

Частное учреждение образования
«Колледж бизнеса и права»

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ УЧЕТА
ТЕХНИЧЕСКОЙ ЭКСПЛУАТАЦИИ ЗДАНИЙ И СООРУЖЕНИЙ
ВЕРСИЯ 1.0.1

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине
«Конструирование программ и языки программирования»

КП Т.594002.401

Преподаватель

(М.А. Пархоменко)

Учащийся

(И.Д. Алейчик)

Содержание

Введение.....	4
1 Постановка задачи.....	5
1.1 Организационная сущность задачи	5
1.2 Информационная модель.....	5
2 Вычислительная система.....	9
2.1 Используемые технические средства	9
2.2 Инструменты разработки	9
3 Проектирование задачи	10
3.1 Требования к приложению	10
3.2 Концептуальный прототип.....	10
3.3 Организация данных	11
3.4 Функции: логическая и физическая организация	13
4 Описание программного средства.....	14
4.1. Общие сведения.....	14
4.2 Функциональное назначение	14
4.3 Входные и выходные данные.....	14
5 Методика испытаний	15
5.1 Технические требования.....	15
5.2 Функциональное тестирование.....	15
6 Применение.....	17
6.1 Назначение программы.....	17
6.2 Условия применения.....	17
6.3 Справочная система	17
Заключение	18
Список используемых источников	19
Приложение А Текст программы	20
Приложение В Выходные документы.....	33

					КП Т.594002.401					
Изм.	Лист	№ докум.	Подпись	Дата	Программное средство для автоматизации учета технической эксплуатации зданий и сооружений	Лит.		Лист	Листов	
Разраб.		Алейчик И.Д.				у		3	33	
Провер.						КПБ				
Реценз.										
Н. Контр.										
Утверд.										

Введение

Во время развития компьютерных технологий, хранение информации на бумаге становится все более и более невостребованным, так как информация на бумаге занимает много места, анализ этой информации требует большого количества усилий и времени и поиск нужного документа также становится трудоемким.

С целью облегчения и повышения эффективности работы возникла необходимость в использовании современных компьютерных программ.

Целью разработки проекта на тему «Программная реализация технической эксплуатации зданий и сооружений» является создание программы, которая позволит уменьшить затраты времени, используемого при выполнении действий над записями за счет автоматизации вычислительных процессов.

Пояснительная записка к курсовому проекту состоит из шести разделов, содержащих необходимую информацию по организации эксплуатации программного приложения.

В первом разделе «Постановка задачи» раскрывается организационная сущность задачи, описывается предметная область и круг задач, которые должны быть автоматизированы. Описывается задача, перечисляются основные функции программы. Строится информационная модель, отражающая сущности задачи, их свойства и взаимосвязи.

Во втором разделе «Вычислительная система» перечисляются требования к аппаратному обеспечению и конфигурации компьютера, проводится характеристика операционной системы, обоснование выбранной среды для разработки приложения. Описываются новые возможности программы, а также ее отличия от предыдущих версий.

В третьем разделе «Проектирование задачи» проводится объектно-ориентированный анализ задачи, строится концептуальный прототип системы меню, диалоговых окон и элементов управления.

В четвертом разделе «Описание программного средства» представлены общие сведения о программном средстве и его функциональном назначении.

В пятом разделе «Методика испытаний» описываются требования к техническим средствам для проведения испытаний, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Представляются результаты функционального и полного тестирования.

Шестой раздел «Применение» предназначен для описания сведений о назначении программного средства и области его применения. В этом разделе приводится структура справочной системы, а также методика ее использования.

В заключении будет проанализировано созданное программное приложение, определена степень соответствия поставленной задачи и выполненной работы.

Приложение будет содержать текст программы.

В графической части будут представлены диаграммы вариантов использования, классов, деятельности, последовательности и компонентов.

1 Постановка задачи

1.1 Организационная сущность задачи

Рассмотрим предметную область задачи. Приложение, обеспечивающее составление годовых планов текущего ремонта зданий и сооружений зависит от таких показателей, как: действия над записями такие как удалить, добавить, изменить запись.

Задачей данного курсового проекта является автоматизация процесса операций над записями о техническом состоянии зданий. Программное приложение должно автоматически выполнять выбранное действие над записями.

Исходя из исследования предметной области, основными задачами, подлежащими автоматизации, будут являться.

- возможность входа под паролем и именем;
- возможность провести регистрацию пользователя;
- выбор действия над записями;
- просмотра информации;
- поиска информации
- возможность вывода информации в файл формата docx, xlsx;
- предоставление справочной информации.

Данные необходимо сохранять в виде файла.

В настоящее время существуют аналогичные программы, предназначенные для автоматизации процесса операций над записями о техническом состоянии зданий.

Однако разработка программы, решающей подобный круг задач, только расширит спектр предлагаемых товаров на рынке программного обеспечения.

1.2 Информационная модель

Цель моделирования данных состоит в обеспечении разработчика информационной системы концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются диаграммы "сущность-связь" (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных. Нотация ERD была впервые введена П. Ченном и получила дальнейшее развитие в работах Баркера. Диаграмма «сущность-связь» представлена на рисунке 1.1.

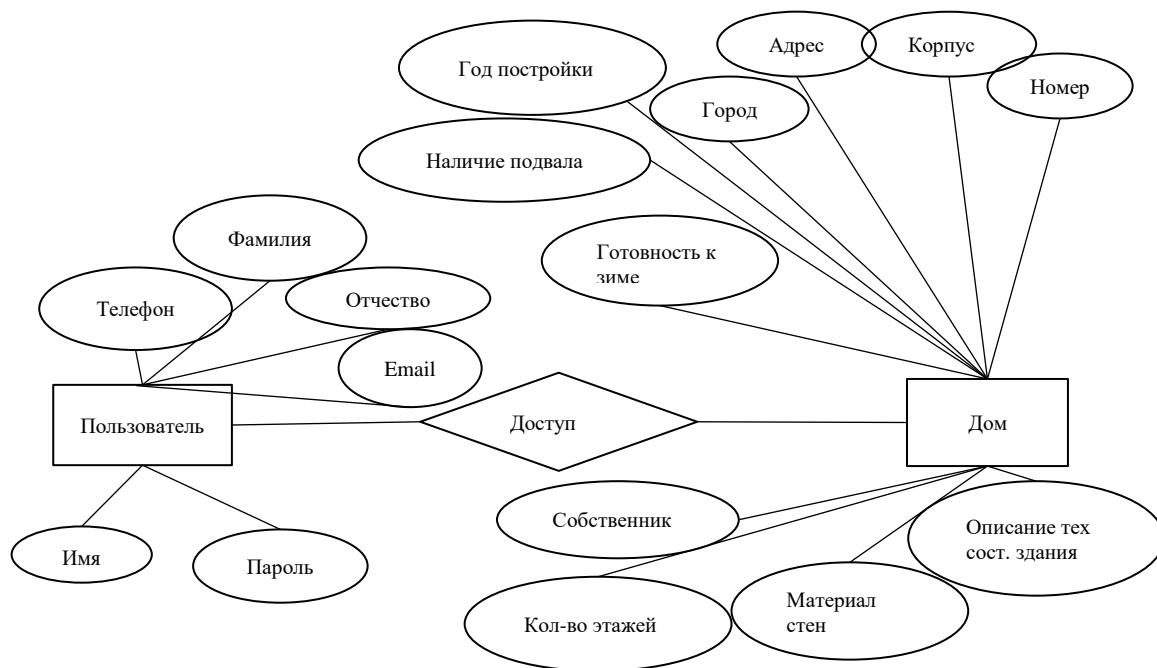


Рисунок 1.1 – Диаграмма «сущность-связь»

Исходя из исследования предметной области, можно выделить следующие сущности разработки: «пользователь» и «здание»

Для сущности «User» атрибутами будут являться:

- фамилия, имя, отчество;
- email;
- телефон;
- пароль.

Для сущности «House» можно выделить следующие атрибуты:

- адрес;
- корпус
- номер здания
- город;
- собственник;
- эксплуатационная организация
- год постройки
- материал стен
- кол-во этажей
- наличие подвала
- готовность к зиме
- председатели
- представители
- описание

Суть диаграммы вариантов использования состоит в том, что проектируемая система представляется в виде множества сущностей или актёров, взаимодействующих с системой с помощью, так называемых, вариантов использования.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы.

Актёр представляет собой внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует её функциональные возможности для достижения определённых целей или решения частных задач. При этом актёры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актёр может рассматриваться как некоторая отдельная роль относительно конкретного варианта использования.

Данный программный продукт имеет следующие основные (Include) функции:

- оформление отчета на ведение работ в виде текстового файла в формате Microsoft Office Word, в соответствии с действующими нормативными документами;
- вывод результата на печать;
- ведение базы данных.

К вспомогательным функциям, расширяющим возможности системы, относятся следующие функции:

- предоставление шаблона отчета;
- просмотр справочной информации о программе и разработчике.

Диаграмма вариантов использования представлена в графической части на листе 1.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Диаграмма классов для проектируемой системы представлена в графической части на листе 2.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Графически диаграмма деятельности представляется в форме графа, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому.

Основная цель использования диаграмм деятельности - визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения.

Диаграмма деятельности для функции поиска информации представлена в графической части на листе 3.

Для моделирования взаимодействия объектов в UML используются соответствующие диаграммы взаимодействия. Если рассматривать взаимодействия объектов во времени, тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности.

Временной аспект поведения имеет существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели и используются диаграммы последовательности, в которых ключевым моментом является динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения: одно - слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии; второе - вертикальная временная ось, направленная сверху вниз, на которой начальному моменту времени соответствует самая верхняя часть диаграммы.

Диаграмма последовательности для проектируемой системы представлена в графической части на листе 4.

Рассмотренные ранее диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления. Особенность логического представления заключается в том, что оно оперирует понятиями, которые не имеют

самостоятельного материального воплощения. Другими словами, различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически. Они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

Основное назначение логического представления состоит в анализе структурных и функциональных отношений между элементами модели системы. Однако для создания конкретной физической системы необходимо некоторым образом реализовать все элементы логического представления в конкретные материальные сущности. Для описания таких реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели.

Диаграмма компонентов описывает объекты реального мира - компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами.

Вид диаграммы компонентов для данной проектируемой системы представлен в графической части на листе 5.

2 Вычислительная система

2.1 Используемые технические средства

Конфигурация компьютера, на котором будет разрабатываться программное приложение:

- процессор Intel Core i5 2500 Mhz;
- оперативная память DDR3 8Gb;
- жёсткий диск HDD 1 Tb;
- видеокарта nVidia GeForce GT 740m 2048Mb;
- материнская плата Lenovo;
- мышь Defender.

2.2 Инструменты разработки

Инструментами разработки будут являться:

- операционная система Windows 10 Pro Build 17134;
- среда Microsoft Visual Studio 2017 Enterprise;
- язык программирования C#, HTML, CSS, XAML;
- Edraw Max 8.4.

Операционная система Windows 10 появилась относительно недавно – она стала доступной с 29 июля 2015 года. Компания Microsoft при разработке продолжала свой путь, направленный на унификацию. Допускается установка на компьютеры, ноутбуки, планшеты, а также смартфоны и консоли.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

C# — это универсальный объектно-ориентированный язык программирования, обеспечивающий безопасность преобразования типов. Целью его создания было повышение производительности труда программистов. Поэтому в языке сбалансированы - простота, выразительность и эффективность.

HTML - стандартизированный язык разметки документов во всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

CSS - Формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

XAML - Расширяемый язык разметки для приложений - основанный на XML язык разметки для декларативного программирования приложений, разработанный Microsoft.

Edraw Max 8.4 - 2D-программное обеспечение для создания технических диаграмм, блок-схем, организационных диаграмм, диаграмм связей, сетевых диаграмм, поэтажных планов, бизнес-схем, бизнес-диаграмм и инженерных схем.

3 Проектирование задачи

3.1 Требования к приложению

Программное средство должно выполнять все основные функции, определённые на диаграмме вариантов использования:

- ведение записей о техническом состоянии зданий;
- оформление отчетов виде текстового файла в формате Microsoft Office Word или Excel, в соответствии с действующими нормативными документами;
- вывод сформированного отчета на печать;
- ведение базы данных.

В данном программном средстве в качестве средств защиты будет выступать авторизация.

Графический интерфейс должен быть простым, интуитивно понятным и дружелюбным.

3.2 Концептуальный прототип

Концептуальный прототип состоит из описания внешнего пользовательского интерфейса, а именно, элементов управления.

При создании данного приложения важную роль играют формы, так как они являются основным диалоговым средством работы пользователя. Разрабатываемое приложение будет содержать несколько форм: две основные формы и четыре дополнительных. Такая структура интерфейса позволит классифицировать основные функции программы по группам.

При проектировании концептуального прототипа предполагается, что при загрузке программы первой будет загружаться форма «LoginWindow». На ней будет отображаться меню с пунктами: «Email», «Пароль», «Зарегистрироваться». При нажатии на кнопку «Вход» будет вызвана форма «MainWindow».

На форме «LoginWindow» будут находиться:

- компонент TextBox для ввода данных;
- кнопка «Вход» откроет форму «MainWindow», в которой производится выбор данных для заполнения;

Форма «LoginWindow» представлена на рисунке 3.1

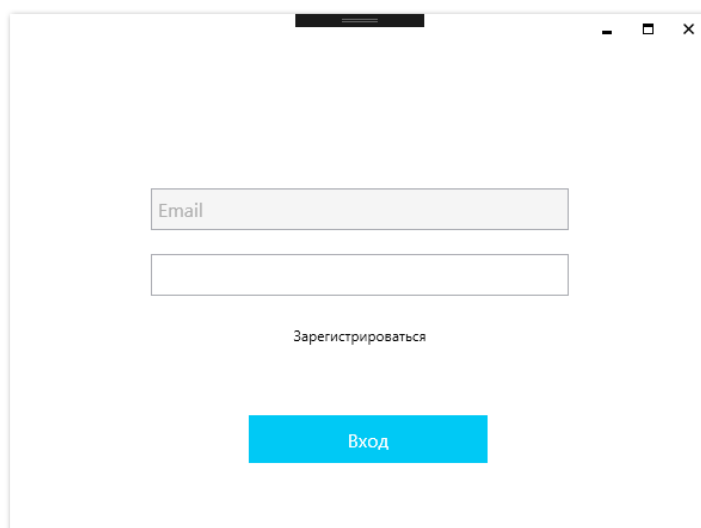


Рисунок 3.1 – Форма «LoginWindow»

Форма «MainWindow» будет содержать:

- кнопку «Menu» для получения всплывающего меню с объектами контекста данных в элементе управления «ListBox»;
- кнопку «+» для получения меню ввода данных и передачи в базу данных;
- кнопку «Delete» для удаления объекта с данными;
- кнопку «Edit» для перехода на страницу, на которой можно выполнить редактирование данных;
- кнопка «Form» формирует отчет с выводом окна сохранения, где можно задать формат файла и путь сохранения;

– TextBox для вывода данных о зданиях и их техническом состоянии;

Страницы «EditPage» имеют одинаковую структуру. Каждая из них содержит:

- кнопку «Сохранить» для фиксирования внесённых изменений в базе данных;
- компонент TextBox для представления соответствующей информации.
- компонент ScrollViewer предназначен для прокрутки страницы;

Практически на всех формах используется компонент Label для осуществления соответствующих надписей на них и создания ссылок на соответствующие страницы.

Форма «MainWindow» представлена на рисунке 3.2

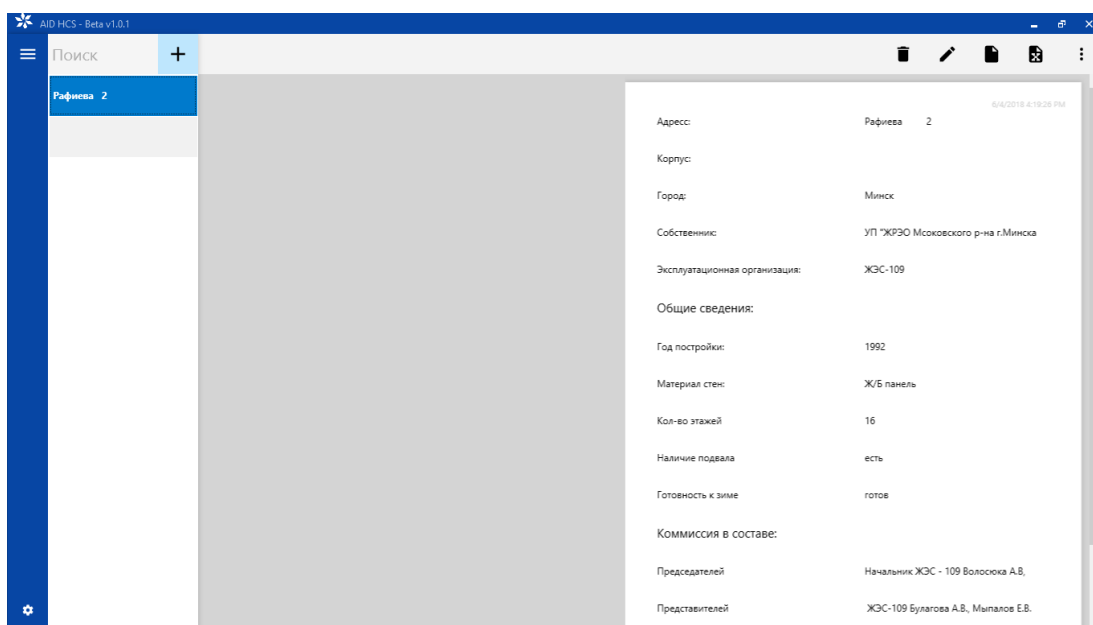


Рисунок 3.2 – Форма «MainWindow»

3.3 Организация данных

Организация данных подразумевает создание модели данных, главными элементами которой являются сущности и их связи.

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двумерной таблицы, в которой строки соответствуют записям, а столбцы - атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация — это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

В разделе 1.2 на рисунке 1.1 представлена схема Баркера-Чена, на которой отображены главные сущности задачи: «User», «House». Каждой сущности ставится в соответствие таблица базы данных. Для сущности «User» - таблица «Users». Для сущности «House» - таблица «Houses».

Структура базы данных разрабатываемого программного средства на данный момент включает две таблицы.

Структура данных таблиц, и их краткое описание приводится в таблицах 1-2.

Таблица «Users» хранит информацию о сотрудниках организации. Структура приведена в таблице 1.

Таблица 1 – Структура таблицы «Users»

Имя поля	Тип поля	Размер поля	Описание поля
id	int	4	Код предприятия
surname	String	255	Фамилия
name	String	255	Имя
patronymic	string	4	Отчество
phone	int	255	Телефон
email	string	255	Электронный адрес почты
password	int	4	Пароль

Таблица «Home» хранит информацию о зданиях. Структура приведена в таблице 2.

Таблица 2 – Структура таблицы «Houses»

Имя поля	Тип поля	Размер поля / маска ввода	Описание поля
FirstName	String	255	Имя собственника
LastName	string	255	Фамилия собственника
Num	int	4	Номер
Address	string	255	Адрес
Korp	string	255	Корпус
City	string	255	Город
Owner	string	255	Собственник
Exp_Org	date/time	##.##.####	Эксплуатационная организация
Year	string	255	Год постройки
Mat_Walls	string	255	Материал стен
Count_Floor	string	4	Кол-во этажей
Basement	string	255	Наличие подвала
Ready_Winter	string	255	Готовность к зиме

Продолжение таблицы 2

Predsdateli	string	255	Председатели
Predstaviteli	string	255	Представители
Description	string	255	Описание

3.4 Функции: логическая и физическая организация

Рассмотрим основные функции программы.

При нажатии на кнопку «Сформировать отчет» будет вызван метод, позволяющий сформировать отчет и представить её в формате документа Microsoft Office Word. Программный код реализации данного метода представлен ниже.

```
public ICommand Form
{
    get
    {
        return new DelegateCommand<House>((house) =>
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog(); //переменная типа класса SaveFileDialog
            string curDir = Directory.GetCurrentDirectory(); // получение текущего местоположения
            TemplateFile = String.Format("file:///{{0}}/template/TKP45Temp2.docx", curDir); //получений файла
            шаблона по текущему местположению
            var wordApp = new Word.Application();
            try {
                wordApp.Visible = false; //не открываем word
                var wordDoc = wordApp.Documents.Open(TemplateFile); //Открываем шаблон
                ReplaceWordStub("{Address}", SelectedHouse.Address, wordDoc); //заменяю указанный текст
                текстом из БД
                saveFileDialog.Filter= "Microsoft Word 2007-10 (*.docx)|*.docx | All files (*.*)|*.*"; // фильтр для
                выбора расширения файла
                if (saveFileDialog.ShowDialog() == true)
                {
                    curDir = saveFileDialog.FileName;
                    wordDoc.SaveAs(curDir); // сохранение файла
                } catch {}
            }
        });
    }
}
```

При нажатии на кнопку «Сформировать в Excel» будет вызван метод, позволяющий сформировать отчет и представить её в формате документа Microsoft Excel. Программный код реализации данного метода представлен ниже.

```
private void ExcelForm_Click(object sender, RoutedEventArgs e)
{
    Excel.Application excel = new Excel.Application();
    excel.Visible = true; // открываем excel
    Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);
    Excel.Worksheet sheet1 = (Excel.Worksheet)workbook.Sheets[1];
    sheet1.Columns.AutoFit();
    for (int j = 0; j < DataGridView.Columns.Count; j++) // считываем данные из DataGridView
    {
        Excel.Range myRange = (Excel.Range)sheet1.Cells[1, j + 1];
        sheet1.Cells[1, j + 1].Font.Bold = true;
        sheet1.Columns[j + 1].ColumnWidth = 15;
        myRange.Value2 = DataGridView.Columns[j].Header;
    }
    for (int i = 0; i < DataGridView.Columns.Count; i++) // записываем данными из DataGridView в excel
    {
        for (int j = 0; j < DataGridView.Items.Count; j++)
        {
            TextBlock b = DataGridView.Columns[i].GetCellContent(DataGridView.Items[j]) as TextBlock;
            Microsoft.Office.Interop.Excel.Range myRange =
            (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[j + 2, i + 1]; myRange.Value2 = b.Text;
        }
    }
}
```

Полный текст программы представлен в приложении А.

4 Описание программного средства

4.1. Общие сведения

- Наименование: AID HCS (AID Housing Communal Service);
- ПО необходимое для работы: Net Framework v4.6-7;
- объем занимаемой памяти приложением: 32МБ
- Процесс инсталляции: Setup.exe.

4.2 Функциональное назначение

- Программное средство предназначено для удобного и легкого ведения данных, а также создания отчетов технического состояния зданий;
- Форма входа на старте для подтверждения личности рабочего;
- Сетевая поддержка имеется.

4.3 Входные и выходные данные

К входной информации относятся номер здания, адреса, корпусы, города, собственник, эксплуатационная организация год постройки, материалы стен, кол-во этажей, наличие подвала, готовность к (зиме/весне), председатели и представители, и описание технического состояния сооружения, задаваемый вручную, и файл базы данных HousesData.json, user.db содержащий таблицы «Houses», «Users» структура которых представлена в разделе 3.3 в таблицах 1-2.

Постоянной информацией являются: наименование собственника и его адрес, наименование рабочего, наименование, фамилия и инициалы руководителя предприятия.

Выходная информация будет представлена в виде сформированного бланка. Шаблон документа представлен в приложении Б. В документе сначала указывается дата. Далее следуют такие реквизиты как наименование собственника и его подпись, далее следует город и дата создания документа. Указываем адрес, корпус (если есть), номер здания, наименование собственника здания и после этого именованная эксплуатирующей организации. Потом заполняем общие сведения здания: год постройки, материал стен, кол-во этажей, наличие подвала, готовность к зиме. Дальше следует заполнить комиссию в составе: председателей и представителей. После следует дать общую информацию о состоянии элементов здания.

5 Методика испытаний

5.1 Технические требования

Минимальные системные требования к приложению:

- 30 Мб свободного места на винчестере;
- 30 Мб свободной оперативной памяти;
- процессор Intel Pentium 2500 MHz или совместимый аналог;
- видеокарта с объемом памяти не менее 128 Мб.
- операционная система семейства Microsoft Windows NT;
- клавиатура, мышь.

5.2 Функциональное тестирование

В процессе написания программного продукта необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

Для входа в программу необходимо ввести данные и нажать на кнопку «Войти», вход представлен на рисунке 5.1

Рисунок 5.1 – Форма входа

Для входа в программу необходимо ввести верные данные, вход при введении неверных данных представлен на рисунке 5.2

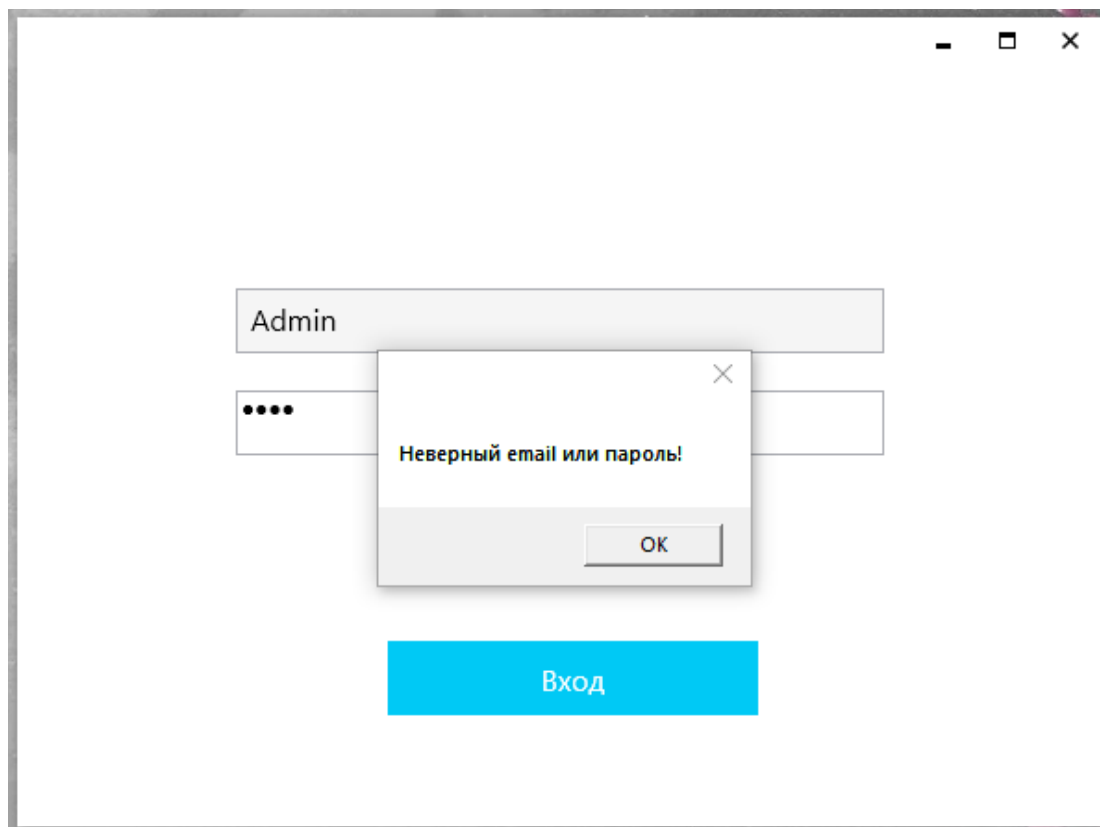


Рисунок 5.2 – Форма входа, введение неверных данных

Для регистрации необходимо нажать на кнопку «Зарегистрироваться» в форме «LoginWindow», форма регистрации представлена на рисунке 5.3.

A screenshot of a registration form titled 'Зарегистрироваться' (Register). The form is titled 'Создание учетной записи' (Create account). It contains several input fields: 'Фамилия*' (Surname*), 'Имя*' (Name*), 'Отчество*' (Patronymic*), 'Адрес электронной почты:*' (Email address:*), a field with the value '+375', and 'Придумайте пароль*' (Create a password*). Below these fields is a 'Зарегистрироваться' button. At the bottom, there is a checkbox with the text: 'Нажимая кнопку «Зарегистрироваться», я принимаю условия Пользовательского соглашения и даю своё согласие на обработку моей персональной информации.' (By clicking the 'Register' button, I accept the User Agreement and give my consent to the processing of my personal information.)

Рисунок 5.3 – Форма регистрации

6 Применение

6.1 Назначение программы

Данное программное средство позволяет ввести базу данных по зданиям и вывод информации в виде отчета или шаблона документа. Можно применять только внутри организации, при передаче информации сотруднику или другой организации.

6.2 Условия применения

Для применения данного программного средства необходимы следующие технические требования:

- процессор Intel Pentium или выше;
- минимальный объем оперативной памяти — 2048 Мбайт;
- операционная система Windows 7 и выше;
- Framework v4.6-7
- рекомендуется монитор типа VGA или с лучшей разрешающей способностью;
- устройство для чтения дисков;
- клавиатура;
- мышь.

6.3 Справочная система

Справочная система программного средства представляет собой отдельный файл «index.html». В справочной системе даны ответы на типичные вопросы, возникающие при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Справка имеет следующие разделы:

- о программе;
- как пользоваться;
- условия использования;

Структура справочной системы представлена на рисунке 6.1.

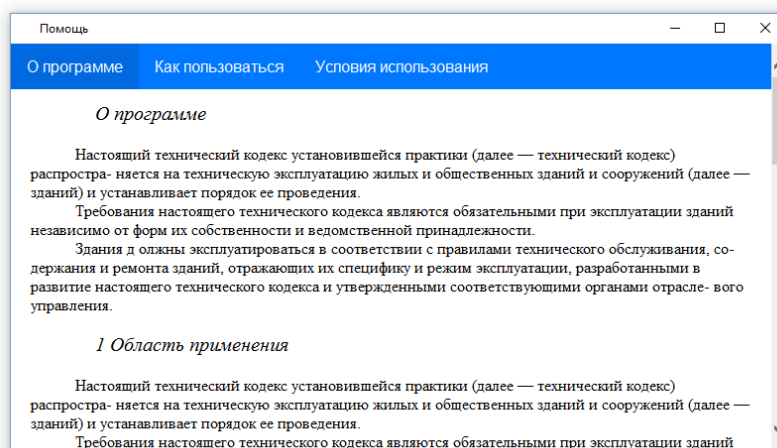


Рисунок 6.1 – Структура справочной системы

Заключение

За время практики по созданию и сопровождению программного обеспечения был изучен процесс создания, внедрения и сопровождения программных приложений в различных средах программирования. Во время практики удалось накопить неоценимый опыт в разработке приложений в интегрированной среде визуального программирования Visual C#, развить умения самостоятельно находить оптимальный метод решения задачи. Кроме того, изучить принципы организации и технологии реализации программного обеспечения, развить умения разработки программ в соответствии с требованиями технического задания, обеспечивая высокий уровень качества программного обеспечения и экономической эффективности. Получить навыки оформления комплекта документации на созданное программное обеспечение.

Список используемых источников

- 1 Орлов С. А. Технологии разработки программного обеспечения: Учебник для вузов. 4-е изд. / С. А. Орлов, Б. Я. Цилькер – СПб.: Питер, 2012. — 608 с.
- 2 Рудаков А. В., Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования / А.В. Рудаков, Г.Н. Федорова М.: Издательский центр «Академия»; 2014. — 192 с.
- 3 Тепляков С. Паттерны проектирования на платформе .NET. / С. Тепляков – СПб.: Питер, 2015. — 320 с.
- 4 ГОСТ 19701-90 Схемы алгоритмов, программ, данных. – М.: из-во стандартов, 1990
- 5 ГОСТ 19301-2000 Программа и методика испытаний. – М.: из-во стандартов, 2000
- 6 ГОСТ 19401-2000 Текст программы. – М.: из-во стандартов, 2000
- 7 ГОСТ 19402-2000 Описание программы. – М.: из-во стандартов, 2000
- 8 ГОСТ 2.105-95 Общие требования к тестовым документам. – М.: из-во стандартов, 1995
- 9 Руководство по программированию [Электронный ресурс] / Режим доступа: msdn.microsoft.com/ru-ru

Приложение А
(обязательное)
Текст программы

EditViewModel.cs

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using TKP.Model;
using Microsoft.Win32;
using System.Collections.ObjectModel;
using System.Windows.Controls;
using System.Windows.Input;

namespace TKP.ViewModel
{
    class EditViewModel : BaseViewModel
    {
        public House HouseInfo { get; set; }
        public DelegateCommand<Window> Save //сохранение изменений
        {
            get
            {
                return new DelegateCommand<Window>((w) =>
                {
                    foreach (var key in HouseInfo.KeyWords)
                    {
                        if (DataBase.GetInstance().KeyWords.FirstOrDefault(s => key.Value ==
s) == null)
                        {
                            DataBase.GetInstance().KeyWords.Add(key.Value);
                        }
                    }
                    w?.Close();
                });
            }
        }
    }
}
```

LoginViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data.SQLite;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using TKP.ViewModel;
namespace TKP.Views
{
    /// <summary>
    /// Логика взаимодействия для RegistrationWindow.xaml
    /// </summary>
    public partial class RegistrationWindow : Window
    {
        public RegistrationWindow()
        {
            InitializeComponent();
            RegistrationViewModel vm = new RegistrationViewModel();
            this.DataContext = vm;
            if (vm.CloseAction == null)
                vm.CloseAction = new Action(this.Close);
        }

        private void Button_Click(object sender, RoutedEventArgs e) //регистрация пользователя
        {
            var reg = new RegistrationWindow();
            SQLiteConnection connect = new SQLiteConnection(@"Data Source=users.db;
version=3");
            string sql = "INSERT INTO users (Name, Surname, Patronymic,Email,Password) VALUES
(@Name, @Surname, @Patronymic,@Email,@Password)";
            SQLiteCommand cmd_SQL = new SQLiteCommand(sql, connect);

            cmd_SQL.Parameters.AddWithValue("@Name", txtEmail.Text);
            cmd_SQL.Parameters.AddWithValue("@Surname", txtSurname.Text);
            cmd_SQL.Parameters.AddWithValue("@Patronymic", txtPatronymic.Text);
            cmd_SQL.Parameters.AddWithValue("@Email", txtEmail.Text);
            cmd_SQL.Parameters.AddWithValue("@Phone", txtPhone );
            cmd_SQL.Parameters.AddWithValue("@Password", txtPassword.Text);
            try
            {
                connect.Open();
                int n = cmd_SQL.ExecuteNonQuery();
            }
            catch (SqlException ex)
            {
                throw new ApplicationException("error insert new_tovar", ex);
            }
            finally
            {
                connect.Close();
            }
            this.Close();
        }
    }
}
```

```

using DevExpress.Mvvm;
using Newtonsoft.Json;
using System;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Data;
using System.Windows.Input;
using System.Windows;
using TKP;
using TKP.Model;
using TKP.Views;
using System.Windows.Controls;
using Microsoft.Office.Core;
using Word = Microsoft.Office.Interop.Word;
using Microsoft.Win32;
using System.Collections.Generic;
using System.Text;
using System.Windows.Documents;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Runtime.InteropServices;
using System.Windows.Interop;
using TKP.ViewModel;
namespace TKP.ViewModel
{
    public class MainViewModel : BaseViewModel
    {
        public ObservableCollection<House> Houses { get; set; }
        public ICollectionView HousesView { get; set; }
        public House SelectedHouse { get; set; }
        List<string> _source = new List<string> { "готов", "не готов" };
        public List<string> Source
        {
            get { return _source; }
        }
        string _theSelectedItem = null;
        public string TheSelectedItem
        {
            get { return _theSelectedItem; }
            set { _theSelectedItem = value; }
        }
        // Поиск
        private string _SearchText { get; set; }
        public string SearchText
        {
            get => _SearchText;
            set
            {

```

```

try
{
    _SearchText = value;
    HousesView.Filter = (obj) =>
    {
        if (obj is House house)
        {
            switch (SearchText.FirstOrDefault())
            {
                case '@': return house.KeyWords.FirstOrDefault(s =>
s.Value.ToLower().Contains(SearchText.Remove(0, 1).ToLower())) != null;
                case '$':
                    if (DateTime.TryParse(SearchText.Remove(0, 1), out DateTime date))
                        return house.PublishData.Date == date.Date;
                    return false;

                default: return house.Address.ToLower().Contains(SearchText.ToLower()) ||
house.Num.ToLower().Contains(SearchText.ToLower());
            }
        }

        return false;
    };
    HousesView.Refresh();
}
catch { }
}
}
// Frame навигация
private Page _CurrentPage;
public Page CurrentPage
{
    get => _CurrentPage;
    set
    {
        _CurrentPage = value;
    }
}
public Page Helps;
public MainViewModel()
{
    try
    {
        //Helps = new Main();
        //CurrentPage = Helps;
        OverlayService.GetInstance().Show = (str) =>
        {
            OverlayService.GetInstance().Text = str;
        };
        //Сериализация БД JSON и запись в коллекцию ObservableCollection
        Houses = File.Exists("HousesData.json") ?
JsonConvert.DeserializeObject<ObservableCollection<House>>(File.ReadAllText("HousesData.json"
)) : new ObservableCollection<House>();
        Houses.CollectionChanged += (s, e) =>

```

```

    {
        File.WriteAllText("HousesData.json", JsonConvert.SerializeObject(Houses));
    };
    BindingOperations.EnableCollectionSynchronization(Houses, new object());
    HousesView = CollectionViewSource.GetDefaultView(Houses);
}
catch { }
}

```

#region Переменные получающие и возвращающие вводимые значения

```

private string _FirstName { get; set; }
public string FirstName
{
    get => _FirstName;
    set { _FirstName = value; }
}
private string _LastName { get; set; }
public string LastName
{
    get => _LastName;
    set { _LastName = value; }
}
private string _Patronymic { get; set; }
public string Patronymic
{
    get => _Patronymic;
    set { _Patronymic = value; }
}
private string _Num { get; set; }
public string Num
{
    get => _Num;
    set { _Num = value; }
}
private string _Korp { get; set; }
public string Korp
{
    get => _Korp;
    set { _Korp = value; }
}
private string _MatWalls { get; set; }
public string MatWalls
{
    get => _MatWalls;
    set { _MatWalls = value; }
}
private string _Basement { get; set; }
public string Basement
{
    get => _Basement;
    set { _Basement = value; }
}
private string _Predsedateli { get; set; }

```

```

public string Predsedateli
{
    get => _Predsedateli;
    set { _Predsedateli = value; }
}
private string _Predstoviteli { get; set; }
public string Predstoviteli
{
    get => _Predstoviteli;
    set { _Predstoviteli = value; }
}
private string _Address { get; set; }
public string Address
{
    get => _Address;
    set { _Address = value; }
}
private string _City { get; set; }
public string City
{
    get => _City;
    set { _City = value; }
}
private string _Owner { get; set; }
public string Owner {
    get => _Owner;
    set { _Owner = value; }
}
private string _ExpOrg { get; set; }
public string ExpOrg
{
    get => _ExpOrg;
    set { _ExpOrg = value; }
}
private string _CountFloor { get; set; }
public string CountFloor
{
    get => _CountFloor;
    set { _CountFloor = value; }
}
private string _ReadyWinter { get; set; }
public string ReadyWinter
{
    get => _ReadyWinter;
    set { _ReadyWinter = value; }
}
private string _Year { get; set; }
public string Year
{
    get => _Year;
    set { _Year = value; }
}
private string _Description { get; set; }
public string Description

```



```

{
    get => _Description;
    set { _Description = value; }
}
#endregion
//Добовление элемента в ListView
public ICommand AddItem
{
    get
    {
        return new DelegateCommand(async () =>
        {
            await Task.Factory.StartNew(() =>
            {
                Houses.Add(new House
                {
                    Address = _Address,
                    City = _City,
                    Owner = _Owner,
                    ExpOrg = _ExpOrg,
                    Year = _Year,
                    CountFloor = _CountFloor,
                    PublishData = DateTime.Now,
                    ReadyWinter = _theSelectedItem,
                    Description = _Description,
                    //FirstName = _FirstName,
                    //LastName = _LastName,
                    //Patronymic = _Patronymic,
                    Num = _Num,
                    Korp = _Korp,
                    MatWalls = _MatWalls,
                    Basement = _Basement,
                    Predsedateli = _Predsedateli,
                    Predstoviteli = _Predstoviteli,

                });
                SelectedHouse = Houses.FirstOrDefault();
                OverlayService.GetInstance().Close();
            });
        });
    }
}
DateTime date = new DateTime();
SaveFileDialog saveFileDialog = new SaveFileDialog();
public string TemplateFile;
//Формирование отчета в Word
public ICommand Form
{
    get
    {
        return new DelegateCommand<House>((house) =>
        {
            string curDir = Directory.GetCurrentDirectory();

```

```

TemplateFile = String.Format("file:///0}/template/TKP45Template.docx", curDir);
var wordApp = new Word.Application();
try
{
    wordApp.Visible = false;
    var wordDoc = wordApp.Documents.Open(TemplateFile); // замена указанного текста
шаблона на текст и бд
    ReplaceWordStub("{ Address}", SelectedHouse.Address, wordDoc);
    ReplaceWordStub("{ City}", SelectedHouse.City, wordDoc);
    ReplaceWordStub("{ Owner}", SelectedHouse.Owner, wordDoc);
    ReplaceWordStub("{ ExpOrg}", SelectedHouse.ExpOrg, wordDoc);
    ReplaceWordStub("{ Year}", SelectedHouse.Year, wordDoc);
    ReplaceWordStub("{ CountFloor}", SelectedHouse.CountFloor, wordDoc);
    ReplaceWordStub("{ PublishData}", SelectedHouse.PublishData.ToString("d"),
wordDoc);
    ReplaceWordStub("{ ReadyWinter}", SelectedHouse.ReadyWinter, wordDoc);
    ReplaceWordStub("{ Description}", SelectedHouse.Description, wordDoc);
    //ReplaceWordStub("{ FirstName}", SelectedHouse.FirstName, wordDoc);
    //ReplaceWordStub("{ LastName}", SelectedHouse.LastName, wordDoc);
    //ReplaceWordStub("{ Patronymic}", SelectedHouse.Patronymic, wordDoc);
    ReplaceWordStub("{ Num}", SelectedHouse.Num, wordDoc);
    ReplaceWordStub("{ Korp}", SelectedHouse.Korp, wordDoc);
    ReplaceWordStub("{ MatWalls}", SelectedHouse.MatWalls, wordDoc);
    ReplaceWordStub("{ Basement}", SelectedHouse.Basement, wordDoc);
    ReplaceWordStub("{ Predsedateli}", SelectedHouse.Predsedateli, wordDoc);
    ReplaceWordStub("{ Predstoviteli}", SelectedHouse.Predstoviteli, wordDoc);
    saveFileDialog.Filter = "Документ Word (*.docx)|*.docx | All files (*.*)|*.*";
    if (saveFileDialog.ShowDialog() == true)
    {
        curDir = saveFileDialog.FileName; // запись выбранного пути в переменную.
    }
    else
    {
        wordDoc.Close();
    }
    wordDoc.SaveAs(curDir); // сохранение файла по выбранному пути
    wordDoc.Close()
}
catch { }
});
}
}

//Замена ключевых слов в шаблоне на данные из БД
private void ReplaceWordStub(string stubToReplace, string text, Word.Document
wordDocument)
{
    try
    {
        var range = wordDocument.Content;
        range.Find.ClearFormatting();
        range.Find.Execute(FindText: stubToReplace, ReplaceWith: text);
    }
    catch { }
}

```

```

    }
    /// Редактирование
    public ICommand Edit
    {
        get
        {
            return new DelegateCommand<House>((house) =>
            {
                var w = new EditWindow();
                var vm = new EditViewModel
                {
                    HouseInfo = house,
                };
                w.DataContext = vm;
                w.ShowDialog();
                File.WriteAllText("HousesData.json", JsonConvert.SerializeObject(Houses));

            }, (house) => house != null);
        }
    }
    // Сортировка
    public ICommand Sort
    {
        get
        {
            return new DelegateCommand(() =>
            {
                if (HousesView.SortDescriptions.Count > 0)
                {
                    HousesView.SortDescriptions.Clear();
                }
                else
                {
                    HousesView.SortDescriptions.Add(new SortDescription("Address",
ListSortDirection.Ascending));
                }
            });
        }
    }
    // ВЫЗОВ ФОРМЫ «О ПРОГРАММЕ»
    public ICommand About
    {
        get
        {
            return new DelegateCommand(() =>
            {
                var about = new WindowAboutProgram();
                about.ShowDialog();
            });
        }
    }
    //Удаление записи
    public ICommand Delete
    {

```

```

get
{
    return new DelegateCommand<House>((house) =>
    {
        Houses.Remove(house);
        SelectedHouse = Houses.FirstOrDefault();

    }, (house) => house != null);
}
}
//ВВОД КЛЮЧЕВОГО СЛОВА В ПОЛЕ ПОИСКА
public ICommand KeyWordClick
{
    get
    {
        return new DelegateCommand<KeyWordItem>((word) =>
        {
            if (word != null)
            {
                SearchText = "@" + word.Value;
            }
        });
    }
}
//кнопка Помощь
public ICommand HelpDown
{
    get
    {
        return new DelegateCommand(() =>
        { var h = new WindowHelp(); h.Show();
        });
    }
}
}
}

```

LoginWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Configuration;
using System.Data.SQLite;
using System.Data;

using TKP.ViewModel;
using System.ComponentModel;

namespace TKP.Views
{
    /// <summary>
    /// Логика взаимодействия для LoginWindow.xaml
    /// </summary>
    public partial class LoginWindow : Window
    {
        public LoginWindow()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e) // вход в учетную запись
        {
            var h = new Home();
            SQLiteConnection sqlCon = new SQLiteConnection(@"Data Source=users.db; Version=3;");
            try
            {
                if (sqlCon.State == ConnectionState.Closed)
                    sqlCon.Open();
                String query = "SELECT COUNT(1) FROM [Users] WHERE Email=@Email AND Password=@Password";
                SQLiteCommand sqlCmd = new SQLiteCommand(query, sqlCon);
                sqlCmd.CommandType = CommandType.Text;
                sqlCmd.Parameters.AddWithValue("@Email", txtUsername.Text); // проверка данных
                sqlCmd.Parameters.AddWithValue("@Password", txtPassword.Password);
                int count = Convert.ToInt32(sqlCmd.ExecuteScalar());
                if (count == 1)
                {
                    h.Show();
                    this.Close();
                }
                else
                {
                    MessageBox.Show("Неверный email или пароль! ");
                }
            } catch (Exception ex) { MessageBox.Show(ex.Message); }
            finally { sqlCon.Close(); }
        }
    }
}
```

RegistrationWindow.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data.SQLite;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using TKP.ViewModel;
namespace TKP.Views
{
    /// <summary>
    /// Логика взаимодействия для RegistrationWindow.xaml
    /// </summary>
    public partial class RegistrationWindow : Window
    {
        public RegistrationWindow()
        {
            InitializeComponent();
            RegistrationViewModel vm = new RegistrationViewModel();
            this.DataContext = vm;
            if (vm.CloseAction == null)
                vm.CloseAction = new Action(this.Close);
        }
        private void Button_Click(object sender, RoutedEventArgs e) //регистраци пользователя
        {
            var reg = new RegistrationWindow();
            SQLiteConnection connect = new SQLiteConnection(@"Data Source=users.db;
version=3");
            string sql = "INSERT INTO users (Name, Surname, Patronymic,Email,Password) VALUES
(@Name, @Surname, @Patronymic,@Email,@Password)"; // добавление дынных в бд
            SQLiteCommand cmd_SQL = new SQLiteCommand(sql, connect);
            cmd_SQL.Parameters.AddWithValue("@Name", txtEmail.Text);
            cmd_SQL.Parameters.AddWithValue("@Surname", txtSurname.Text);
            cmd_SQL.Parameters.AddWithValue("@Patronymic", txtPatronymic.Text);
            cmd_SQL.Parameters.AddWithValue("@Email", txtEmail.Text);
            cmd_SQL.Parameters.AddWithValue("@Phone", txtPhone );
            // sqlCommand.Parameters.AddWithValue("@Position",txtPosition);
            cmd_SQL.Parameters.AddWithValue("@Password", txtPassword.Text);
            try
            {
                connect.Open();
                int n = cmd_SQL.ExecuteNonQuery();
            }
            catch (SQLException ex)
            {
                throw new ApplicationException("error insert new_tovar", ex);
            }
            finally
            {
                connect.Close();
            }
            this.Close();
        }
    }
}

```

MainWindow.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Runtime.InteropServices;
using System.Windows.Interop;
using TKP.Views;
using TKP.Model;
using TKP.ViewModel;
using Excel = Microsoft.Office.Interop.Excel;
using Word = Microsoft.Office.Interop.Word;
using Microsoft.Office.Core;
using System.IO;
namespace TKP
{
    /// Логика взаимодействия для Home.xaml
    public partial class Home : Window
    {
        public Home()
        {
            InitializeComponent();
        }
        private void btnOpenMenu_Click(object sender, RoutedEventArgs e)
        {btnCloseMenu.Visibility = Visibility.Visible; btnOpenMenu.Visibility =
        Visibility.Collapsed;} // сккрытие и раскрытие кнопок
        private void btnCloseMenu_Click(object sender, RoutedEventArgs e)
        { btnCloseMenu.Visibility = Visibility.Collapsed; btnOpenMenu.Visibility =
        Visibility.Visible;}
        private void btnHelp_Click_1(object sender, RoutedEventArgs e)
        {WindowHelp windowHelp = new WindowHelp(); windowHelp.Show();}
        private void Button_Click(object sender, RoutedEventArgs e)
        {string link = "https://vk.com/abs_software_studio";
        System.Diagnostics.Process.Start(link);}
        private void ExcelForm_Click(object sender, RoutedEventArgs e) // формирование отчета
        в excel
        {Excel.Application excel = new Excel.Application(); excel.Visible = true;
        Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);
        Excel.Worksheet sheet1 = (Excel.Worksheet)workbook.Sheets[1];
        sheet1.Columns.AutoFit();
        for (int j = 0; j < DataGridView.Columns.Count; j++)
        {
            Excel.Range myRange = (Excel.Range)sheet1.Cells[1, j + 1];
            sheet1.Cells[1, j + 1].Font.Bold = true;
            sheet1.Columns[j + 1].ColumnWidth = 15;
            myRange.Value2 = DataGridView.Columns[j].Header;}
        for (int i = 0; i < DataGridView.Columns.Count; i++)
        {
            for (int j = 0; j < DataGridView.Items.Count; j++)
            { TextBlock b = DataGridView.Columns[i].GetCellContent(DataGridView.Items[j])
            as TextBlock; Microsoft.Office.Interop.Excel.Range myRange =
            (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[j + 2, i + 1]; myRange.Value2 = b.Text;
            }
        }
    }
}

```

Приложение В
(обязательное)
Выходные документы

Приложение Г
(обязательное)

Форма акта общего (осеннего) осмотра здания (о готовности к зиме)
«УТВЕРЖДАЮ»

Собственник здания или уполномоченное лицо

личная подпись

фамилия, инициалы

« ____ » _____ 200 ____ г.
дата

АКТ

общего (осеннего) осмотра здания (о готовности к зиме)

Минск
город

04.06.2018

Здание № 2 корпус по ул.(пер.) Рафиева
находится в хозяйственном ведении, оперативном управлении УП «ЖРЭО Мсоковского р-на
г.Минска

наименование собственника или уполномоченного лица

Эксплуатационная организация ЖЭС-109

наименование

Общие сведения:

1.1 Год постройки 1992 материал стен

1.2 Количество этажей 16 наличие подвала есть

Результаты готовности здания к зиме/весне: готов

Комиссия в составе:

председателя Начальник ЖЭС - 109 Волосюка А.В.,

представителей ЖЭС-109 Булагова А.В., Мыпалов Е.В.

произвел(а) проверку готовности к эксплуатации в зимних условиях вышеуказанного здания и
установил(а):

Фундамент: удовл; Цикли: удовл; Стены наружные: удовл; Колонны: удовл; Перегородки:
удовл;

Удостоверяющий лист
электронного документа – курсовой проект

Тема КП Программное средство для автоматизации учета технической эксплуатации зданий и сооружений

Обозначение КП _____ КП Т.594002.401

Разработчик Алейчик И.Д. Руководитель Пархоменко М.А.
(Ф.И.О.) (Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

Состав электронного документа	Разработчик	Руководитель
Пояснительная записка (на бумажном носителе формата А4), ПЗ.doc		
ДВИ.docx		
Классов.docx		
Деятельности.docx		
Последовательности.docx		
Компонентов.docx		
Папка с проектом		
Тип носителя: диск		

Этикетка
для курсовых проектов

Курсовой проект

Тема «Программное средство для автоматизации учета технической эксплуатации зданий и сооружений»

КП Т.594002.401

Разработан: 01.06.2018

Утвержден: _____

Разработчик: Алейчик И.Д.

Руководитель: Пархоменко М.А.

Технические средства: видеокарта с 128Мб видеопамяти, оперативная память 2048Мбайт

Программные средства: _____

Состав документа:

Пояснительная записка – ПЗ.doc

Графическая часть – ДВИ.docx, Классов.docx, Деятельности.docx,
Последовательности.docx, Компонентов.docx

Папка Проекта – _____

Сведения о защите информации: логин Admin , пароль 5252