Формат долговременного хранения данных в БД
4. Пользовательский интерфейс приложения
5. Набор автоматизируемых функций

Срок представления к защите курсового проекта (работы): до «___» _____ 201__ г.

Задание на выполнение курсовой проект (работу) выдал _____ **(** Мирзоян Д.И. **)**

Подпись руководителя *Ф.И.О. руководителя*

Задание на курсовой проект (работу) получил _____ **(** Корчиков М.Д. **)**

Подпись обучающегося *Ф.И.О. исполнителя*

«___»_____20___г.

Техническое задание:

1. Наименование работы

Разработка информационной системы «Сервисный центр».

2. Исполнитель

Корчиков Михаил Дмитриевич Шифр: 18И0997

Основание для разработки:

Учебный план бакалавриата по направлению 09.03.04 «Программная инженерия».

3. Цель и назначение работы

Данный продукт разработан в качестве примера для студентов, изучающих объектно-ориентированное программирование. Так же предполагается его использование для обеспечения работы сервисного центра.

4. Технические требования

Рассмотреть процессы приема в ремонт, выдачи и экспертизы. Автоматизировать отчеты о среднем времени ремонта определенных моделей, средней стоимости ремонта определенных моделей, основным причинам ремонта определенных групп моделей.

5. Содержание работы

В результате выполнения работы решаются следующие задачи:

1. Составление теоретического материала.
2. Разработка структуры системы
3. Оформление Расчетно-пояснительной записки
4. Программная реализация системы
5. Защита работы и демонстрация работы системы

6. Порядок сдачи работы

Выполнения работы (РПЗ и ИС в соответствии со стандартами кафедры КИС)

Руководитель **Исполнитель**

(подпись, дата) (подпись, дата)

Аннотация

В данной курсовой работе рассмотрено создание информационной системы «Сервисный центр», приложение типа Desctop application. Рассмотрены процессы: анализа предметной области, проектированя приложения, реализации приложения сервисного центра и тестирования приложения с оценкой стоимости.

Была написана инструкция для пользователя, которая содержит набор действий пользователя для осуществления типовых сценариев работы с системой «сервисный центр».

Графические материалы

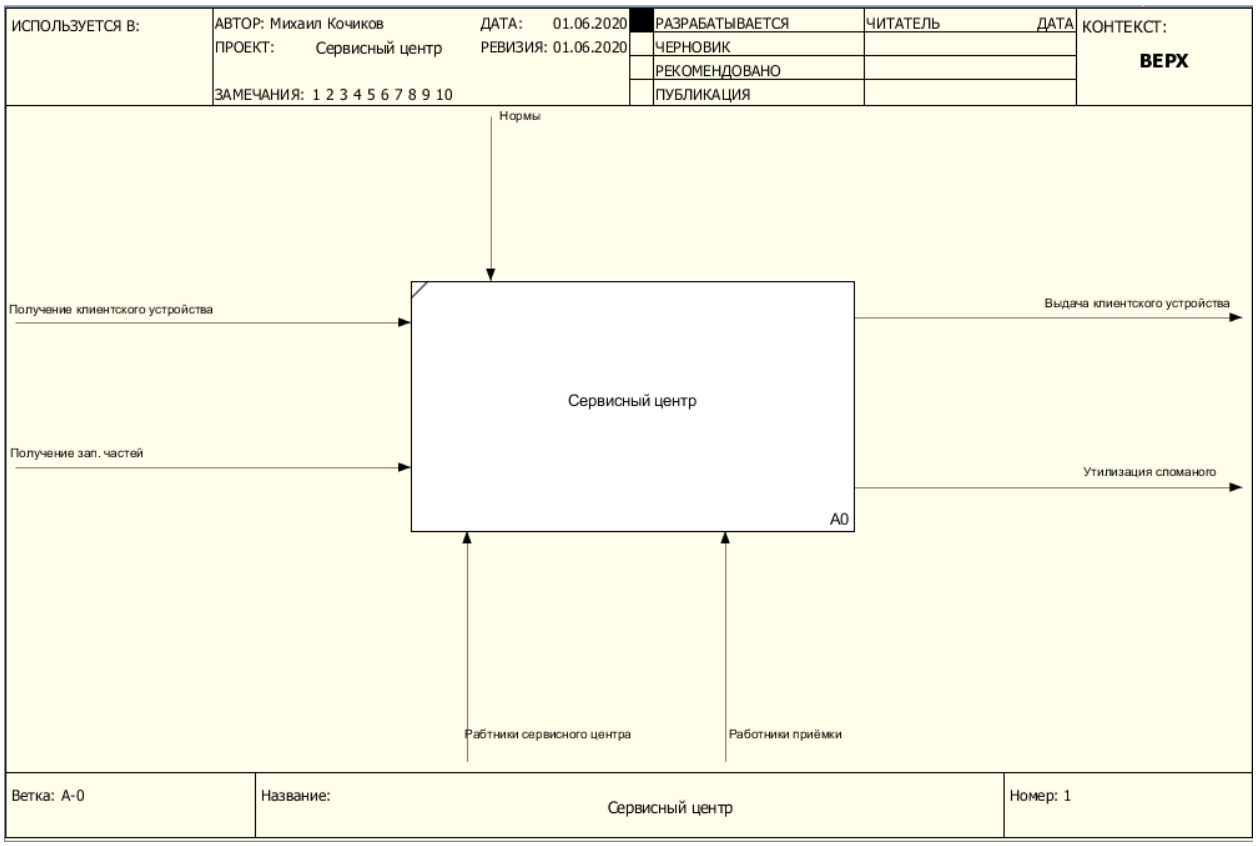

Рисунок. Диаграмма процесса.

Оглавление

Введение

В данной курсовой работе представлена реализация информационной системы «Сервисный центр». В неё входят система авторизации разных типов пользователей. Системы получения и выдачи устройства заказчика. Система оценки повреждений определённого устройства с установкой стоимости ремонта и установкой определённых повреждений. Поиска определённого устройства в базе данных всех устройств, находящихся в сервисном центре, для определения статуса выполнения работы. Система составлений отчётов по номеру модели выводящая статистику среднего времени ремонта, средней стоимости ремонта и основрую причину ремонта. Система добавления и удаления пользователей с разными правадми, которым отключены определённые функции в зависимости от типа пользователя. Система вывода содержания таблиц, которые могут понадобится для поиска или проверки информации.

Программная часть реализована при помощи обьектно-ориентированного языка C# разработанного компанией Microsoft, а также интерфейса программирования приложений Windows Forms, отвечающего за пользовательский графический интерфейс.

Ведение действий сервисного центра – сложный процесс, требующий внимания и повышенной ответственности. Программное приложение «Сервисный центр» даёт возможность удобного учёта устройств, принятых на обслуживание. Оно исключает проблемы с потерей или недостаточной информацией на каком из этапов, таких как экспертиза, починка или других, находиться устройство.

Данное приложение актуально сейчас, как никогда – в наше время идет активное развитие информационных технологий в компаниях, занимающихся ведением сервисной деятельности, поэтому появление подобного ПО может заменить ручной учет составляющих этого бизнеса что ползволит ускорить деятельность занимавшую значительное время при ручном труде.

1. Анализ предметной области

1.1 Исследование предметной области

При исследовании предметной области встал вопрос о том, как можно максимально упростить для пользователей использование разрабатываемого приложения.

Отталкиваясь от темы курсовой работы, было принято решение реализовать информационную систему как «Desktop application».

При анализе предметной области были выявлены основные подразделения охвата компании и их критерии:

1. Устройства в рабте

Устройства, находящиеся в работе, подразумевает под собой все устройства, находящиеся у сервисного цента на експертизе, ожидании выдачи, ожидании оплаты и во время работы над устранением неполадок.

Характеристика устройства, принятого сервисным центром подразумевает:

- Модельный номер;
- Стадия, на котором находится устройство;
- Время приёма или время начала восстановления устройства.
- ФИО клиента/владельца устройства;
- Стоимость работ, назначенная после экспертизы;
- Перечисление причин, по которым будет проводиться ремонт данного устройства.

2. Выданые устройства

Устройства, над которыми уже была проведена работа, подразумевает под собой все устройства, прошедшие ремонт в сервисном центре выданые кликнтам.

Характеристики устройства, выданые клиентам:

- Модельный номер;
- Время начала работы по устранению неполадки;
- Время окончаний работы по устранию неполадки;

- Стоимость, назначенная за устранение неполадки;
- ФИО кликнта.

3. Пичины поломок

Причины поломок каждого определённого модельного номера устройства, прошедшего ремонт в сервисном центре.

Характеристики:

- Модельный номер;
- Причина поломки.

4. Таблица данных причин поломок

Все причины, по которым может произойти поломка устройства. Список причин может пополняться пользователем-администратором.

Характеристики, хранящиеся в базе данных:

- Возможные причины выхода из строя.

1.2 Обзор существующих в предметной области систем

Так как информационная система «Сервисный центр» напрямую относиться к CRM (Customer Relationship Management) системам, следут так же учитывать уже существующие стандартные системы и технологии, задействованные в целевом направлении нашей информационной системы.

1.2.1 LiveSklad

LiveSklad являеться CRM системой не имеющей бесплатной версии. Облачная платформа для автоматизации сервисного центра, помогающая взаимодействовать с клиентом и контролировать все процессы в сервисе. Подходит для любых размеров бизнеса, ориентирована на сервисные центры по ремонту техники, автосервисы, ателье и другие сферы деятельности. Включает в себя не только ведение деятельности сервиса, но также имеет в себе инструменты для удобной арганизации менеджмента складского товара.

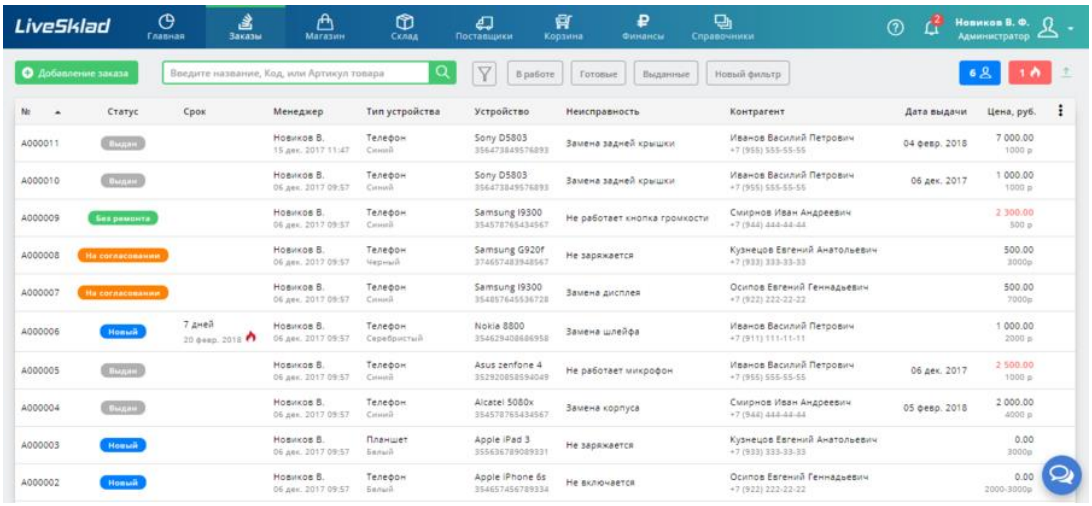

Рис. 1. LiveSklad.

1.2.2 HubEx

HubEx является CRM системой, не имеющей бесплатной версии. Облачная платформа для автоматизации сервисного центра, помогающая взаимодействовать с клиентом и контролировать все процессы в сервисе. Так же данное решение является кроссплатформенным.

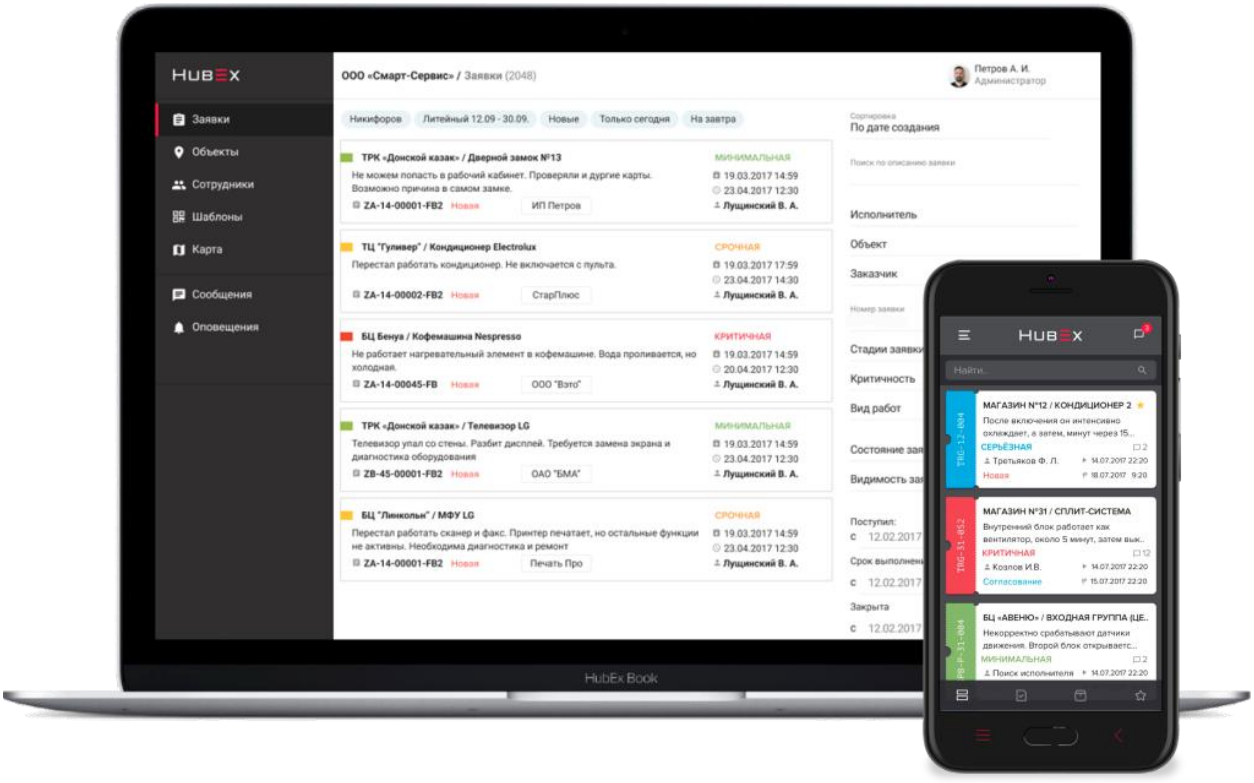

Рис. 2. HubEx.

1.2.3 WorkPan

Программа WorkPan представляет собой систему автоматизации бизнес-процессов, нацеленную на решение ряда организационных задач. Основная цель проекта – обеспечение программного обеспечения, призванного облегчить работу и оказать помощь в ведении коммерческой деятельности. Имеет инструменты для организации логистики, менеджмента кассы.

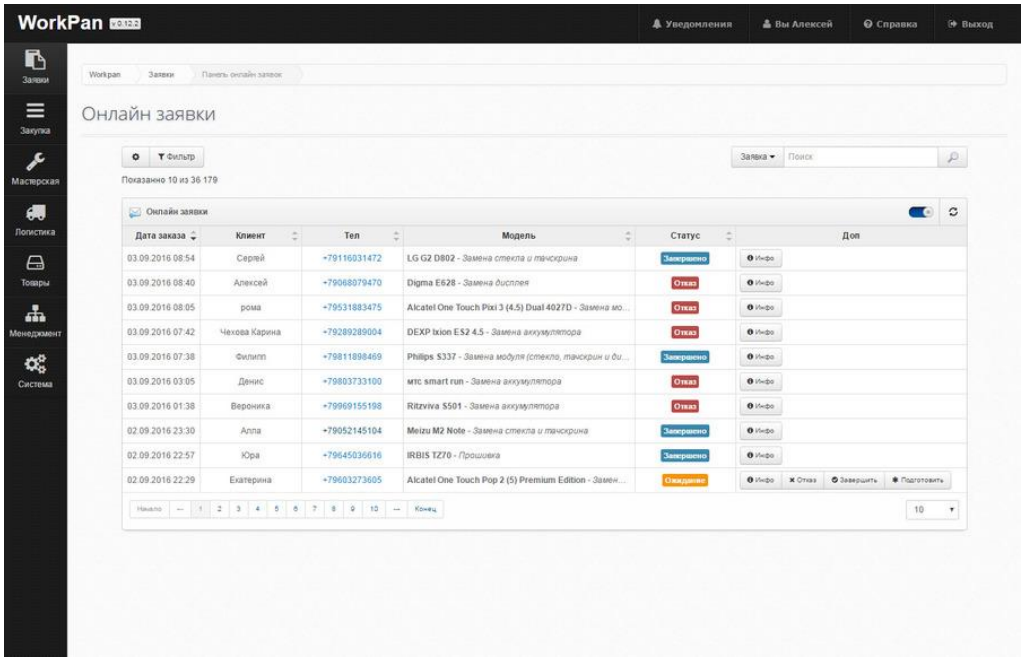

Рис. 3. WorkPan.

Оценив реализацию данных проектов и сравнив их возможности для себя выбрали основной функционал, который будет реализован и в нашей информационной системе.

1.3 Обзор существующих в предметной области технологий

1.3.1 SQL

Для хранения информации о всех устройствах, поступивших в сервисный центр будет исспльзован SQL. SQL —декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. При всех своих

изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов. Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую. Существуют системы, разработчики которых изначально ориентировались на применение по меньшей мере нескольких СУБД. Естественно, что при применении некоторых специфичных для реализации возможностей такой переносимости добиться уже очень трудно. С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса. Однако не стоит думать, что это полностью универсальный принцип — программист описывает набор данных для выборки или модификации, однако ему при этом полезно представлять, как СУБД будет разбирать текст его запроса. Чем сложнее сконструирован запрос, тем больше он допускает вариантов написания, различных по скорости выполнения, но одинаковых по итоговому набору данных.

1.3.2 SQLite

Слово «встраиваемый» (embedded) означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт

того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени.

1.3.3 Windows Forms

Windows Forms — это интеллектуальная клиентская технология для .NET Framework, набор управляемых библиотек, упрощающих типичные задачи приложения, такие как чтение и запись в файловую систему. При использовании среды разработки, такой как Visual Studio, можно создавать Windows Forms интеллектуальные клиентские приложения, которые отображают сведения, запрашивают ввод данных от пользователей и обмениваются данными с удаленными компьютерами по сети.

В Windows Forms форма — это визуальная поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем помещения элементов управления на форму и написания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления — это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события с помощью кода и обрабатывает события при их

возникновении. Подробнее см. в разделе Создание обработчиков событий в Windows Forms.

Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. Список всех элементов управления, которые можно использовать в форме, представлены в разделе Элементы управления для использования в формах Windows Forms. Если существующий элемент управления не удовлетворяет потребностям, в Windows Forms можно создать пользовательские элементы управления с помощью класса UserControl.

2. Проектирование приложения

2.1. Разработка пользовательского интерфейса

Проектная часть курсового проекта описывает задачи, связанные с проектированием. К таким задачам относятся:

- формирование общих требований;
- определение функциональной части проекта;
- проектирование графического интерфейса пользователя.

2.2. Проектное решение

Система «Сервисный центр» разработана как приложение для компьютеров, с удобным пользовательским интерфейсом. Данные хранятся в файлах с расширением .sqllite.

Основным языком реализации является C# (по заданию на курсовой проект). Также при разработке использовались такие инструментальные средства:

- Windows Forms;
- SQLite;
- Entity Framework.

2.2.1. Разработка спецификации

В качестве основной спецификацией будет являться являться функционал, который данное приложени должно выполнять. Данный функционал является основополагающим: он задает основыне свойства системы и в нем заключены ее возможности. К таким функциональным требованиям относятся:

- Вывод всех таблиц базы данных;
- Приём и выдача устройства;
- Ремонт устройства;
- Оценка с выставлением стоимости и описанием неисправностей;
- Создание пользователей с разными привлегиями;
- Вывод среднего времени ремонта по модели;
- Вывод средней стоимости по моедли;
- Вывод основной причины ремонта;
- Пополнение списка возможных повреждений;
- Изменения информации в базе данных.

2.3. Реализация программного продукта

2.3.1. Реализация возможности хранения

Все данные, хранящиеся в программе, так же сохраняются в базу данных, реализованную на основе SQLlite.

Предусмотрена одна база данных хранящая в себе пять разных таблиц: которые реализуют хранение информации о устройствах, находящихся в работе, выданых устройствах, все возможные причины поломок, все бывшие в ремонте модели и их выявленные повреждения, так же в одной из таблиц хранится информация о пользователях приложения.

При открытии приложения из базы данных загружается вся информация о устройствах, находящихся в сервисном центре.

При каких-либо дейсвия связанных с изменением базы данных, данные в базе изменяются.

Для обеспечения хранения и общения с базой данных SQLite была исспользована библиотека System.Data.SQLite которая использующая ADO.NET и Entity Framework для работы и упрощения базовых операций:

- Выполнение запросов;
- Подключение к базе данных;
- Выбор таблицы;
- Изменения базы данных.

Приведу пример как выводится таблица в элемент Windows Forms под названием dataGridView, для предаставления выбора определённого устройства, находящегося в сервисном центре или для получения информации о всех устройсвах выданых клиентам или находящихся на обслуживании в сервисном центе. Так же может выводить остальные таблицы базы данных, которые не доступны пользователям с привилегиями гостя и обычного пользователя. Таблицы доступные для просмотра только администратору содержат:

- Прошедшие обслуживание модели и их неисправности;
- Причины возможных поломок исспользуемые для оценки повреждений;
- Пользовательские данные под которыми заходят разные пользователи данного приложения.

Листинг 1. Отрисовка таблицы.

```
private void DrawTable(int tableIndex, string name, string model)
{
            lastTablePrint = (byte)tableIndex;
            dataGridView1.Columns.Clear();
            dataGridView1.Refresh();

            string tableName = Enum.GetName(typeof(tables),
tableIndex);
            SQLiteCommand command = new SQLiteCommand($"SELECT * FROM
{tableName} WHERE FIO = \"{name}\" AND modelNum = \"{model}\"",
dbConnection);
            SQLiteDataReader reader = null;
            reader = command.ExecuteReader();

            for (int i = 0; i < columns[tableIndex].Length; ++i)
            {
```

```
            dataGridView1.Columns.Add("", columns[tableIndex][i]);
        }

        while (reader.Read())
        {
            string[] fileds = new string[reader.FieldCount];
            for (int i = 0; i < reader.FieldCount; ++i)
            {
                fileds[i] = reader[i].ToString();
            }
            dataGridView1.Rows.Add(fileds);
        }

        reader.Close();
    }
```

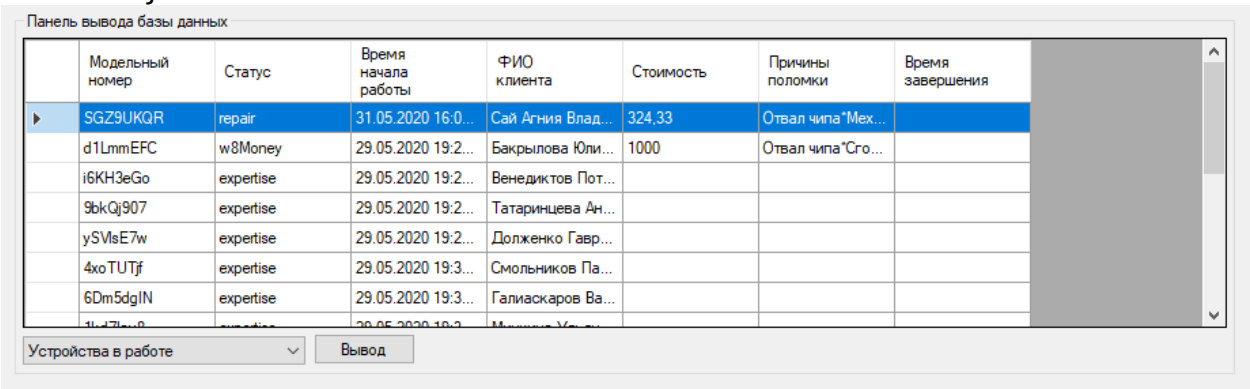

Рис. 4. Пример вывода таблицы базы данных в пользовательский интерфейс.

2.3.2. Реализация программного изделия

Во время разработки следует разбить програмное обеспечение на несколько областей по смыслу, для удобства пользования.

Пользовательский ввод должне проверяться на основе допустимых значений и выводить ошибку если что-то пошло не так.

Для того чтобы точнее проинформировать пользователя о неккоректном вводе, следует разбить одну большую проверку, на несколько мнеьших, что позволит увеличить разннообразие отображаемых сообщений с предупреждением об ошибке ввода или другой ошибке.

2.4. Описание структуры интерфейса пользователя

Пользовательский графический интерфейс был спроектирован с условием удобства и интуетивности при исспользовании. По причине широково функуионала приложени и большого колическа элементов, мной было принято решение разделить функционал приложения на основные

группы, иссползолуя элемент «Group box». Так же что бы не переполнять информацей одно окно, было сделано несколько окон приложения, относящимся к разному функционалу. Разделения функционала в разные формы (окна) не принисло усложнения интерфейса и усложнения польвательского взаимодействия.

3. Реализация приложения

3.1. Вспомогательные библиотеки

3.1.1. System.Data.SQLite

В некоторых проектах достаточно часто возникает необходимость в хранении данных, объем которых уже нельзя назвать маленьким, но в тоже время использовать какую-либо СУБД слишком накладно из-за сложности развертывания приложения. И тут на помощь приходит такая прекрасная вещь как SQLite – компактная встраиваемая база данных с которой работает и наше приложение. System.Data.SQLite — ADO.NET провайдера для работы с SQLite. Данный провайдер так же позволяет задействовать все дополнительные возможности последних версий .NET, такие как LINQ, Entity Framework. Первое, с чего стоит начать – это выбор платформы, на которую нацелено создаваемое приложение. Надо отдать должное разработчикам провайдера для SQLite, которые обеспечили поддержку почти всех доступных платформ. Все, что необходимо сделать – выбрать нужную версию сборки System.Data.SQLite.dll. Сама сборка скомпилирована в различных вариациях, что удобно для разворачивания приложения не только на одной конкретной платфоме. Пример применения библиотеки для получения данных из таблицы, в данном случае получение времени начала и конца работы:

Листинг 2. Средее время работы.

```
private void AvarageTimeBtn_Click(object sender, EventArgs e)
            {
                List<double> times = new List<double>();
                byte tableIndex = 1;
                lastTablePrint = (byte)tableIndex;

                string tableName = Enum.GetName(typeof(tables),
tableIndex);
                SQLiteCommand command = new
SQLiteCommand($"SELECT StartTime, EndTime FROM {tableName} WHERE
modelNum ='{ModelNumTb.Text}'", dbConnection);
                SQLiteDataReader reader =
command.ExecuteReader();

                while (reader.Read())
                {
                    string[] fileds = new
string[reader.FieldCount];
                    for (int i = 0; i < reader.FieldCount; ++i)
                    {
                        fileds[i] = reader[i].ToString();
                    }
                    TimeSpan deltaTime =
DateTime.Parse(fileds[1]) - DateTime.Parse(fileds[0]); //Считает
разницу.
                    times.Add(deltaTime.TotalHours);
                }

                reader.Close();

                if (times.Count != 0)
                {
                    double res = 0;
                    for (int i = 0; i < times.Count; ++i)
                    {
```

```
                    res += times[i];
                }
                res /= times.Count;
                MessageBox.Show($"Среднее время ремонта
устройств с модельным номером {ModelNumTb.Text} составляет {res}
часов", "Результат вашего запроса.", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
            else
            {
                MessageBox.Show("Не найдено информации о
устройстве.", "Результат вашего запроса.", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
        }
```

3.1.2. Windows Forms

Так как реализация программной части информационной системы «Сервисный центр» на языке обьектно-ориентированного программирования C# разработаного и поддерживаемого компанией Microsoft для реализации пользовательского интерфейса приложения было была выбрана технология Windows Forms, удобная для быстрого создания различный пользловательских графический интерфейсов, которые будут не только удобны пользователю, но и программисту, который будет реализовывать функционал приложения и каждой отедльной функции которая вызываеться пользователем при каких-либо его действиях. Windows Forms позволяет разрабатывать приложение с полнофункциональным графическим интерфейсом, простое в развертывании и обновлении, способное работать при наличии или отсутствии подключения к Интернету и использующее более безопасный доступ к ресурсам на локальном компьютере по сравнению с традиционными приложениями Windows. Windows Forms имет широкий функционал, при создании приложения этот инструмент достаточно гибок что бы его исспользовать в приложениях различного обьёма и масштаба. Так

же его удобство в обширном списке обрабатываемых событий для каждого возможного элемента. Форма в Windows Forms — визуальная поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем помещения элементов управления на форму и написания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления — это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных. При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события с помощью кода и обрабатывает события при их возникновении. Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. Список всех элементов управления, которые можно использовать в форме, представлены в разделе Элементы управления для использования в формах Windows Forms. Если существующий элемент управления не удовлетворяет потребностям, в Windows Forms можно создать пользовательские элементы управления с помощью класса UserControl. Вы можете легко создавать элементы управления с привязкой к данным с помощью *параметров*. Большинство интеллектуальных клиентских приложений должны сохранять некоторые сведения о своем состоянии во время выполнения, такие как последние известные размеры форм, а также сохранять пользовательские предпочтения, например, место сохранения файлов по умолчанию. Параметры приложения отвечает этим требованиям, предоставляя простой способ хранения обоих типов сведений на клиентском компьютере. После определения этих параметров с помощью Visual Studio или редактора кода параметры сохраняются в формате XML и автоматически считываются обратно в память во время выполнения.

У Windows Forms есть аналог — Windows Presentation Foundation (WPF) предоставляет библиотеку общих двухмерных фигур, нарисованных с помощью векторов, таких, как прямоугольники и эллипсы, а также

графические пути. И в своей функциональности фигуры реализуют многие возможности, которые доступны обычным элементам управления.

Двухмерная графика в WPF включает визуальные эффекты, такие как градиенты, точечные рисунки, чертежи, рисунки с видео, поворот, масштабирование и наклон. WPF также включает возможности трехмерной отрисовки, интегрированные с двухмерной графикой, что позволяет создавать более яркий и интересный пользовательский интерфейс. Так же к особенностям WPF можно отнести исспользование языка разметки XAML который. Всё это делает WPF более гибким инструментом в отношении визуального интерфейса. Но так как нам не требуется такая гибкая система реализации интерфейса и не требуется отрисовывать трёхмерные обьекты, мы остановились на технологии Windows Forms.

4. Тестирование приложения

4.1. Цель испытаний

Целью проведения испытаний является:

- Проверка работоспособности функция продукта;
- Проверка соответствия заявленным характеристикам и требованиям из технического задания;
- Проверка готовности программы к проведению приёмочных испытаний;
- Определение уязвимостей проекта и дальнейшее их устранение.

4.2. Тестирование

Вводимые данные	Ожидаемый результат	Выведеный результат
Моделный в в поиск по модели	Правильно выведенная таблица только с данным модельным номером	Таблица только с данным модельным номером
Создание	Отказ от создания по	Отказ от создания по

пользователя с уже существующим логином.	причине того, что данный логин уже занят другим пользователем.	причине того, что данный логин уже занят другим пользователем.
Не существующий в базе данных модельный номер в поиске по номеру.	Пустая таблица.	Пустая таблица.
Пустой модельный номер в поиске по модельному номеру.	Отказ от выполнения операции.	Отказ от выполнения операции.
Пустое поле ФИО при поиске по ФИО.	Отказ от выполнения операции.	Отказ от выполнения операции.
Пустое поле модели при поиске по номеру модели и ФИО	Отказ от выполнения операции.	Отказ от выполнения операции.
Пустое поле ФИО при поиске по номеру модели и ФИО	Отказ от выполнения операции.	Отказ от выполнения операции.
Попытка выбора не правильной Базы данных	Вывод ошибки.	Вывод ошибки.
Выставление оценки при выборе модели не из таблицы с устройствами неходящимися в работе сервисного центра.	Отказ от выполнения операции.	Отказ от выполнения операции.
Выставление оценки	Отказ от выполнения	Отказ от выполнения

без введённой суммы для оплаты.	операции.	операции.
Ввод невозможного для преобразования в число текста.	Вывод ошибки. Отказ от выполнения операции.	Вывод ошибки. Отказ от выполнения операции.
Попытка обновления не изменнённого списка поверждений.	Вывод списка.	Вывод списка.
Попытка обновления списка повреждений, после добавления нового повреждения.	Вывод обновлённого списка.	Вывод обновлённого списка.
Перевод неоценённого устройства в ожидание оплаты.	Отказ от выполнения операции.	Отказ от выполнения операции.
Попытка отсановки починки устройства не находящегося в стадии починки.	Отказ от выполнения операции.	Отказ от выполнения операции.
Добавление новой причины поломки.	Появление новой причины поломки.	Появления новой причины поломки.
Добавления новой причины поломки с пустым полем причины.	Отказ от выполнения операции.	Отказ от выполнения операции.
Вхдод в приложение с непривильными данными.	Вывод ошибки информационного окна о том, что данный пользователь не найден.	Вывод ошибки информационного окна о том, что данный пользователь не найден.

Вход в приложения с правильными данными.	Открытие приложения.	Открытия приложения.
Вход в приложения с правами пользователя.	Ограничение доступа к функциям администатора.	Ограничение доступа к функциям администатора.
Вход в приложение с правами гостя.	Ограничение доступа ко всем функциям воздействующим с базой данных, кроме вывода устройств в работе и выданых клиенту.	Ограничение доступа ко всем функциям воздействующим с базой данных, кроме вывода устройств в работе и выданых клиенту.

Тестовые запуски показали, что приложение:

- Работоспособно;
- Устойчиво к ошибкам пользователя;
- Выполнено в соответствие с требованиями.

5. Инструкция

Для начала работы нудно открыть приложение и выбрать базу данных, которая может находится там же где и приложения или в другом каталоге или даже на Samba сервере.

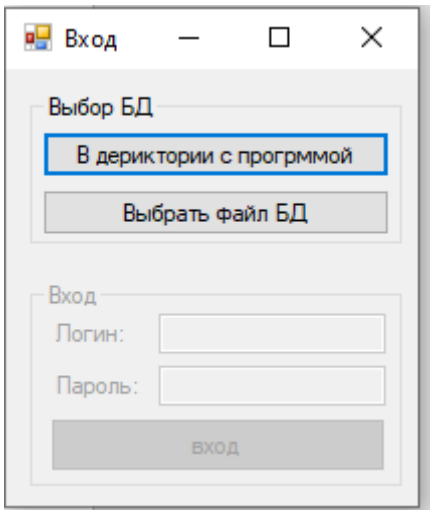

Рис. 5. Окно выбора расположения базы данных.

После выбора базы данных, если вы всё сделали правильно вам будет доступно окно входа, где будет требоваться ввести данные о пользователе для входа в приложение.

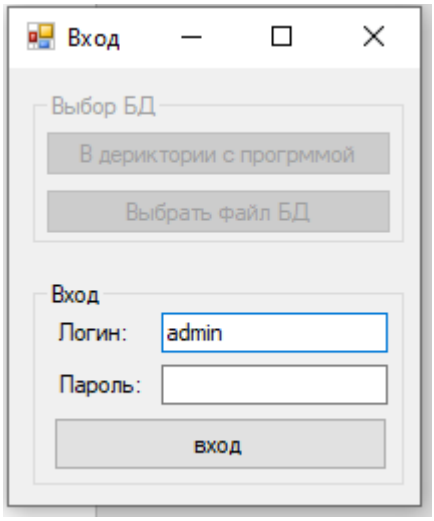

Рис. 6. Окно входа.

После входа, откроектся новое окно в котором доступен весь функционал приложения, кроме того, который выделен администратору, т.к. он вынесен в отдельное окно.

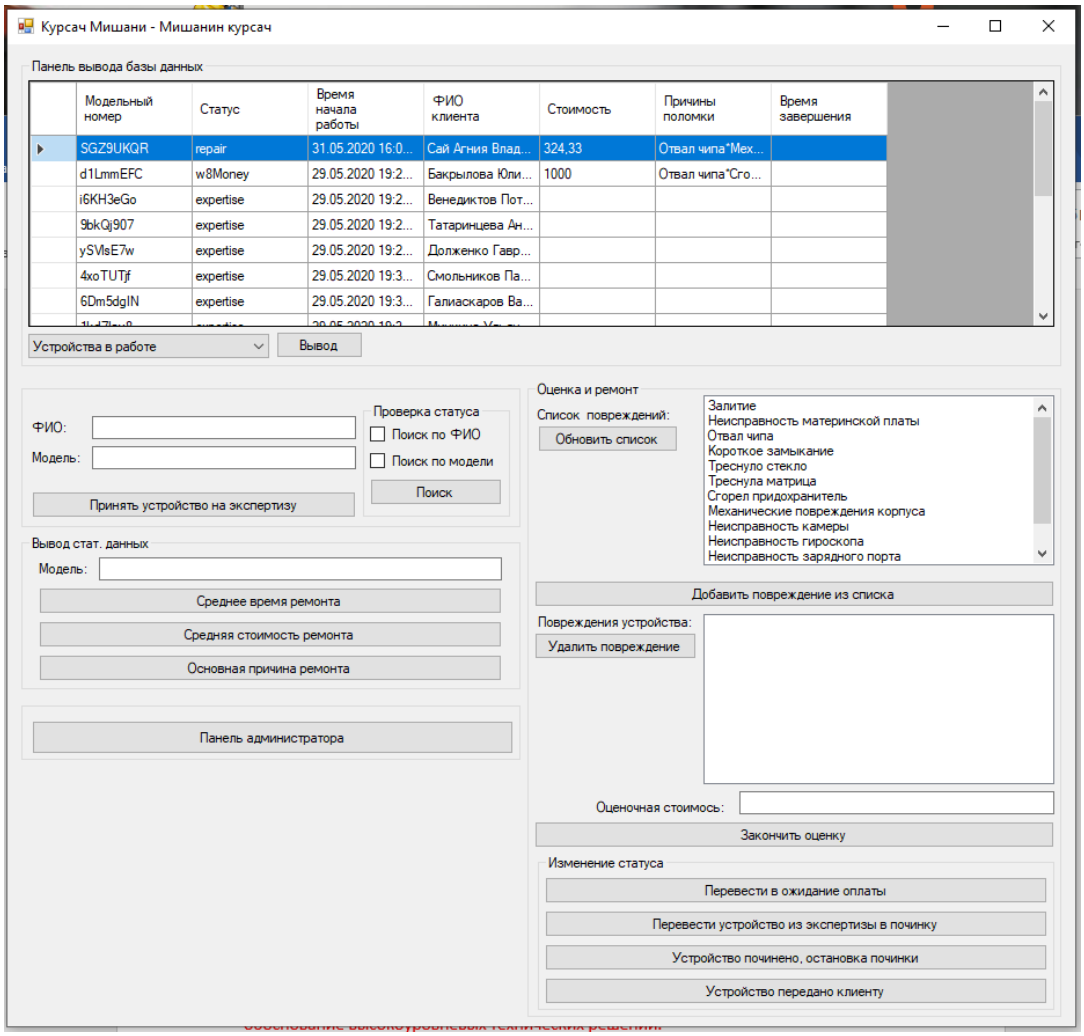

Рис. 6. Основной интерфейс приложения.

Для поиска или добавления нового устройства в базу данных устройств в работе, требуеться заполнить поля ФИО и Модель. При поиске, одно из полей может быть пустое, т.к. требуется выбрать параметры, по которым будет проходить поиск. Зелёной стрелочкой показано в каком блоке приложения находятся элементы задейсвующиеся при этих операциях.

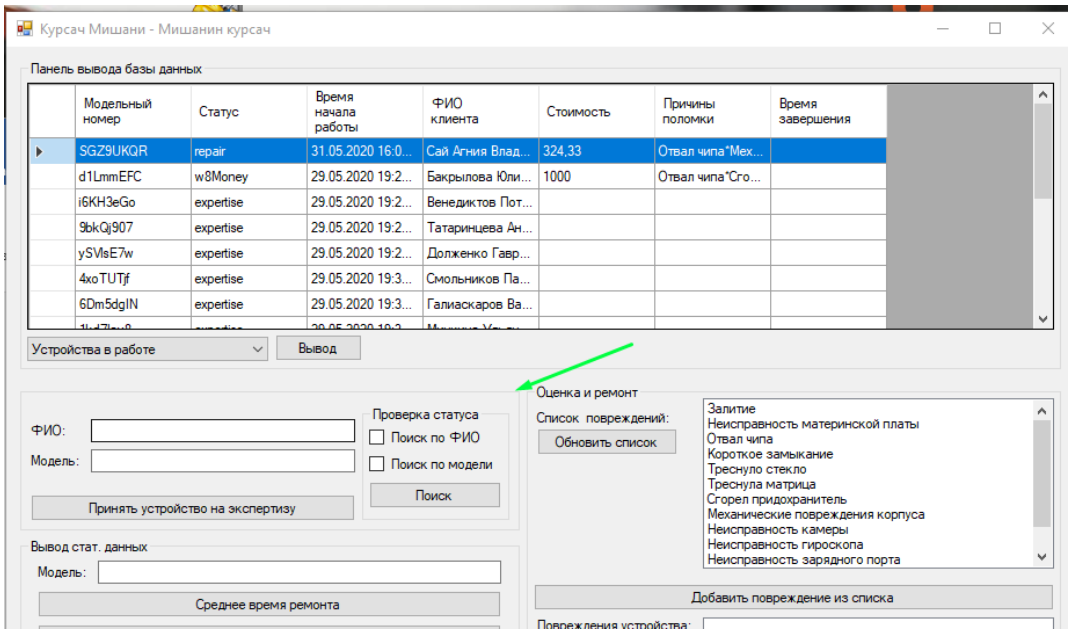

Рис. 7. Элементы, отвечающие за поиск и получение устройста.

Для вывода статистики по модели, требуется ввести номер модели и выбрать какую статистику выодить.

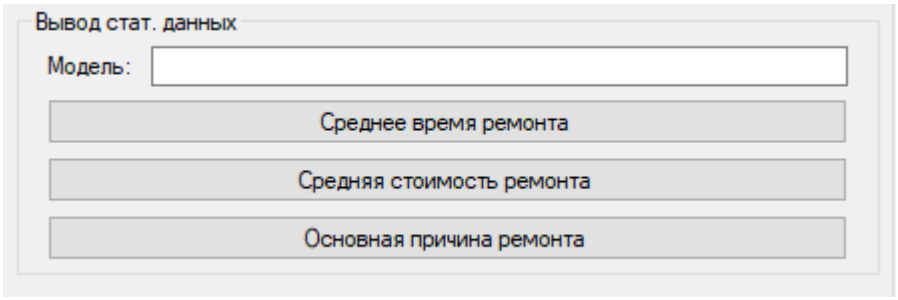

Рис. 8. Интерйейс вывода статистики.

Для изменения статуса и оценки устройства, а также при передаче устройства клиенту все действия происходят в блоке «Оценка и ремонт». В нём производятся действия по оценке ремонта, выбора неисправностей, перевода устройства из экспертизы в ожидание оплаты, из ожидания оплаты в востановление, а также его выдача клиенту.

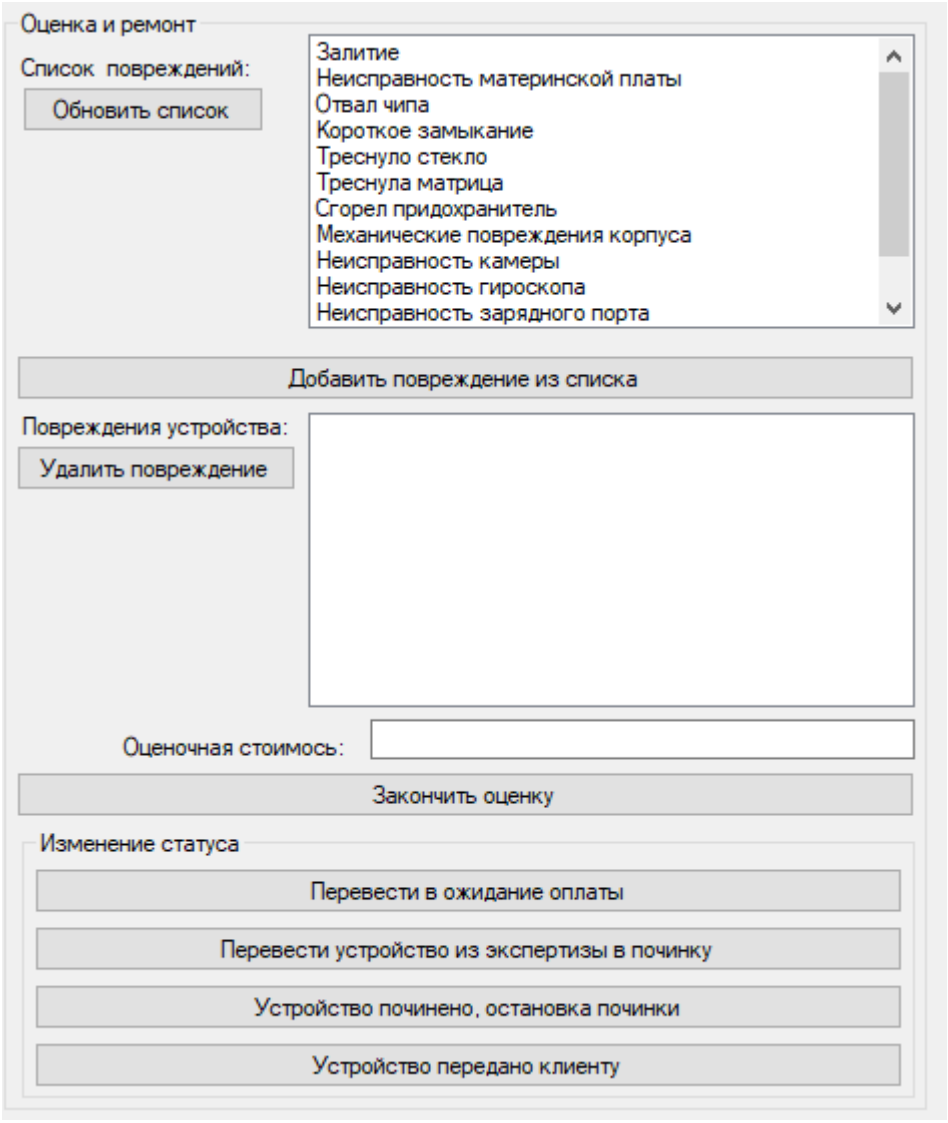

Рис. 9. Интерфейс действий над устройством.

Для добавления нового пользователя или возможной причины поломки, пользователь с правами администратора должен зайти в приложение и нажать на доступную ему кнопку «Панель администратора», после чего откроеться новое окно, с этим функцианалом. Для создания нового пользователя, требуется выбрать одну из привилегий доступных новому пользолвателю ввести логин и пароль, под которым будут входить новые пользователи. Так же из этого интерфейсам можно удалить пользователя, для этого достаточно ввести логин пользователя и нажать на кнопку удалить.

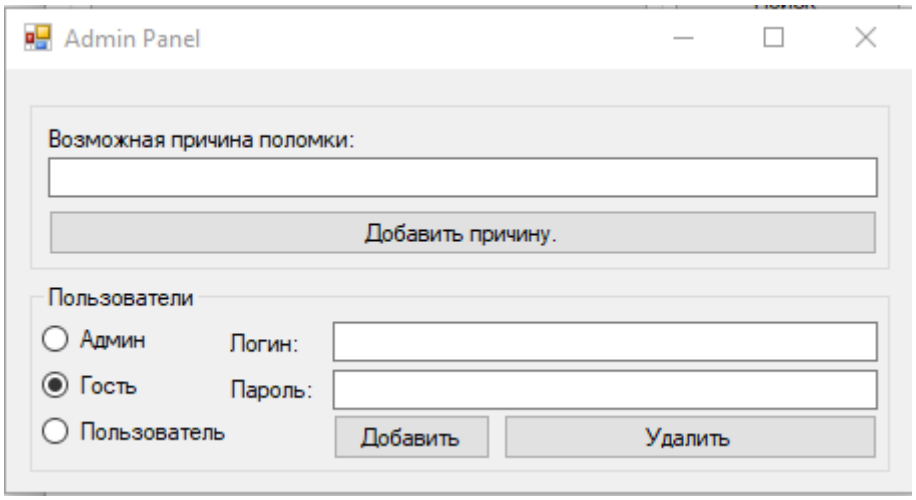

Рис. 10. Интерфейс администратора.

6. Заключение

В результате выполнения курсовой работы, была спроектирована и разработана информационная система «Сервисный центр». Реализующая работу сервисного цетра по ремонту устройств. Имеющая удобный и понятный пользовательский интерфейс.

Реализованная система имеет широкий ряд функциональных возможностей, такие как: поиск по фио, поиск по номеру модели, поиск по фио и номеру модели, создание и удаления пользователей с разными возможностями. Вывода среднего времени ремонта по модельному номеру, вывода средней стоимости ремонта по модельному номеру, вывода основной причины реомнта по номеру модели, выбора множества различных повреждений для каждого устройства, выставление оценочной стоимости, удобный перевод устройств, принятых сервисным центром в разные разные статусы.

В процессе работы над курсовой работой были закреплены навыки:

- Анализа предметной области;
- Проектирования, разработки баз данных;
- Разработки приложений на языке C#;
- Разработки приложения с проектированием пользовательского интерфейса с исспользованием Windows Forms;
- Составления SQL запросов;
- Тестирования программных компонентов;
- Работы с базами данных, таких как SQLlite;
- Подготовки технической документации;
- Работа с библиотекой System.Data.SQLite;
- Составления пользовательской инструкции;

7. Список иссползуемых источников

Нормативные документы:

1. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2017. – 32 с.;

2. О введении в действие Инструкции по организации и проведению курсового проектирования. – М.: РТУ МИРЭА, Приказ №1325 от 05.10.2018. – 17 с.;

Бумажные источники:

1. Anthony, DeBarros Practical SQL. - 1-59327-827-6 изд. - San Francisco : Nostarch, 2018.

2. Mishra R.K., Raman S.R. Introduction to PySpark SQL. - 978-1-4842-4334-3 изд. - California: Apress, 2019.

3. John Sharp Microsoft Visual C# Step by Step. - 978-1-5093-0776-0 изд. - Washington: Microsoft Corporation, 2018.

8. Приложени

8.1. Листинг

8.1.1. Файл формы авторизации

```
using System;
using System.Windows.Forms;
using System.Data.SQLite;

namespace KursWork
{
    public partial class Form2 : Form
    {
        private bool Logged = false;

        Form1 form;

        public Form2(Form1 form)
        {
            InitializeComponent();
            this.form = form;
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            GroupBoxJoin.Enabled = false;
        }

        private void JoinBtn_Click(object sender, EventArgs e)
        {
            if (LoginTB.Text != "" && PassTB.Text != "")
            {
                byte tableIndex = 4;

                string tableName =
Enum.GetName(typeof(Form1.tables), tableIndex);
```

```
                SQLiteCommand command = new
SQLiteCommand($"SELECT permission FROM {tableName} WHERE login
='{LoginTB.Text.Trim()}' AND pass='{PassTB.Text.Trim()}'",
form.GetDBConnection());
                SQLiteDataReader reader =
command.ExecuteReader();

                while (reader.Read())
                {
                    if (reader.FieldCount != 0)
                    {

form.LoginProgramm(reader[0].ToString().Trim());
                        Logged = true;
                        reader.Close();
                        this.Close();
                        return;
                    }
                }
                reader.Close();
                MessageBox.Show($"Пользователь с такими
данными не найден.", "Результат входа.", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
        }

        private void Form2_FormClosing(object sender,
FormClosingEventArgs e)
        {
            if (!Logged)
            {
                form.Close();
            }
        }
```

```
        private void SearchBtn_Click(object sender, EventArgs
    e)    //Поиск базы данных
        {
            using (OpenFileDialog openDialog = new
    OpenFileDialog())
            {
                openDialog.Filter = "База данных|*.sqllite;";
                if (openDialog.ShowDialog(this) ==
    DialogResult.OK)
                {
                    form.SetNameDB(openDialog.FileName);
                    GroupBoxJoin.Enabled = true;
                    GroupBoxSelect.Enabled = false;
                }
            }
        }

        private void HereBtn_Click(object sender, EventArgs e)
    //Выбор БД в своей дериктории
        {
            form.SetNameDB();
            GroupBoxJoin.Enabled = true;
            GroupBoxSelect.Enabled = false;
        }
    }
}
```

8.1.2. Файл формы администратора

```
using System;
using System.Windows.Forms;
using System.Data.SQLite;

namespace KursWork
{
    public partial class Form3 : Form
```

```
    {
        Form1 form;

        public Form3(Form1 form)
        {
            InitializeComponent();
            this.form = form;
            guestRB.Select();
        }

        private void AddReasonBtn_Click(object sender, EventArgs
e)     //Добавление причины в базу данных причин.
        {
            string res = reasonTB.Text.Trim();
            if (res != "")
            {
                string dateTime = DateTime.Now.ToString("yyyy-MM-
dd HH:mm:ss"); //Получение даты по формату DateTime в sql
                string tableName =
Enum.GetName(typeof(Form1.tables), 2);
                SQLiteCommand command = new SQLiteCommand($"INSERT
INTO {tableName} (Reason) VALUES ('{res}')",
form.GetDBConnection());
                command.ExecuteNonQuery();
                form.UpdDamageList();
                reasonTB.Text = "";
            }
            else
            {
                MessageBox.Show("Введите причину.", "",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
```

```
        private void AddUserBtn_Click(object sender, EventArgs e)
//Добавление нового польлователя.
        {
            string pass = PassTB.Text.Trim();
            string login = LoginTB.Text.Trim();
            if (pass != "" && login != "")
            {
                if (IsNewLogin(login))
                {
                    string dateTime = DateTime.Now.ToString("yyyy-
MM-dd HH:mm:ss"); //Получение даты по формату DateTime в sql
                    string tableName =
Enum.GetName(typeof(Form1.tables), 4);
                    SQLiteCommand command = null;

                    if (adminRB.Checked)
                    {
                        command = new SQLiteCommand($"INSERT INTO
{tableName} (login, pass, permission) VALUES ('{login}', '{pass}',
'admin')", form.GetDBConnection());
                    }
                    else if (userRB.Checked)
                    {
                        command = new SQLiteCommand($"INSERT INTO
{tableName} (login, pass, permission) VALUES ('{login}', '{pass}',
'user')", form.GetDBConnection());
                    }
                    else
                    {
                        command = new SQLiteCommand($"INSERT INTO
{tableName} (login, pass, permission) VALUES ('{login}', '{pass}',
'guest')", form.GetDBConnection());
                    }
                    command.ExecuteNonQuery();
```

```
                    LoginTB.Text = PassTB.Text = "";
                }
                else
                {
                    MessageBox.Show("Пользователь с таким логином
уже существует.", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                }
            }
            else
            {
                MessageBox.Show("Введите данные для нового
пользователя.", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
        }

        private void DelUserBtn_Click(object sender, EventArgs e)
//Удаление пользователя по его логину.
        {
            try
            {
                string tableName =
Enum.GetName(typeof(Form1.tables), 4);
                SQLiteCommand command = new SQLiteCommand($"DELETE
FROM {tableName} WHERE login='{LoginTB.Text.Trim()}'",
form.GetDBConnection());
                command.ExecuteNonQuery();
            }
            catch
            {
                ;
            }
        }
```

```
        private bool IsNewLogin(string login)   //Проверка, есть
ли данный логин в базе данных.
        {
            bool isNewUser = true;
            string tableName = Enum.GetName(typeof(Form1.tables),
4);
            SQLiteCommand command = new SQLiteCommand($"SELECT
login FROM {tableName} WHERE login ='{login}'",
form.GetDBConnection());
            SQLiteDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                if(reader.FieldCount != 0)
                {
                    isNewUser = false;
                }
            }

            reader.Close();
            return isNewUser;
        }
    }
}
```