Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКИЙ ДИПЛОМНИЙ ПРОЕКТ

на тему:

Програмне забезпечення для розробки системи опалення будинку

ПОЯСНЮВАЛЬНА ЗАПИСКА

08-23.БДП.010.00.000 ПЗ

Виконав студент 2 курсу, групи КІ-20мсз
спеціальності123 - Комп'ютерна інженерія
Тодоренко В. О.

Керівник к.т.н., доц. каф. ОТ
Черняк О. І.
"15" 06 2022 р.

Рецензент к.т.н., доц. каф. ЗІ
Куперштейн Л. М.
"16" 06 2022 р.

Допущено до захисту
д.т.н., проф. Азаров О.Д.

"17" 06 2022 р.

ВНТУ 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітній рівень - бакалавр
Спеціальність - 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров

"08" 02 2022 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКИЙ ДИПЛОМНИЙ ПРОЕКТ
студенту **Тодоренку Вадиму Олександровичу**

1 Тема роботи «Програмне забезпечення для розробки системи опалення будинку» керівник роботи Черняк Олександр Іванович к.т.н., доцент, затверджено наказом вищого навчального закладу від **24 березня 2022** року № **66.**

2 Строк подання студентом роботи **07.06.22**.

3 Вихідні дані до роботи: технічний опис програмного застосунку, мова програмування C#, віконний додаток, середовище розробки Microsoft Visual Studio.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз технологій об'єктно–орієнтованого програмування для розробки віконних додатків, розробка структури та алгоритмів роботи програми для проектування опаленння, програмна реалізація та тестування програми.

5 Графічний матеріал - діаграма класів.

6 Консультанти розділів роботи наведені в таблиці 1.

Таблиця 1 - Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Черняк О.І., к.т.н., доц.		

7 Дата видачі завдання _____

8 Календарний план наведено в таблиці 2.

Таблиця 2 - Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів проекту (роботи)	Приміт-ка
1	Постановка задачі роботи	09.03.22	
2	Пошук матеріалів по технологіям розробки віконних додатків	10.03.22 -25.03.22	всел.
3	Структурне проектування додатку організації та оптимізації часу та робочого процессу	26.03.22 - 28.03.22	всел.
4	Обгрунтування та вибір засобів реалізації системи	29.03.22 - 04.04.22	всел.
5	Розробка головного вікна, методів створення календарю та подій	05.04.22 -29.04.22	всел.
6	Підготовка матеріалів та розробка алгоритму збереження та виведення списку подій	30.04.22 -27.05.22	всел.
7	Оформлення пояснювальної записки та ілюстративного матеріалу	28.05.22 - 06.06.22	всел.
8	Перевірка якості виконання бакалаврської роботи та усунення недоліків	07.06.22	всел.

Студент _____ Тодоренко В.О.

Керівник _____ Черняк О.І.

Анотація

Роботу виконав Тодоренко Вадим Олександрович

Пояснювальна записка містить 120 сторінок, 4 таблиці, 14 рисунків, 7 лістингів та 10 посиланнь.

Дану бакалаврську дипломну роботу присвячено створенню додатку для розробки системи опалення будинку

В роботі був виконаний аналіз найбільш сучасних методів розробки віконних додатків.

В результаті виконання роботи було зроблено програмне забезпечення для розробки системи опалення будинку, яке зроблено в середовищі Microsoft Visual studio і мові програмування C#.

Перевірка працездатності системи протестована шляхом практичного тестування системи.

Ключові слова: програмне забезпечення, система опалення, проектування, інженерія, будівництво, тестування.

Abstract

The work was performed by Todorenko Vadim Alexandrovich

Explanatory note 120 pages, 4 tables, 14 figures, 7 lists and 3 provisions The bachelor's diploma to the robot is dedicated to the creation of an addendum for the development of the scorching booth system.

In the robot, there is a victorious analysis of the largest number of modern methods for analyzing vikonnyh addenda.

As a result, a roboti created in a program protected for the development of a system scorched design was developed, which was developed in Microsoft Visual Studio using C# software.

The re-verification of the practicality of the system was protested by the way of approving the testing of the system.

Keywords: software security, scorching system, design, engineering, life, testing.

Зміст

Вступ 7
1 Аналіз технологій об’єкто-орієнтованого програмування для розробки віконних додатків 9
1.1 Створення додатків з використанням ООП підходу 9
1.2 Огляд та порівняння JSON та XML для збереження даних 13
1.3. Аналіз існуючих систем проектування опалення. 17
2 Розробка структури та алгоритмів роботи програми для проектування опалення 22
2.1 Розробка інтерфейсу користувача. 22
2.2 Проектування класів та алгоритмів обробки. 25
2.3 Огляд структури збережених даних 26
3 Програмна реалізація та тестування програми 29
3.1 Огляд існуючих мов програмування 29
3.2 Вибір IDE для розробки 32
3.3 Реалізація програми. 33
3.4 Тестування програми 36
Висновки 39
Перелік джерел посилань 40
ДОДАТОК А Технічне завдання 41
ДОДАТОК Б Код збереження програми 45
ДОДАТОК В Фрагмент коду бази даних 45
ДОДАТОК Г Код програми роботи з Exel 83
ДОДАТОК Д Код програми взаємодія з об’єктами 94
ДОДАТОК Е Протокол перевірки навчальної(кваліфікаційної) роботи 117

Вступ

Разом з розвитком цивілізації люди покращують свій рівень комфорту. Тепло є одним з аспектів поняття комфорт. Адже в надто холодному чи надто гарячому середовищі нам некомфортно. Для підтримання тепло створено безліч рішень від стародавніх багать і до сучасних систем клімат контролю. Проте з розвитком систем підтримання тепла стає все складніше їх проектувати.

Малювати схематично на куску паперу схему розведення труб, котлів, різних інших пристроїв стало просто недоцільно. Тому було створено безліч систем проектування, які зараз активно використовуються. Проте вони всі надто складні і потребують вузьких технічних знань, котрі потрібно довго отримувати. Крім того дані системи нерідко надлишкові, оскільки намагаються бути корисними всім технічним спеціальностям від будівельників і до інженерів електриків.

Тому **актуальність** даної роботи в тому, що розроблена програма буде спеціалізована для проектування опалення будинку, не потребуватиме суттєвих технічних навиків роботи з ПК та не потребуватиме складних інструкцій з інсталяції та налаштування.

Метою роботи є розробка програмного забезпечення для проектування системи опалення будинку в середовищі Visual Studio_2019, яке дозволить створювати схеми опалення будинків покімнатно

Задачі дослідження бакалаврської роботи:

- проаналізувати методи та технології розробки додатків ООП;
- проаналізувати існуючі аналоги;
- проаналізувати та обрати засоби реалізації системи;
- виконати моделювання;
- виконати розробку віконного додатку за допомогою мови програмування C#;
- протестувати розроблений додаток.

Об’єкт дослідження - процес створення програмного забезпечення для проектування опалення будинку.

Предмет дослідження - додаток, що використовується для проектування опалення будинку.

Методи дослідження бакалаврської роботи: у роботі використано принципи об’єктно-орієнтованого програмування мовою C# для реалізації підходу.

Апробація результатів бакалаврської роботи: зроблено доповідь на L науково–технічній конференції підрозділів Вінницького національного технічного університету.

Практичне значення отриманих результатів полягає в можливості використовування розробленої системи для проектування опалення будинку.

1 Аналіз технологій об’єктно-орієвантованого програмування для розробки віконних додатків

1.1 Створення додатків з використанням ООП підходу

Об'єктно-орієнтоване програмування - парадигма програмування, в якій основними концепціями є поняття об'єктів і класів. У центрі ООП знаходиться поняття об'єкта.

ООП тримається на трьох стовпах - наслідування, інкапсуляція та поліморфізм [1].

Наслідування – це механізм, який дозволяє класам нащадкам успадковувати поля класів предків.

Приклад наведено на лістингу 1.1.

Лістинг 1.1. Зразок наслідування

```
public class Base
{
public int id { get; set; }
public string name { get; set; }
public int size { get; set; }
public List<Size> sizes { get; set; }
}
public class BaseImage:Base
{
public string img { get; set; }
public int width { get; set; }
public int height { get; set; }
}
```

Як бачимо клас BaseImage наслідується від класу Base. Це означає, що у класі BaseImage є всі полі класу Base, окрім тих, що відмічені модифікатором доступу private.

Інкапсуляція – це механізм ООП, що дозволяє знизити зв’язність об’єктів. За концепцією ООП доступ напряму до полів об’єкту заборонений і потрібно працювати лише з публічними методами. Це дозволяє контролювати поля, їх стан та не допускати невірних значень (наприклад це дозволяє підтримувати значення температури більшим, за абсолютний мінімум), проте це нерідко створює проблеми і програмісти початківці нерідко нехтують цим правилом [2].

Інкапсуляція забезпечується модифікаторами доступу. Є 5 модифікаторів доступу:

- public - доступ можливий звідки завгодно;
- private - доступ лише в межах даного класу;
- protected - доступ з поточного чи наслідуваного класу;
- internal - доступ з поточної збірки;
- protected internal - доступ з поточної збірки та наслідуваних класів;
- private protected - лише наслідувані класи з поточної збірки.

Краще це все ілюструє таблиця 1.2.

Таблиця 1.2. Модифікатори доступу.

Модифікатори	Поточний клас	Наслідуваний клас з поточної збірки	Наслідуваний клас з іншої збірки	Ненаслідуваний клас з поточної збірки	Ненаслідуваний клас з іншої збірки
Private					
Private protected					
Protected					
Internal					
Protected internal					
Public					

Поліморфізм - один зі стовпів ООП, суть якого в тому, що один і той же фрагмент коду, може працювати з різними типами даних [3].

До поліморфізму включають перевантаження методів, конструкторів та операторів, явне та неявне приведення типів, проте основну його суть забезпечують абстрактні класи та віртуальні методи.

На лістингу 1.2. показано абстрактний клас

Лістинг 1.2. Абстрактний клас

```
abstract class BaseClass
   {
      protected int _x = 100;
      protected int _y = 150;
      public abstract void AbstractMethod();
      public abstract int X    { get; }
      public abstract int Y    { get; }
   }
   class DerivedClass : BaseClass
   {
      public override void AbstractMethod()
      {
         _x++;
         _y++;
      }
      public override int X
      {
         get
         {
            return _x + 10;
         }
```

Продовження лістингу 1.2

```
        }
        public override int Y
        {
            get
            {
                return _y + 10;
            }
        }
        static void Main()
        {
            DerivedClass o = new DerivedClass();
            o.AbstractMethod();
            Console.WriteLine("x = {0}, y = {1}", o.X, o.Y);
        }
    }
    // Output: x = 111, y = 161
```

Абстрактний клас - це базовий клас, від якого не можна створити екземпляру. Проте від нього можна успадковувати та перевизначати його методи [4]. Для перевизначення методів використовують ключове слово override.

Віртуальні методи - це методи, які мусять бути присутні у наслідуваних класах.

На лістингу 1.3. показано роботу з віртуальними методами

Лістинг 1.3. Віртуальні методи

```
class Person
{
    public string Name { get; set; }
```

Продовження лістингу 1.3

```
    public Person(string name)
    {
        Name = name;
    }
    public virtual void Print()
    {
        Console.WriteLine(Name);
    }
}
class Employee : Person
{
    public string Company { get; set; }
    public Employee(string name, string company) : base(name)
    {
        Company = company;
    }
}
Person bob = new Person("Bob");
bob.Print(); // виклик методу Print з класа Person

Employee tom = new Employee("Tom", "Microsoft");
tom.Print(); // виклик методу Print з класа Person
```

Також віртуальні методи можна перевизначити. Для цього використовують ключове слово override.

1.2 Огляд та порівняння JSON та XML для збереження даних

Як JSON, так і XML - це текстові формати, доступні для читання, з підтримкою створення, читання та декодування в реальних додатках. Оба є ієрархічними та

незалежними від мов текстових позначок для обміну даними. Незважаючи на загальні риси, вони відрізняються в багатьох аспектах, таких як типи даних, багатослів'я, стек інструментів і т.д.. Хоча XML є текстовим мовним розділом, який спеціалізується на бізнес-трансляції у інтернеті, JSON – це легкий формат відкритого стандарту для обміну даними, який розширюється за допомогою JavaScript. XML розшифровується як «розширена мова розмітки», а json – це «нотація об'єктів JavaScript», який є підмножиною синтаксису мови JavaScript і є повністю незалежним від мови [5].

XML (скорочення Extensive Markup Language) – це текстовий формат даних, отриманий з SGML (ISO 8879) і написаний аналогічним чином, за яким слідує HTML. Формат XML існує вже багато років і був насамперед розроблений для вирішення завдань великомасштабних електронних публікацій.

Те, що він робить - це аутсорсингові дані. Він зберігає дані в текстовому форматі, а не інтегрує їх у HTML-документ, що робить його ідеальним для представлення ієрархічних даних, таких як документи, транзакції, рахунки-фактури, книги та багато іншого.

Це незалежний обмін даними у форматі, який кодує документи у форматі, який є машиночитаним та легкочитаним. Це гнучкий спосіб створення інформаційних форматів та спільного використання структурованих даних у World Wide Web.

Це насправді підмножина мови sgml (Стандартна узагальнена мова розмітки), схожа на HTML, яка містить розмітку символів, щоб описати зміст сторінки, що дозволяє користувачам визначати свої власні мови розмітки.

Основна перевага XML - він незалежний від платформи, що означає, що користувачі можуть отримувати дані з інших програм, таких як SQL і конвертувати їх у XML, потім обмінюватися даними з іншими платформами. Простіше кажучи, це документально-орієнтована технологія, яка забезпечує можливість зберігання та відображення даних як у машино-читаному, так і в людино-читаному форматі.

Це більше схоже на мета-мову не властиву семантиці, що робить її ідеальним форматом для створення спеціальних даних та документування інформаційних форматів.

JSON (скорочення від JavaScript Object Notation) є ще одним текстовим форматом обміну даними, який використовує текстові та числові типи даних для представлення об'єктів. Це формат відкритого стандарту, що базується на підмножині мови програмування JavaScript і повністю не залежить від мови.

Це спосіб передачі об'єктів даних, що складаються з масивів типу даних та пар атрибут-значення між сервером та веб-браузером. Він використовує формат, що легко читати, для представлення простих структур даних у веб-додатках на основі коду.

Завдяки своїй гнучкості JSON краще підходить для обміну даними між веб-додатками та веб-сервісами. В якості мови розмітки XML тільки додає додаткову інформацію в простий текст, тоді як JSON, як випливає з назви, є способом представлення даних.

Він також використовується в настільних та серверних середовищах програмування. На відміну від XML, JSON використовує простий підхід для представлення структурних даних без складної нотації та алгоритмів, а також його легко освоїти, що робить його ідеальним способом створення більш інтерактивних сторінок.

Як кажуть, проблема одного - це перевага іншого. Синтаксис XML не містить семантики, але він багатослівний, що означає, що його складність ускладнює використання кожної програми. XML був розроблений для підвищення зручності читання, але не для ефективності. Синтаксис JSON набагато компактніший із встановленою семантикою, що робить його кращим форматом даних у порівнянні з XML.

XML - це спрощена версія SGML, що використовується для зберігання та представлення структурованих даних у форматі [6], який є машино зчитуваним та легкочитаним. Він призначений для підвищення зручності читання, оскільки це

мова розмітки, який додає додаткову інформацію в звичайний текст. JSON, з іншого боку, є легким форматом обміну даними, що використовується для представлення ієрархічних даних, і заснований на синтаксисі об'єкта JavaScript.

XML являється документально-орієнтованою технологією, що використовується для кодування даних у форматі, зручному для людини. Це гнучкий формат файлів, що підходить для використання в Інтернеті. JSON означає "JavaScript Object Notation", і, як випливає з назви, він базується на мові програмування JavaScript.

XML був розроблений консорціумом всесвітньої павутини (WWW) як добре документований відкритий стандартний формат, що містить набір правил про те, як зашифрувати документи у легкочитаному та машиночитаному форматі. JSON був розроблений Дугласом Крокфордом як простий, легкий формат для обміну даними.

У JSON немає початкових і кінцевих тегів, і синтаксис легший за XML, оскільки він орієнтований на дані з меншою надмірністю, що робить його ідеальною альтернативою для обміну даними по XML. З іншого боку, XML має більше символів для представлення однакових даних.

JSON - формат підтримує текст та числові типи даних, включаючи цілі та рядки [7]. Структуровані дані можуть бути представлені з використанням масивів та об'єктів. XML має пряму підтримку типу масиву, але він підтримує багато типів даних, таких як цифри, текст, зображення, графіки, діаграми і т.д.

Хоча JSON та XML є двома найбільш популярними форматами файлів для обміну даними, вони служать для різних цілей. Обидва текстові формати для читання людиною з добре документованими відкритими стандартами в World Wide Web. Однією з фундаментальних відмінностей між ними є те, що JSON орієнтований на дані, тоді як XML документований. Обидва вони прості та зрозумілі і не залежать від мови, і кожен із них краще підходить для різних завдань. Простіше кажучи, XML - це всього лише мова розмітки, яка використовується для додавання додаткової інформації у звичайний текст, тоді як

JSON - це ефективний спосіб представлення структурованих даних у легкочитаному форматі.

1.3. Аналіз існуючих систем проектування опалення.

На даний час є безліч програм для проектування опалення. Проте якщо описувати їх всіх, то знадобиться велика кількість часу та паперу, тому варто виділити кілька.

Було обрано:

- ZWCAD;
- nanoCAD BIM;
- Instal-Therm HCR;
- Rehau.

Ці всі програми в різних “вагових категоріях”, проте мають одну спільну ціль – допомогти спроектувати систему оплення.

ZWCAD - це розробка ZWSoft. Програма розрахована на професіоналів та має безліч налаштувань та функцій (рисунок 1.1).

Основні переваги:

- універсальність;
- звичний інтерфейс;
- цілком сумісний з ACAD, але є самостійною платформою;
- великий вибір програм;
- підтримка 2D та 3D;
- служба підтримки.
- підвищена продуктивність
- покращений API
- підтримка DWG-файлів
- інноваційний
- звичайний інтерфейс

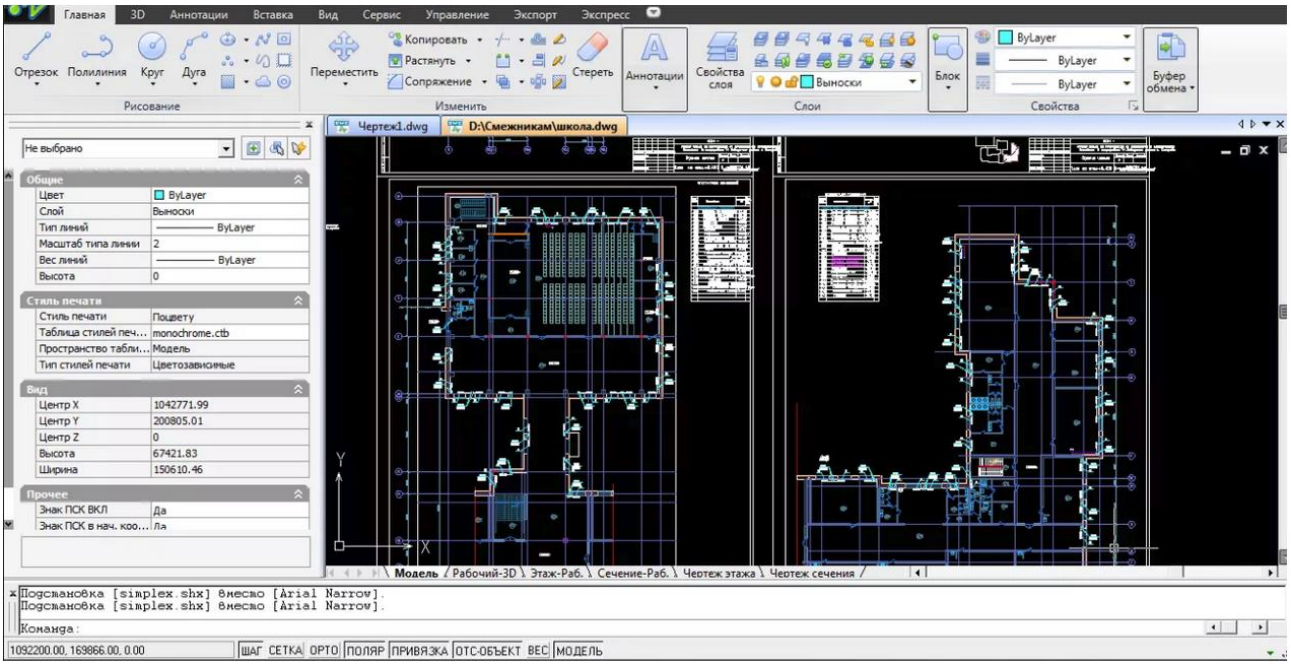

Рис. 1.1. Зразок інтерфейу ZWCAD

NanoCAD BIM - ще одна програма вищого ешелону (рисунок 1.2).

Її основними перевагами є:

- розрахунок тепловтрат з врахуванням ізоляції;

- налаштовувані виноски;

- перевірка системи (чи вистачить потужності, чи всі труби підключені…);

- вбудовується в платформу NanoCAD, яка є платформою для інженерних розрахунків.

Instal-Therm HCR - це програма призначена для проектування невеликих систем, наприклад квартири чи дачі (рисунок 1.3).

Основні переваги:

- визначення оптимального діаметра труб на певних ділянках системи, необхідних для стабілізації тиску в магістралях з урахуванням встановлених радіаторів та котла;

- гідравлічний розрахунок;

- підбір запірної арматури - муфт, трійників, фасонних виробів та з'єднувачів, а всі програми для проектування систем опалення повинні мати цю функцію, яка залежить від матеріалу виготовлення трубопроводу;

- обчислення параметрів редукторів, регуляторів тиску;

- моделювання параметрів циркуляційних течій на ділянках магістралі, вибір елементів регулювання.

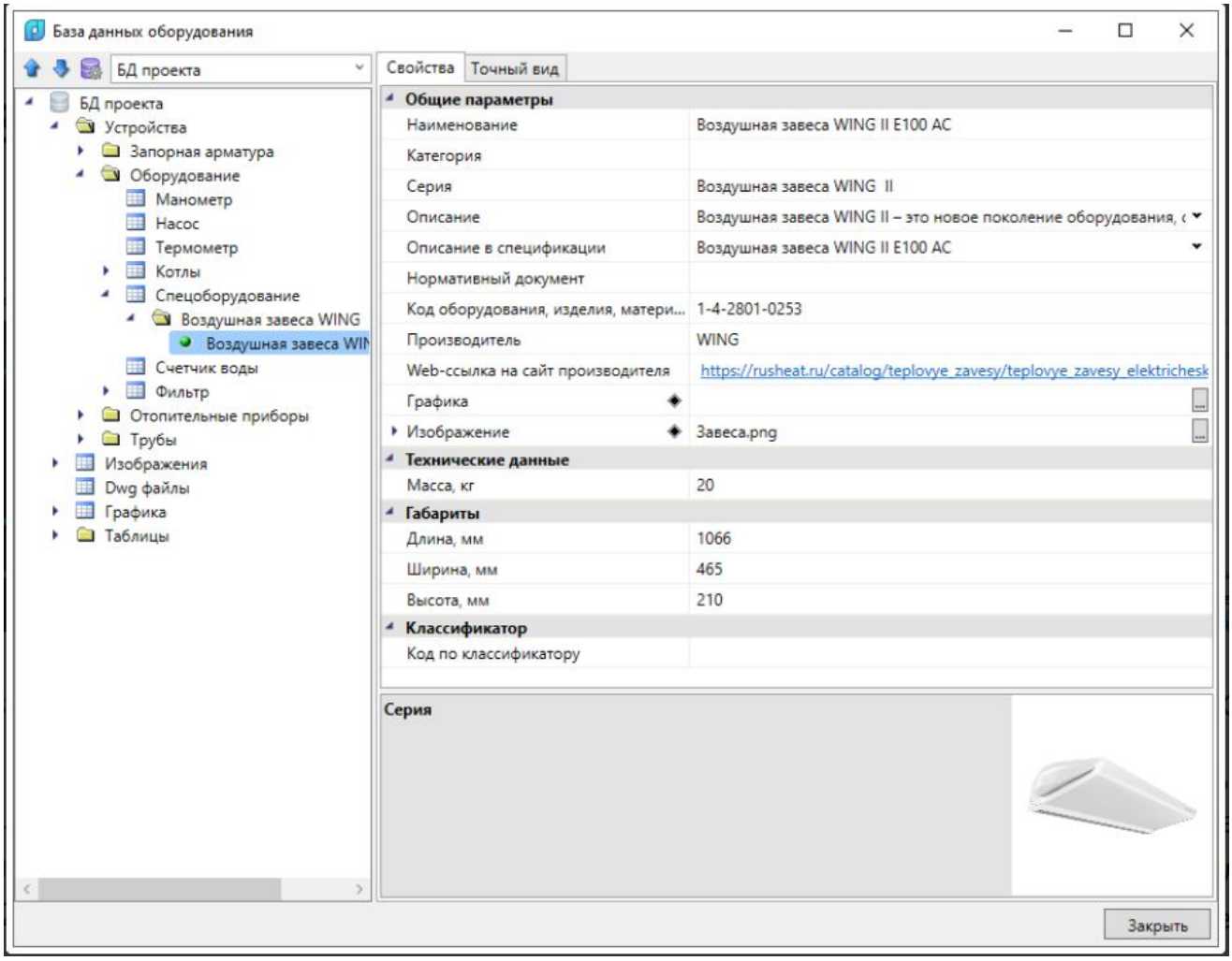

Рис. 1.2- Зразок інтерфейу nanoCAD BIM

Rehau - інженерний комплекс від однойменної компанії, де є можливість проектувати опалення (рисунок 1.4).

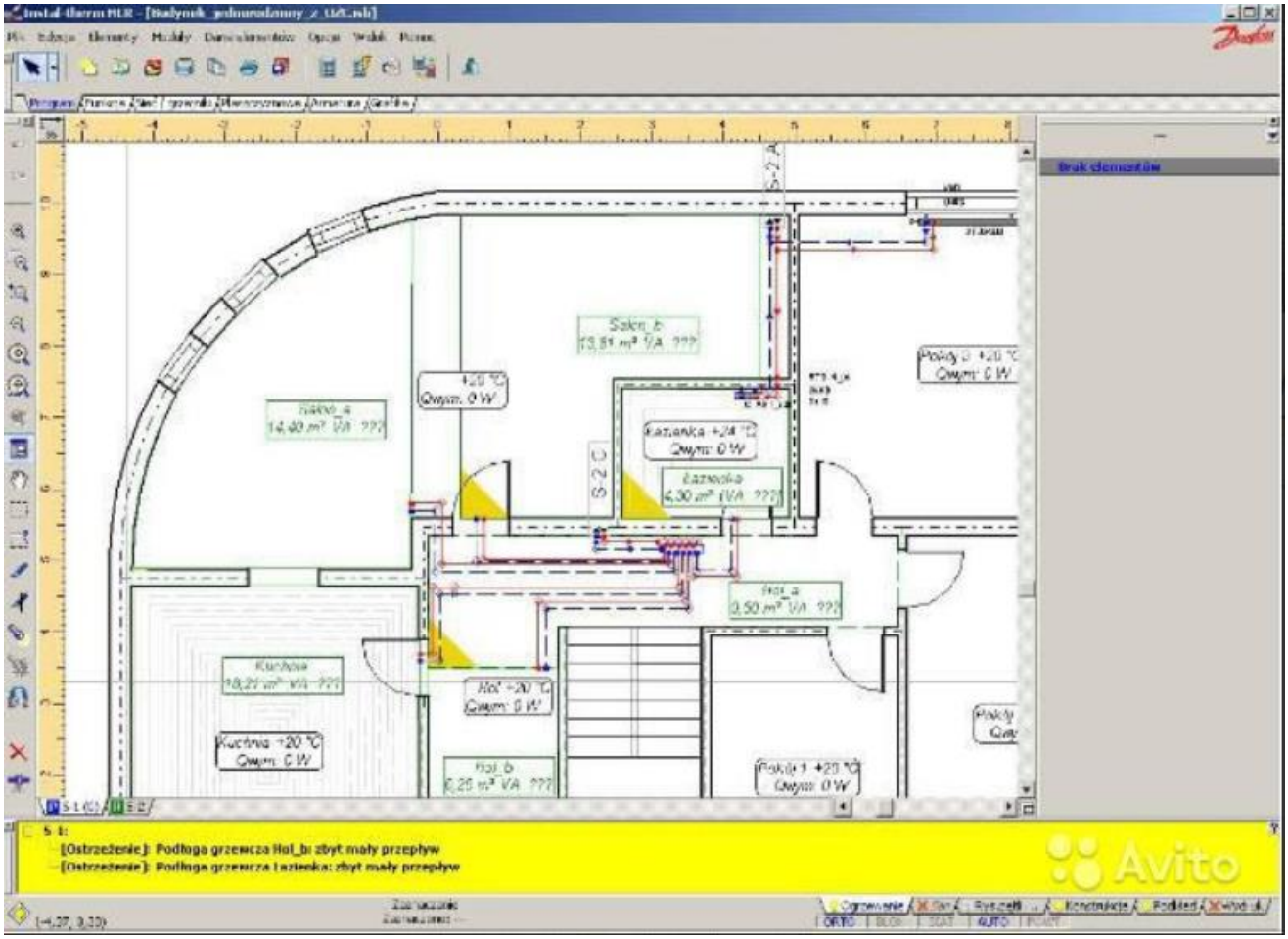

Рис. 1.3- Зразок інтерфейу Instal-Therm HCR

Особливості системи:

- це адаптована система Autocad, за допомогою якої можна створити комплексний розрахунок інженерних комунікацій житлового будинку, але окрім опалення вона включає обчислення параметрів водопостачання, каналізації, а також систему охолодження приміщення;

- ця програма не призначена для малювання опалення, але її основна функція - надання користувачеві інформації про характеристики та властивості всіх типів будівельних матеріалів і може використовуватися в комплексі з іншим програмним забезпеченням, а також при виконанні ручних розрахунків;

- незамінна програма під час проектування систем опалення, адже з її допомогою можна розрахувати теплові втрати будівлі, і тому визначити оптимальну потужність теплопостачання.

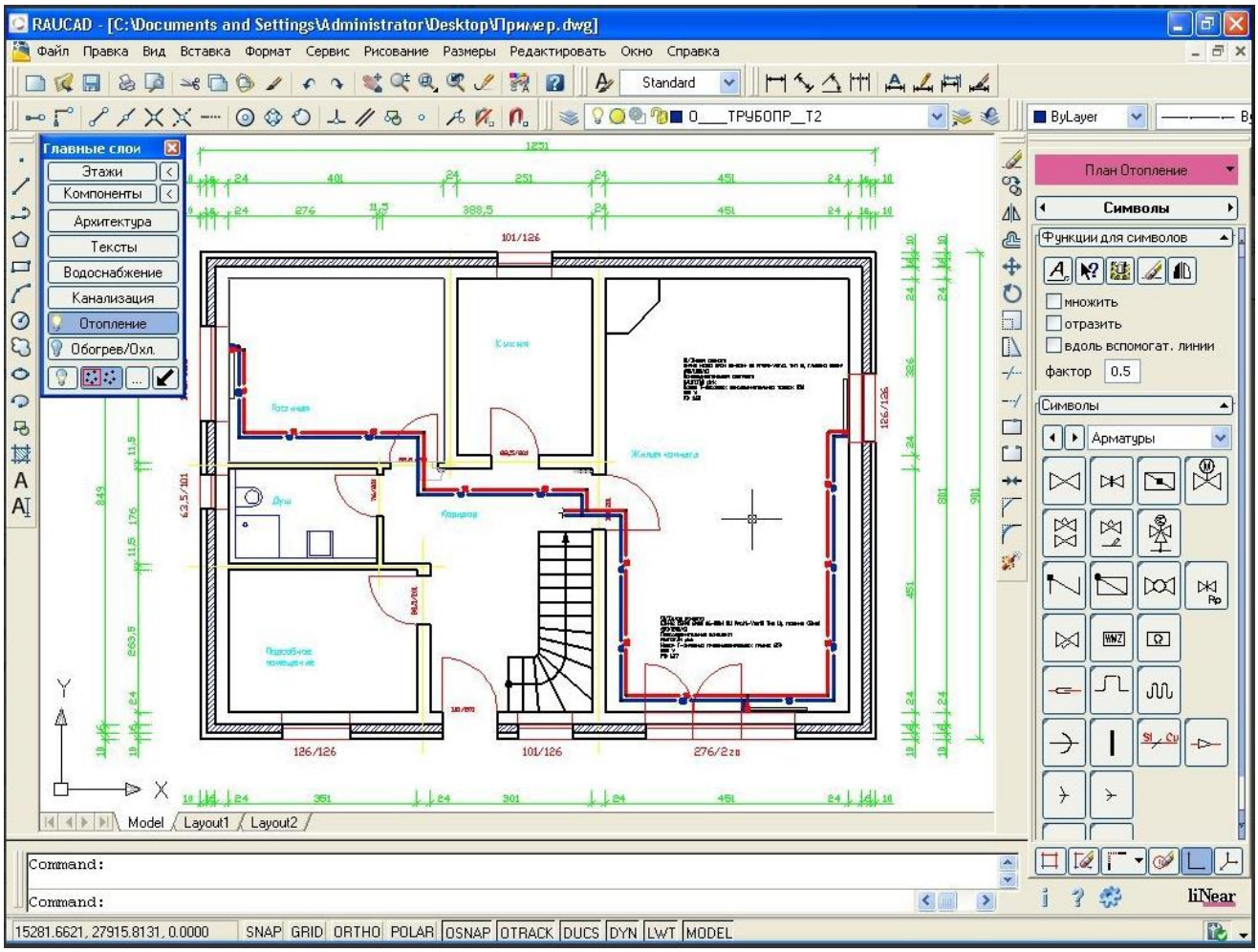

Рис. 1.4- Зразок інтерфейу Rehau

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМИ ДЛЯ ПРОЕКТУВАННЯ ОПАЛЕНННЯ

2.1 Розробка інтерфейсу користувача.

Інтерфейс користувача розроблявся з використанням технології WPF. Це найсучасніший підхід до розробки інтерфейсів прикладних програм під віндовс. Основна перевага WPF - гнучкість налаштувань компонентів та створення користувацьких елементів управління.

Нижче наведено скріншоти інтерфейсу (див. рисунки 2.1-2.5).

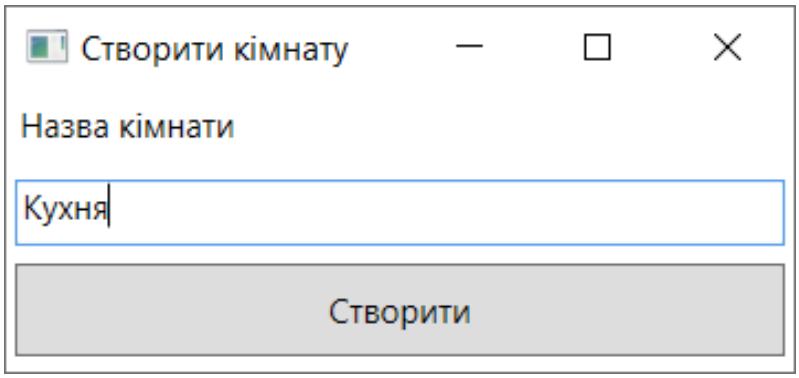

Рис.2.1- Вікно створення кімнати

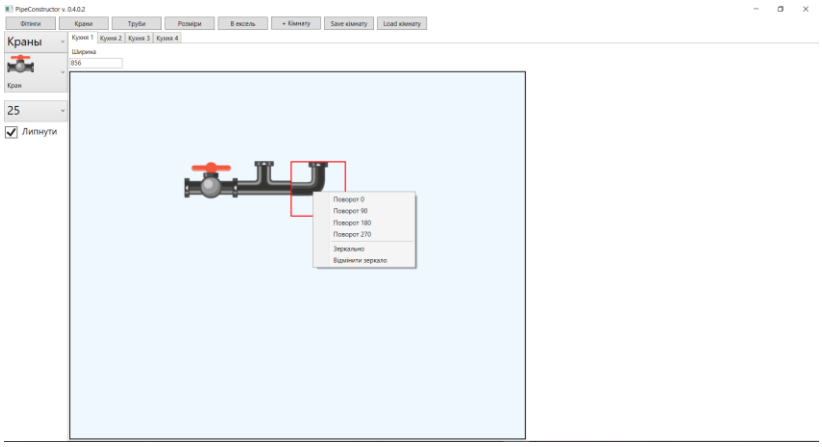

Рис.2.2- Вікно робочої області

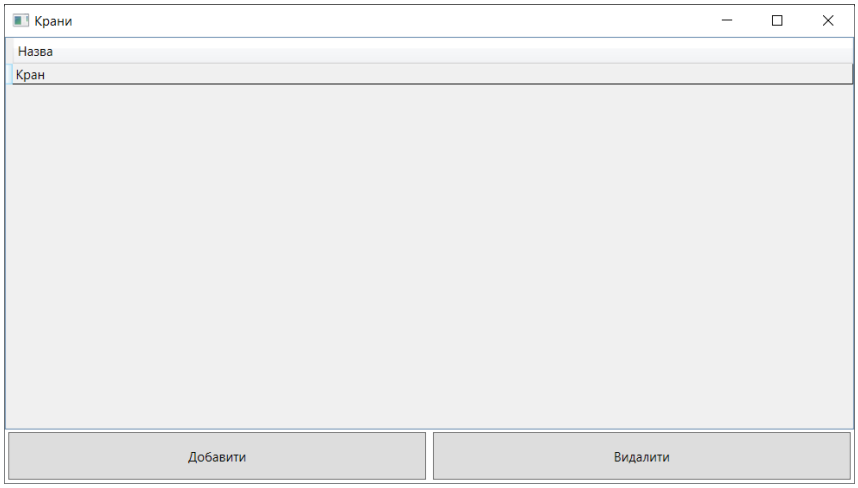

Рис.2.3- Вікно огляду кранів

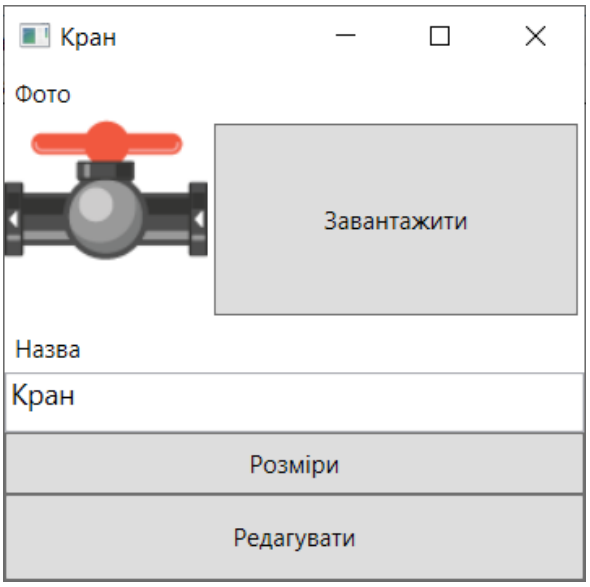

Рис.2.4- Вікно редагування кранів

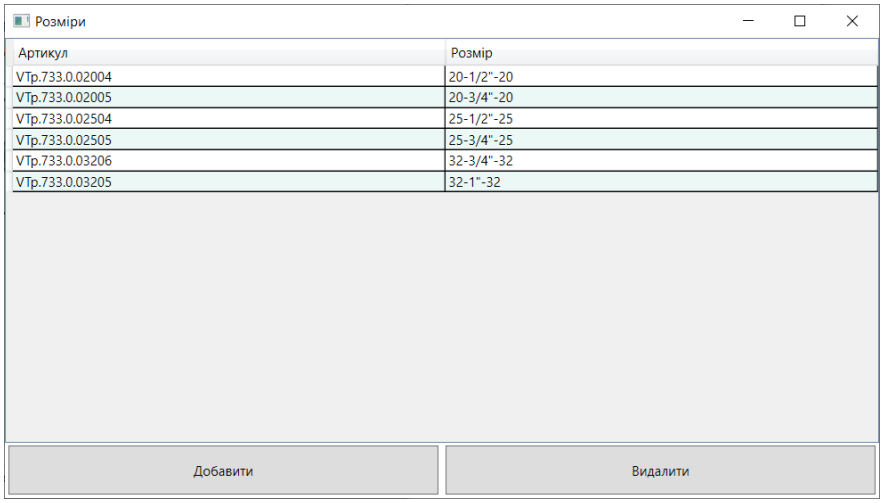

Рис.2.5 -Вікно огляду розмірного ряду

Нижче наведена шкала розмірів і таблиця специфікації (див. рисунки 2.6- 2.9).

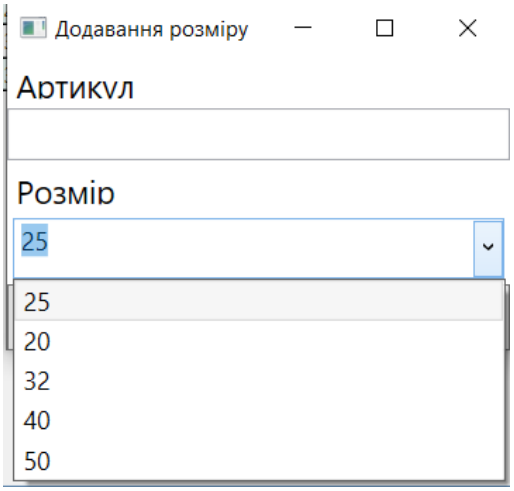

Рис.2.6-Вікно добавляння розміру до ряду

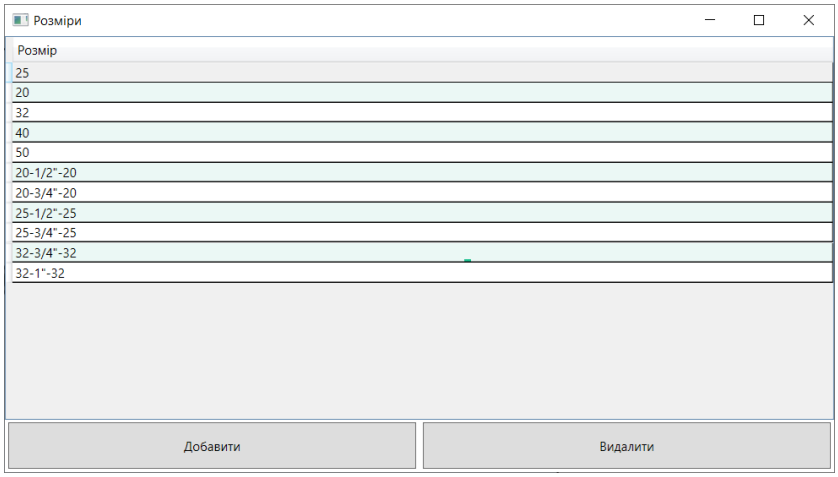

Рис.2.7-Вікно огляду розмірів

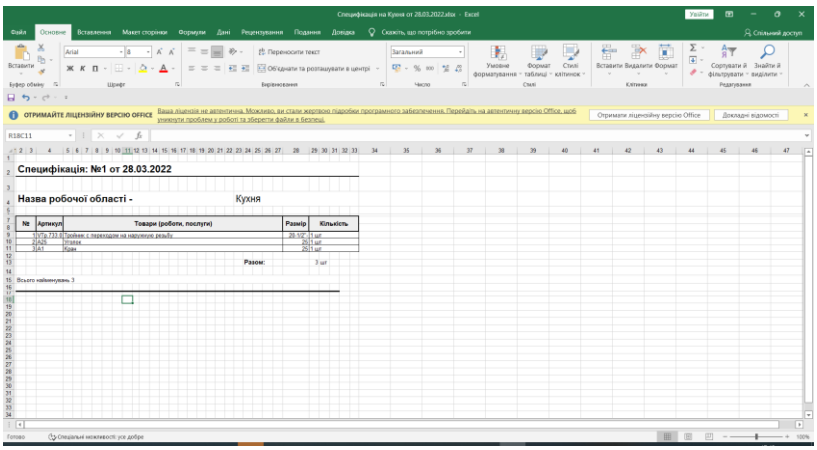

Рис.2.8-Зразок файлу специфікації

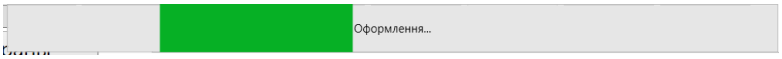

Рис.2.9-Прогрес тривалих операцій

Окрім даних форм в проекті присутні і інші, їх загалом 12, проте вони повністю або частково повторюють представлені вище форми.

2.2 Проектування класів та алгоритмів обробки.

В даному проекті представлено такі основні класи:

- DataBase;
- Base;
- BaseImage;
- Clutch;
- Tap;
- Pipe;
- Room;
- Size;
- Core;
- ExcelReporter;
- StringCipher.

Також є і інші класи, які потрібні для перетворення даних чи допоміжні для шифрування чи іншого(рисунок 2.10).

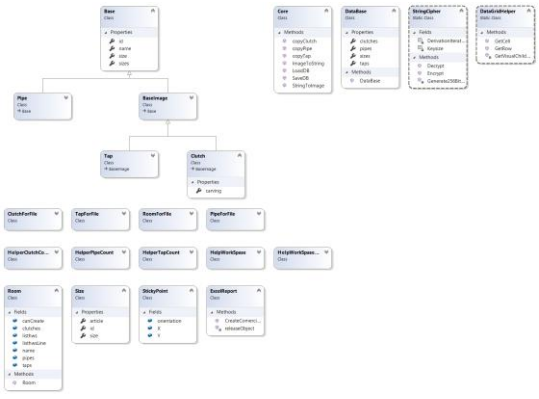

Рис. 2.10-Діаграма класів

Для читання та запису використовується клас Core та клас StringCipher. В першому є функції серіалізації та десеріалізації класу в структуру json, а в другому функції для шифрування даних. Для шифрування використовується симетричний шифр AES з ключем на 256 біт, що є нац. стандартом США та в цілому вважається достатньо надійним на сьогоднішній день. Саме шифрування відбувається статичним ключем з “сіллю”.

Перетаскування елементів на робочій області реалізоване з допомогою технології drag&drop. Аби елементи “липли” один до одного створені точки липкості (клас StickyPoint). Кожна точка має свої координати та напрям в якому має бути інша точка аби відбулося злипання. Варто зазначити що напрями двох точок мають бути протилежними і вказувати одна на одну.

У випадку, якщо відбувся збій і два елементи, які злипли треба розліпити, а вони не хочуть, то передбачена галочка відключення режиму липкості. Даний режим не призначений для роботи, оскільки потрібно дуже точно позиціонувати елементи, але він чудово підійде, якщо потрібно розліпити два елементи, і не хочеться для цього розбирати всю конструкцію.

2.3 Огляд структури збережених даних

Збереження даних відбувається з допомогою механізму серіалізації класу кімнати.

Загальний зразок серіалізованого файлу зображено на лістингу 2.1.

Лістинг 2.1. Зразок збереженого проекту

```
{
  "name": "Кухня",
  "clutches": [
    {
      "id": 1,
      "size": 1,
```

Продовження лістингу 2.1.


```
    "left": 275.8,
    "top": 157.84,
    "lvl": 0,
    "orientation": 1,
    "mirror": false
   },
   {
    "id": 2,
    "size": 5,
    "left": 297.0,
    "top": 137.44000000000003,
    "lvl": 0,
    "orientation": 2,
    "mirror": false
   },
   {
    "id": 4,
    "size": 0,
    "left": 297.0,
    "top": 237.44000000000003,
    "lvl": 0,
    "orientation": 1,
    "mirror": false
   }
  ],
  "taps": [
   {
    "id": 0,
```

Продовження лістингу 2.1.


```
      "size": 0,
      "left": 197.0,
      "top": 237.44000000000005,
      "lvl": 0,
      "orientation": 1,
      "mirror": false
    }
  ],
  "pipes": [],
  "wallLen": [
    650.0,
    650.0,
    650.0,
    650.0
  ]
}
```


Як бачимо можна виділити назву кімнати та три списки з фітінгами, кранами та трубами, а також довжини кожної стіни кімнати.

Пройдемося по списках. Перший список містить інформацію про фітінги. Кожен елемент списку містить ід елементу, його координати на робочій області, орієнтацію, зеркальність та номер стіни на якій це все повинно розміщуватись.

Далі йдуть крани, які мають практично таку ж структуру як і фітінги.

Останнім списком є список труб, єдиною їх відмінністю є дві координати: початок та кінець.

Останнім полем йде список довжин стін. По замовчуванню це значення рівне 650, проте його можна змінити. Так як труби в основному розміщують по горизонталі, то вертикаль у стін безрозмірна.

3 Програмна реалізація та тестування програми

3.1 Огляд існуючих мов програмування

На даний час є лише кілька мов програмування, котрі активно використовуються для написання комерційних програм.

До них можна віднести:

- C/C++;
- C#;
- Java;
- Python;
- Php;
- JavaScript.

Розглянемо їх детальніше.

C та C++ це одна з найстаріших мов програмування, яка досі залишається актуальною. І хоча саме C стала початком програмування таким, яким ми його знаємо зараз, проте саме C++ здобула істинну популярність. C++ є ООП мовою, тобто підтримує наслідування, поліморфізм та інкапсуляцію, а також, як і її старша сестра – асемблерні вставки. Саме вони дозволяють інтегрувати дуже старий код в нові додатки, або ж підвищити швидкодію “чітерськими методами”.

Також не варто забувати, що на C та C++ базуються інші мови програмування такі як C# та Java та інші, похідні від них.

C# - мова високого рівня. Похідна від C++ та Java. Основна відмінність від інших мов програмування – все є класом.

Також мова підтримує узагальнення, розділені класи, анонімні методи, linq, асинхронні методи та багато іншого.

Java - ООП мова, яка бере свій початок в далекому 1995 році. Наразі мовою опікується sin microsystem. Довгий час це була єдина мультиплатформна мова. Тобто програми написані на джава можна було запускати без змін як на віндовс, так і на лінукс чи мак. Проте такий підхід давав не дуже хорошу якість реалізації,

а саме програми нерідко відображалися не так як було задумано розробником на платформах відмінних від цільової [8].

Сама java працює на віртуальній машині JVM, яку потрібно попередньо інсталювати в систему. Одна з особливостей концепції віртуальної машини полягає в тому, що виключення не призводять до повного краху системи. Крім того, існують інструменти, які «приєднуються» до середовища періоду виконання і кожен раз, коли сталося певне виключення, записують інформацію з пам'яті для зневадження програми [9]. Ці інструменти автоматизованої обробки виключень надають основну інформацію щодо виключень в програмах на Java.

Python - інтерпритована ООП мова програмування з динамічною типізацією. Розробники мови Python є прихильниками певної філософії програмування, яку називають «The Zen of Python» («Дзен Пайтона»)

Текст філософії звуить так:

- гарне краще за потворне;
- явне краще за неявне;
- просте краще за складне;
- складне краще за заплутане;
- плоске краще за вкладене;
- розріджене краще за щільне;
- легкість читання має значення;
- особливі випадки не є настільки особливими, щоб порушувати правила;
- хоча практичність є важливішою за бездоганність;
- помилки ніколи не повинні проходити непомітно;
- якщо їх приховування не прописано явно;
- зустрівши неоднозначність, опирайтесь спокусі вгадати;
- має бути один - і, бажано, тільки один - очевидний спосіб зробити це;
- хоча спочатку він може бути й не очевидним, якщо ви не голландець;
- зараз - краще, ніж ніколи;
- хоча ніколи, найчастіше, - краще, ніж просто зараз;

- якщо реалізацію важко пояснити - задум поганий;

- якщо реалізацію легко пояснити - можливо, задум добрий;

- простори імен - чудова річ, тож робімо їх більше [10].

На даний час мова набирає популярності, хоча відома з 1990 року, що робить її старшою за всі представлені мови окрім C.

PHP - інтерпритована мова програмування орієнтована на веб розробку. Основною її відміннюстю є відсутність конвеншинів, оскільки вона довгий час розвивалася дуже різношерстною публікою кістяк якої складали веб розробники. “Вмирає” вже який рік поспіль, проте і досі залишається безсмертним флагманом на ринку програмування сайтів та інших веб застосунків.

Мова така ж древня як і java та бере свій початок з 1995 року. Наразі є 8ма версія мови.

У статті «PHP: a fractal of bad design»[11] представлений докладний і ґрунтовний огляд проблем в дизайні мови програмування PHP. Автор показує винятковість PHP як одної з найбільш неопрацьованих мов, проблеми в якій мають систематичний характер. Зокрема у статті продемонстровані проблеми в самій мові, бібліотеці функцій, структурах, механізмах роботи з даними, екосистемі, засобах зневадження. Вказані недоробки в безпеці, надійності, цілісності та передбачуваності.

Якщо розглядати безпеку, то як приклад спочатку порочної практики наводиться розрізненість засобів для чищення та нормалізації даних перед їхнім використанням у різних операціях, що є прекрасною підмогою для виникнення в застосунках вразливостей, що дозволяють здійснити підстановку SQL-коду або вбудовування JavaScript на сторінку. Згадки також заслуговує підхід «небезпечний за умовчанням», який тільки останнім часом став переглядатися розробниками PHP, наприклад, причиною безлічі вразливостей є використання register_globals і підтримка виконання зовнішнього коду за URL в директиві include. З проблем у самому інтерпретаторі відзначається спроба виправити у 2007 році цілочисельну вразливість через перевірку "if (size > INT_MAX) return

NULL;"; помилка в реалізації функції crypt() в PHP 5.3.7 через яку можна було зайти з будь-яким паролем; DoS-уразливість в PHP 5.4, пов'язана з виділенням пам'яті на підставі переданого користувачем значення в HTTP-заголовку Content-Length.

JavaScript - прототипна мова програмування [12]. В основному використовується для створення логіки взаємодії UI веб сайтів. Не використовується як основна мова програмування на великих проектах, за невеликим вийнятком.

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті. В перші роки існування, більшість професійних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-аматорів. Поява AJAX змінила ситуацію та звернула увагу професійної спільноти до мови, а її подальші модифікації за стандартами ES6+ внесли багато корисних можливостей, яких не вистачало для ефективного програмування. В результаті, були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та налагодження), створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером.

3.2 Вибір IDE для розробки

Для розробки програми було обрано IDE Microsoft Visual Studio 2019, яка на даний час є одним з найповніших та найстабільніших інструментів для розробки на C#.

Дана IDE підтримує надбудови серед яких найпопулярнішими є ReSharper, Review Assistant та Visual Assist X.

Також існує Visual Studio Code - це урізана версія візуал студії, проте працює не лише під віндовс, але й під лінукс та мак.

І не варто забувати про проект Mono, це проект, який розвиває IDE, яка дуже нагадує Visual Studio, проте працює лише під лінукс та мак.

3.3 Реалізація програми.

Програма реалізована з використанням мови програмування C# на visual studio 2019. При реалізації програми було використано наступні бібліотеки:

— Newton.Json;

— Mictosoft.Office.Interop.Excel.

Та інші бібліотеки, які є базовими для WPF проектів, а саме:

— System;

— System.Collections.Generic;

— System.IO;

— System.Linq;

— Microsoft.Win32;

— System.Windows.Shapes;

— System.Windows.Input;

— System.Windows.Media.

Розглянемо детальніше дані бібліотеки.

Newton.Json – бібліотека розроблена NewtonSoft, для роботи з структурами json з використанням коду на .Net включаючи C#. Бібліотека дає інструменти для серіалізації та десеріалізації структур та класів у структуру json. Дана фіча використана в проекті див лістинг 3.1.

Лістинг 3.1. Демонстрація серіалізації

```
public void SaveDB(DataBase db)
{
   string path = Environment.CurrentDirectory + "\\data";
   string json = JsonConvert.SerializeObject(db, Formatting.Indented);
   string encryptJson = StringCipher.Encrypt(json, "hsda%iw#5gd");
   File.WriteAllText(path, encryptJson); }
```

Як можна помітити в даному коді присутній виклик класу StringCipher. Даний клас відповідає за шифрування даних алгоритмом AES з ключем на 256 біт.

Окрім серіалізації було використано і десеріалізацію – це зворотній процес до серіалізації і дозволяє з структури json створити об’єкт класу. Код, що використовує даний механізм показано на лістингу 3.2.

Лістинг 3.2. Демонстрація десеріалізації

```
public DataBase LoadDB()
        {
            string path = Environment.CurrentDirectory + "\\data";
            string json = "";
            string[] SB;
            try
            {

                SB = File.ReadAllLines(path);
                DataBase tmpDB = new DataBase();
                for (int i = 0; i < SB.Length; i++)
                    json += SB[i];
                string decryptJson = StringCipher.Decrypt(json, "hsda%iw#5gd");
                DataBase db =
JsonConvert.DeserializeObject<DataBase>(decryptJson);
                return db;
            }
            catch
            {
                DataBase db = new DataBase();
                return db;
            }
        }
```

Даний код трішки більший, оскільки потрібно врахувати та обробити вийняткові ситуації такі як невірний формат файлу, відсутність оперативної пам’яті, недопустимі значення даних та інше.

Другою зовнішньою бібліотекою була Mictosoft.Office.Interop.Excel.

Це частина збірки Mictosoft.Office.Interop, яка надає доступ до роботи з екселем. Дана бібліотека використана для генерації специфікацій кімнати (див. рис. 2.8.).

Дана бібліотека відносно стара, і має безліч вразливостей, проте попри всі її недоліки вона дозволяє працювати з екселем починаючи з версії 2007 і до сучасних 2020. Використання даної бібліотеки показано на лістингу 3.3.

Лістинг 3.3. Робота з Excel

```
Excel.Application xlApp;
        Excel.Workbook xlWorkBook;
        Excel.Worksheet xlWorkSheet;
        object misValue = System.Reflection.Missing.Value;
        xlApp = new Excel.Application();
        string path = Path.Combine(Directory.GetCurrentDirectory(), "Templates\\SpecificationTemplate.xls");
        xlWorkBook = xlApp.Workbooks.Open(path);
        xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
        Excel.Range _excelCells = (Excel.Range)xlWorkSheet.get_Range("A1", "B1").Cells;

        xlWorkSheet.Cells[2, 2] = "Специфікація: №1 от " + DateTime.Now.ToShortDateString();
        xlWorkSheet.Cells[4, 23] = roomName;
```

Це частина більш великої функції, проте навіть тут зрозуміла важливість даної бібліотеки.

Крім того було написано допоміжний клас DataGridHelper, який допомагає працювати з елементами UI DataGrid, в яких міститься таблична інформація. Даний клас дуже спрощує роботу з таблицями, а саме читанням окремих стовпців, без завантаження даних цілком.

3.4 Тестування програми

Тестування - це процес аналізу та дослідження, який надає змогу виявити рівень якості продукту відносно умов, в яких він буде застосовуватись. Методика тестування також включає в себе процес пошуку дефектів, помилок, несправностей. Результат тестування оцінюється за наступними критеріями:

- відповідність вимогам, які надавалися розробниками та проектувальниками;
- відповідність вихідних даних;
- прийнятний час виконання функцій;
- практичність;
- відповідність вимогам замовника.

Кількість тестів навіть для простих програмних компонентів може бути ледь не нескінченним, тому тактика тестування має полягати в тому, що будуть проведені тільки необхідні тести з урахуванням доступного часу та ресурсів. Як результат, програмні засоби тестуються стандартним виконанням програми з метою виявлення багів, помилок або інших дефектів.

Існує багато видів тестування: одні зазвичай виконують самі розробники, а інші - тестувальники. В нашому ж випадку буде використовуватись тестування системи.

Тестування системи - це виконання програмного забезпечення в його остаточної конфігурації, інтегрованого з іншими програмними та апаратними системами.

Є дві основних методики тестування: тестування «білої скриньки» та тестування «чорної скриньки».

Методика «білої скриньки», Об'єктом тестування тут є не зовнішня, а внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним. Зазвичай аналізуються керуючі зв'язки елементів, рідше-інформаційнізв'язки. Тестування за принципом «білої скриньки» характеризується ступенем, в якому тести виконують або покривають логіку (вихідний текст) програми.

Одним із способів вивчення поставленого питання є дослідження методики «чорної скриньки», Основне місце програми тестів «чорної скриньки» це інтерфейс ПЗ.

Нижче наведено результати тестування.

Таблиця 3.1. Тесткейси

№	Назва	Хід тестування	Очікуваний результат	Результат тестування	Пройдено / непройдено
ТК1	Створення кімнати	Запустити програму. Натиснути кнопку “+ Кімнату”. Ввести назву у відповідне поле Натиснути кнопку “Створити”	Створення пустої кімнати з 4 стінами з розміром 650 кожна	Створення пустої кімнати з 4 стінами з розміром 650 кожна	+
ТК2	Добавляння крана на робочу область	Повторити все з ТК1 Обрати в області зліва крани Обрати кран Клікнути на робочій області	На робочій області з'являється піктограма відповідного крана.	На робочій області з'являється піктограма відповідного крана.	+

ТК3	Добавляння труби на робочу область	Повторити з ТК1 Обрати в області зліва труби Зажати на робочій області ліву кнопку миші та тягнути до потрібного місця Відпустити ліву кнопку мишки	На робочій області з'являється пряма лінія чорного кольору	На робочій області з'являється пряма лінія чорного кольору	+
ТК4	Перетаскування крана на інше місце робочої області	Повторити все з ТК2 Клікнути ЛКМ на кран Перетягнути кран на нове місце. Відпустити ЛКМ	Кран переміщається на інше місце	Кран переміщається на інше місце	+
ТК5	Створення ексель специфікації на кімнату	Повторити все з ТК1 Натиснути кнопку “В ексель” Дочекатися виринаючого вікна, яке спитає куди зберегти файл. Обрати адрес та ввести назву файлу.	Створиться файл з специфікацією, який містить всю інформацію про поточну кімнату	Створився відповідний файл.	+

Висновки

В даній роботі було проаналізовано ООП та його роль у створенні віконних додатків. Розглянуто поняття та приклади наслідування, інкапсуляції та поліморфізму. Оглянуто структури даних XML та JSON та виділено їх сильні та слабкі сторони. Також розглянено цілі використання даних структур на практиці. Оглянено інснуючі системи для проектування опалення, виділення їх конкурентні переваги та методи.

Спроектовано інтерфейс користувача та реалізовано його з використанням технології WPF та XAML розмітки. Спроектовано ієрархію класів та показано її на діаграмі класів. Оглянено структуру збереження даних кімнати по пунктах.

Оглянено існуючі популярні мови програмування, виділено їх основні особливості та час початку розробки. Виділено котра з мов найстаріша, а яка суб’єктивно найважливіша для історії програмування. Оглянено середовища розробки та виділено Microsoft Visual Studio 2019, як найсучаснішу IDE яка покриває всі необхідні потреби.

Розглянено основні моменти розробки, наведено зразки коду які демонструють використання тих чи інших бібліотек чи алгоритмів. Описано поняття тестування, розроблено тесткейси та показано результати тестування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Armstrong, Deborah J. (February 2006). The Quarks of Object-Oriented Development. Communications of the ACM 49 (2): 123–128. ISSN 0001-0782.

2. Англійсько-українсько-англійський словник наукової мови (фізика та споріднені науки). Частина I англійсько-українська / уклад. О. Кочерга, Є. Мейнарович. - 2010.

3. Meyer, Bertrand (1997). Object-Oriented Software Construction. Prentice Hall. ISBN 0-13-629155-4.

4. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений. М.: Вильямс, 2008.- 720с.

5. Веб ресурс [режим доступу] http://techforb.blogspot.com/2007/09/accelerating-ajax-based-web-client.html

6. Сергеев Александр Петрович. HTML і XML. Професійна робота = HTML и XML. Профессиональная работа. - М. : «Діалектика», 2004. - С. 880. - ISBN 5-8459-0676-8.

7. Роберт Тейбор. Реалізація XML Web-служб на платформі Microsoft .NET = Реализация XML Web-служб на платформе Microsoft .NET. - М. : «Вільямс», 2002. - С. 464. - ISBN 0-6723-2088-6.

8. Кей С. Хорстманн (2014). Java SE 8. Вводный курс. «Вільямс». ISBN 978-5-8459-1900-7.

9. Фрэд Лонг та ін. (2014). Руководство для программиста на Java: 75 рекомендаций по написанию надежных и защищенных программ. «Вільямс». ISBN 978-5-8459-1897-0.

10. Веб ресурс [режим доступу] https://peps.python.org/pep-0020/

11. Веб ресурс [режим доступу] https://habr.com/ru/post/142140/

12. ECMAScript Language Specification.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д.

" " 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврського дипломного проекту

«Програмне забезпечення для розробки системи опалення будинку»

08–23.БДП.010.00.000 ТЗ

Науковий керівник: доцент к.т.н.

_____ Черняк О.І.

Студент групи 1КІ-20мсз

_____ Тодоренко В.О.

ВНТУ 2022

1 Підстави для виконання бакалаврської дипломної роботи (БДР)наступні:

- актуальність розробки, яка полягає у необхідності вирішення проблеми проектування опалення в будинках.;
- наказ про затвердження теми бакалаврської дипломної роботи.

2 Мета і призначення БДП наступні:

- метою БДП є розробка програмного забезпечення для проектуваня опалення в будинках в середовищі Visual Studio 2019, яке дозволить проектувати схему опалення, наповнювати базу компонентами та створювати специфікації на окремі кімнати.
- призначення розробки полягає у виконанні бакалаврської дипломної роботи із подальшим впровадженням та розвитком.

3 Вихідні дані для виконання БДП наступні:

- технічний опис програмного застосунку;
- мова програмування C#;
- віконний додаток;
- середовище розробки Microsoft Visual Studio.

4 Вимога до виконання БДП наступна:

- створення віконного додатку;
- можливість створювати та редагувати фітінгів, труб, кранів;
- можливість проектування схем опалення;
- можливість створення специфікації на кімнати;
- можливість створення, збереження та завантаження робочих проектів на різних ПК;

5 Робота виконується за вісім етапів, етапи БДП та очікувані результати приведені в таблиці А.1.

6 Матеріали, що подаються до захисту БДП, наступні: пояснювальна записка БДП, графічні і ілюстративні матеріали, протокол попереднього захисту БДП на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до БДП українською та іноземною мовами, нормоконтроль про відповідність оформлення ДП діючим вимогам.

Таблиця А.1 - Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Постановка задачі роботи	09.03.22	09.03.22	Вступ
2	Пошук матеріалів по технологіям розробки віконних додатків	10.03.22	25.03.22	Розділ 1
3	Структурне проектування додатку проектування опалення	26.03.22	28.03.22	Розділ 2
4	Обгрунтування та вибір засобів реалізації системи	29.03.22	04.04.22	Розділ 2
5	Розробка головного вікна, методів додавання елементів на робочу область, створення ексель файлів	05.04.22	29.04.22	Розділ 3
6	Підготовка матеріалів та розробка алгоритму збереження та виведення проекту кімнат	30.04.22	27.05.22	Розділ 3, Працююча система
7	Оформлення пояснювальної записки та ілюстративного матеріалу	28.05.22	06.06.22	Пояснювальна записка
8	Перевірка якості виконання бакалаврської роботи та усунення недоліків	07.06.22	07.06.22	Презентація

7 Порядок контролю виконання та захисту БДП, графічної та розрахункової документації ДП контролюється науковим керівником згідно зі встановленими термінами. Захист ДП відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення БДП викладені в ГОСТ 2.104-95 ЕСКД «Єдина система конструкторської документації», ГОСТ 2.105-95 ЕСКД « Загальні вимоги до текстових документів»

Технічне завдання до виконання отримав _____ Тодоренко В.О.

ДОДАТОК Б

Код збереження програми

```
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace PipeConstructor
{

    public partial class WindowPipe : Window
    {
        public bool isNew;
        public DataBase db;
        public Core core = new Core();
        public Pipe pipe;
        public WindowPipe()
        {
            InitializeComponent();
        }
```

```
        private void Grid_Loaded(object sender, RoutedEventArgs e)
        {
        }

        private void Do_Click(object sender, RoutedEventArgs e)
        {
            if (isNew)
            {

                try
                {
                    pipe.name = Name.Text;
//                      pipe.size = Convert.ToInt32(Size.Text);
                }
                catch { }
                db.pipes.Add(pipe);
                Close();
            }
            else
            {
                try
                {
                    pipe.name = Name.Text;
//                      pipe.size = Convert.ToInt32(Size.Text);
                }
                catch { }
                Close();
            }
```

```
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        if (isNew)
        {
            Do.Content = "Зберегти";
            pipe = new Pipe();
            if (db.pipes.Count == 0)
                pipe.id = 0;
            else
            {
                int max = db.pipes[0].id + 1;
                for (int i = 1; i < db.pipes.Count; i++)
                {
                    if (db.pipes[i].id >= max)
                        max = db.pipes[i].id + 1;
                }
                pipe.id = max;
            }
        }
        else
        {
            Do.Content = "Редагувати";
            Name.Text = pipe.name;
//              Size.Text = pipe.size.ToString();
        }
    }
```

```
        private void Size_Click(object sender, RoutedEventArgs e)
        {
            WindowSizes ws = new WindowSizes();
            ws.db = db;
            ws.pipe = pipe;
            ws.ShowDialog();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace PipeConstructor
{
    /// <summary>
    /// Interaction logic for WindowClutches.xaml
    /// </summary>
    public partial class WindowPipes : Window
    {
```

```
public DataBase db;
Core core = new Core();
public WindowPipes()
{
   InitializeComponent();
}

private void Grid_Loaded(object sender, RoutedEventArgs e)
{
   ToForm();
}
public void ToForm()
{
   DataGridPipes.ItemsSource = null;
   DataGridPipes.ItemsSource = db.pipes;
}

private void Button_Click(object sender, RoutedEventArgs e)
{
   try
   {
      int index = DataGridPipes.SelectedIndex;
      string value = DataGridHelper.GetCell(DataGridPipes, index,
0).ToString().Substring(38);
      int id = Convert.ToInt32(value);
      Pipe pipe = db.pipes.Where(p => p.id == id).First();
      db.pipes.Remove(pipe);
      ToForm();
      core.SaveDB(db);
```

```
        }
        catch { }
    }

    private void Button_Click_1(object sender, RoutedEventArgs e)
    {
        WindowPipe wp = new WindowPipe();
        wp.isNew = true;
        wp.db = db;
        wp.ShowDialog();
        ToForm();
        core.SaveDB(db);
    }



    private void DataGridPipes_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
    {
        try
        {
            int index = DataGridPipes.SelectedIndex;
            string value = DataGridHelper.GetCell(DataGridPipes, index,
0).ToString().Substring(38);
            int id = Convert.ToInt32(value);
            WindowPipe wp = new WindowPipe();
            wp.isNew = false;
            wp.db = db;
            wp.pipe = db.pipes.Where(p => p.id == id).First();
```

```
            wp.ShowDialog();
            ToForm();
            core.SaveDB(db);
        }
        catch { }
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
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.Windows.Threading;

namespace PipeConstructor
{

  public partial class WindowProgress : Window
  {
    public int mode = 0;
```

```
public List<Clutch> clutches;
public List<Tap> taps;
public List<Pipe> pipes;
public string roomName;


public WindowProgress()
{
   InitializeComponent();
}
public DataBase db;
private void Window_Loaded(object sender, RoutedEventArgs e)
{
   WindowStartupLocation = WindowStartupLocation.CenterScreen;
   if (mode == 1)
      new Thread(() => Do()).Start();
   else
      Close();
}

public void Do()
{
   ExcelReport.CreateComercialProposalReport(db, roomName, clutches, taps, pipes, Status);
   Application.Current.Dispatcher.Invoke(
    DispatcherPriority.Background,
      new ThreadStart(delegate
      {
         Close();
```

```
            }));
        }

    }
}
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace PipeConstructor
{

    public partial class WindowSize : Window
    {
        public bool isNew;
        public List<Size> sizes;
        public Core core = new Core();
        public Size size;
        public WindowSize()
```

```
{
    InitializeComponent();
}

private void Grid_Loaded(object sender, RoutedEventArgs e)
{

}

private void Do_Click(object sender, RoutedEventArgs e)
{
    if (sizes == null)
        sizes = new List<Size>();
    if (isNew)
    {
        size = new Size();
        if (sizes.Count == 0)
            size.id = 0;
        else
        {
            int max = sizes[0].id + 1;
            for (int i = 1; i < sizes.Count; i++)
            {
                if (sizes[i].id >= max)
                    max = sizes[i].id + 1;
            }
            size.id = max;
        }
        try
```

```
            {
//                  size.article = Article.Text;
                size.size = Size.Text;
            }
            catch { }
            sizes.Add(size);
            Close();
        }
        else
        {
            try
            {
//                  size.article = Article.Text;
                size.size = Size.Text;
            }
            catch { }
            Close();
        }
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        if (isNew)
        {
            Do.Content = "Зберегти";
        }
        else
        {
            Do.Content = "Редагувати";
```

```
//                Article.Text = size.article;
                Size.Text = size.size;
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
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;

namespace PipeConstructor
{
    /// <summary>
    /// Interaction logic for WindowSizeAdd.xaml
    /// </summary>
    public partial class WindowSizeAdd : Window
    {
        public DataBase db;
        public List<Size> sizes;
```

```
ObservableCollection<string> lsizes = new ObservableCollection<string>();
public List<Size> sizesOnForm = new List<Size>();
public Size size;
public bool isNew;
public WindowSizeAdd()
{
   InitializeComponent();
}
private void Window_Loaded(object sender, RoutedEventArgs e)
{
   if (isNew)
   {
      for (int i = 0; i < db.sizes.Count; i++)
         if (sizes.Where(s => s.id == db.sizes[i].id).ToList().Count == 0)  // якщо
нема
         {
            lsizes.Add(db.sizes[i].size);
            sizesOnForm.Add(db.sizes[i]);
         }

   }
   else
   {
      lsizes.Add(size.size);
      Size.IsEnabled = false;
      Article.Text = size.article;
   }
   Size.ItemsSource = lsizes;
   Size.SelectedIndex = 0;
```

```
        }
        private void btn1_Click(object sender, RoutedEventArgs e)
        {
            if (isNew)
            {
                sizesOnForm[Size.SelectedIndex].article = Article.Text;
                sizes.Add(sizesOnForm[Size.SelectedIndex]);
            }
            else
            {
                size.article = Article.Text;
            }
            Close();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;
```

```
namespace PipeConstructor
{
    /// <summary>
    /// Interaction logic for WindowClutches.xaml
    /// </summary>
    public partial class WindowSizes : Window
    {
        public DataBase db;
        Core core = new Core();
        public Clutch clutch;
        public Tap tap;
        public Pipe pipe;
        public int choose;
        public WindowSizes()
        {
            InitializeComponent();
        }

        private void Grid_Loaded(object sender, RoutedEventArgs e)
        {
            ToForm();
        }
        public void ToForm()
        {
            DataGridSizes.ItemsSource = null;
            if (clutch != null)
            {
                choose = 1;
                if (clutch.sizes == null)
```

```
            clutch.sizes = new List<Size>();
        DataGridSizes.ItemsSource = clutch.sizes;
    }
    else if (tap != null)
    {
        choose = 2;
        if (tap.sizes == null)
            tap.sizes = new List<Size>();
        DataGridSizes.ItemsSource = tap.sizes;
    }
    else if (pipe != null)
    {
        choose = 3;
        if (pipe.sizes == null)
            pipe.sizes = new List<Size>();
        DataGridSizes.ItemsSource = pipe.sizes;
    }
    else
    {
        choose = 4;
        if (db.sizes == null)
            db.sizes = new List<Size>();
        DataGridSizes.ItemsSource = db.sizes;

        ArticleCollumn.MinWidth = 0;
        ArticleCollumn.Width = 0;
    }

}
```

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    try
    {
        int index = DataGridSizes.SelectedIndex;
        string value = DataGridHelper.GetCell(DataGridSizes, index,
0).ToString().Substring(38);
        int id = Convert.ToInt32(value);
        Size size;
        if (choose == 1)
        {
            size = clutch.sizes.Where(s => s.id == id).First();
            clutch.sizes.Remove(size);
        }
        else if (choose == 2)
        {
            size = tap.sizes.Where(s => s.id == id).First();
            tap.sizes.Remove(size);
        }
        else if (choose == 3)
        {
            size = pipe.sizes.Where(s => s.id == id).First();
            pipe.sizes.Remove(size);
        }
        else if (choose == 4)
        {
            size = db.sizes.Where(s => s.id == id).First();
            db.sizes.Remove(size);
```

```
            }
            ToForm();
            core.SaveDB(db);
        }
        catch { }
    }

    private void Button_Click_1(object sender, RoutedEventArgs e)
    {

        if (choose == 1)
        {
            WindowSizeAdd wsa = new WindowSizeAdd();
            wsa.isNew = true;
            wsa.db = db;
            wsa.sizes = clutch.sizes;
            wsa.ShowDialog();
        }
        else if (choose == 2)
        {
            WindowSizeAdd wsa = new WindowSizeAdd();
            wsa.isNew = true;
            wsa.db = db;
            wsa.sizes = tap.sizes;
            wsa.ShowDialog();
        }
        else if (choose == 3)
        {
            WindowSizeAdd wsa = new WindowSizeAdd();
```

```
            wsa.isNew = true;
            wsa.db = db;
            wsa.sizes = pipe.sizes;
            wsa.ShowDialog();
        }
        else if(choose ==4)
        {
            WindowSize ws = new WindowSize();
            ws.sizes = db.sizes;
            ws.isNew = true;
            ws.ShowDialog();
            ToForm();
        }
        ToForm();
        core.SaveDB(db);
    }

    private void DataGridSizes_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
    {
        try
        {
            int index = DataGridSizes.SelectedIndex;
            string value = DataGridHelper.GetCell(DataGridSizes, index,
0).ToString().Substring(38);
            int id = Convert.ToInt32(value);
            WindowSize ws = new WindowSize();
            ws.isNew = false;
            if (choose == 1)
```

```
{
    WindowSizeAdd wsa = new WindowSizeAdd();
    wsa.isNew = false;
    wsa.db = db;
    wsa.sizes = clutch.sizes;
    wsa.size = clutch.sizes[index];
    wsa.ShowDialog();
}
else if (choose == 2)
{
    WindowSizeAdd wsa = new WindowSizeAdd();
    wsa.isNew = false;
    wsa.db = db;
    wsa.sizes = tap.sizes;
    wsa.size = tap.sizes[index];
    wsa.ShowDialog();
}
else if (choose == 3)
{
    WindowSizeAdd wsa = new WindowSizeAdd();
    wsa.isNew = false;
    wsa.db = db;
    wsa.sizes = pipe.sizes;
    wsa.size = pipe.sizes[index];
    wsa.ShowDialog();
}
else if (choose == 4)
{
    ws.sizes = db.sizes;
```

```
                ws.size = db.sizes.Where(s => s.id == id).First();
                ws.ShowDialog();
            }
            ToForm();
            core.SaveDB(db);
        }
        catch { }
    }

  }
}
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace PipeConstructor
{
  /// <summary>
  /// Interaction logic for WindowClutch.xaml
```

```
/// </summary>
public partial class WindowTap : Window
{
    public bool isNew;
    public DataBase db;
    public Core core = new Core();
    public Tap tap;
    public WindowTap()
    {
        InitializeComponent();
    }

    private void Grid_Loaded(object sender, RoutedEventArgs e)
    {
        if (isNew)
        {
            Do.Content = "Зберегти";
            tap = new Tap();
            if (db.taps.Count == 0)
                tap.id = 0;
            else
            {
                int max = db.taps[0].id + 1;
                for (int i = 1; i < db.taps.Count; i++)
                {
                    if (db.taps[i].id >= max)
                        max = db.taps[i].id + 1;
                }
                tap.id = max;
```

```
            }
        }
        else
        {
            Do.Content = "Редагувати";
            Name.Text = tap.name;
//              Size.Text = tap.size.ToString();
            Img.Source = core.StringToImage(tap.img);
        }
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        OpenFileDialog OFD = new OpenFileDialog();
        OFD.Filter = "png files (*.png)|*.png|jpg files (*.jpg)|*.jpg|All files (*.*)|*.*";
        if (OFD.ShowDialog() == true)
        {
            try
            {
                Img.Source = new BitmapImage(new Uri(OFD.FileName));
            }
            catch
            {
                MessageBox.Show("Невірний формат", "Помилка",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
```

```
private void Do_Click(object sender, RoutedEventArgs e)
{

    if (isNew)
    {

        try
        {
            tap.name = Name.Text;
            tap.img = core.ImageToString((BitmapImage)Img.Source);
            //                  tap.size = Convert.ToInt32(Size.Text);
            tap.width = 100;
            tap.height = 100;
        }
        catch { }
        db.taps.Add(tap);
        Close();
    }
    else
    {
        try
        {
            tap.name = Name.Text;
            tap.img = core.ImageToString((BitmapImage)Img.Source);
            //                  tap.size = Convert.ToInt32(Size.Text);
            tap.width = 100;
            tap.height = 100;
        }
        catch { }
```

```
            Close();
        }
    }

    private void Size_Click(object sender, RoutedEventArgs e)
    {
        WindowSizes ws = new WindowSizes();
        ws.db = db;
        ws.tap = tap;
        ws.ShowDialog();
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace PipeConstructor
{
  /// <summary>
```

```
/// Interaction logic for WindowClutches.xaml
/// </summary>
public partial class WindowTaps : Window
{
    public DataBase db;
    Core core = new Core();
    public WindowTaps()
    {
        InitializeComponent();
    }

    private void Grid_Loaded(object sender, RoutedEventArgs e)
    {
        ToForm();
    }
    public void ToForm()
    {
        DataGridTaps.ItemsSource = null;
        DataGridTaps.ItemsSource = db.taps;
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            int index = DataGridTaps.SelectedIndex;
            string value = DataGridHelper.GetCell(DataGridTaps, index,
0).ToString().Substring(38);
            int id = Convert.ToInt32(value);
```

```
            Tap tap = db.taps.Where(t => t.id == id).First();
            db.taps.Remove(tap);
            ToForm();
            core.SaveDB(db);
        }
        catch { }
    }

    private void Button_Click_1(object sender, RoutedEventArgs e)
    {
        WindowTap wt = new WindowTap();
        wt.isNew = true;
        wt.db = db;
        wt.ShowDialog();
        ToForm();
        core.SaveDB(db);
    }

    private void DataGridTaps_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
    {
        try
        {
            int index = DataGridTaps.SelectedIndex;
            string value = DataGridHelper.GetCell(DataGridTaps, index,
0).ToString().Substring(38);
            int id = Convert.ToInt32(value);
            WindowTap wt = new WindowTap();
            wt.isNew = false;
```

```
            wt.db = db;
            wt.tap = db.taps.Where(t => t.id == id).First();
            wt.ShowDialog();
            ToForm();
            core.SaveDB(db);
        }
        catch { }
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace PipeConstructor
{
  /// <summary>
  /// Interaction logic for WindowWallCreate.xaml
  /// </summary>
  public partial class WindowWallCreate : Window
```

```
    {
        public Room room;
        public WindowWallCreate()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            room.canCreate = true;
            room.name = RoomName.Text;
            Close();
        }
    }
}
```

ДОДАТОК В

Фрагмент коду бази данних

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using Newtonsoft.Json;
using static System.Net.Mime.MediaTypeNames;

namespace PipeConstructor
{
    public class Core
    {
        public DataBase LoadDB()
        {
            string path = Environment.CurrentDirectory + "\\data";
            string json = "";
            string[] SB;
            try
            {
                SB = File.ReadAllLines(path);
                DataBase tmpDB = new DataBase();
                for (int i = 0; i < SB.Length; i++)
                    json += SB[i];
                string decryptJson = StringCipher.Decrypt(json, "hsda%iw#5gd");
```

```
            DataBase db = JsonConvert.DeserializeObject<DataBase>(decryptJson);
            return db;
        }
        catch
        {
            DataBase db = new DataBase();
            return db;
        }
    }
    public void SaveDB(DataBase db)
    {
        string path = Environment.CurrentDirectory + "\\data";
        string json = JsonConvert.SerializeObject(db, Formatting.Indented);
        string encryptJson = StringCipher.Encrypt(json, "hsda%iw#5gd");
        File.WriteAllText(path, encryptJson);
    }


    //картинка в текст
    public string ImageToString(BitmapImage im)
    {
        if (im == null)
            return null;

        byte[] data;
        PngBitmapEncoder encoder = new PngBitmapEncoder();
        encoder.Frames.Add(BitmapFrame.Create(im));
        using (MemoryStream ms = new MemoryStream())
        {
```

```
            encoder.Save(ms);
            data = ms.ToArray();
        }
        return Convert.ToBase64String(data);

    }
    // текст в картинку
    public BitmapImage StringToImage(string imageString)
    {
        if (imageString == null)
            return null;
        byte[] array = Convert.FromBase64String(imageString);

        using (var ms = new System.IO.MemoryStream(array))
        {
            var image = new BitmapImage();
            image.BeginInit();
            image.CacheOption = BitmapCacheOption.OnLoad; // here
            image.StreamSource = ms;
            image.EndInit();
            return image;
        }
    }
    public Clutch copyClutch(Clutch clutch)
    {
        Clutch copy = new Clutch();
        copy.id = clutch.id;
        copy.img = clutch.img;
        copy.name = clutch.name;
```

```
            copy.size = clutch.size;
            copy.sizes = clutch.sizes;
            copy.height = clutch.height;
            copy.width = clutch.width;
            return copy;
        }
        public Tap copyTap(Tap tap)
        {
            Tap copy = new Tap();
            copy.id = tap.id;
            copy.img = tap.img;
            copy.name = tap.name;
            copy.size = tap.size;
            copy.sizes = tap.sizes;
            copy.height = tap.height;
            copy.width = tap.width;
            return copy;
        }
        public Pipe copyPipe(Pipe pipe)
        {
            Pipe copy = new Pipe();
            copy.id = pipe.id;
            copy.name = pipe.name;
            copy.size = pipe.size;
            copy.sizes = pipe.sizes;
            return copy;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace PipeConstructor
{

    public class DataBase
    {
        public DataBase()
        {
            clutches = new List<Clutch>();
            pipes = new List<Pipe>();
            taps = new List<Tap>();
            sizes = new List<Size>();
        }
        public List<Clutch> clutches { get; set; }
        public List<Pipe> pipes { get; set; }
        public List<Tap> taps { get; set; }
        public List<Size> sizes { get; set; }

    }
    public class Base
    {
        public int id { get; set; }
        public string name { get; set; }
        public int size { get; set; }
        public List<Size> sizes { get; set; }
    }
```

```
    public class BaseImage:Base
    {
        public string img { get; set; }
        public int width { get; set; }
        public int height { get; set; }
    }
    public class Clutch: BaseImage
    {

        public string carving { get; set; }
    }
    public class Tap : BaseImage
    {

    }
    public class Pipe:Base
    {

    }
    public class Size
    {
        public int id { get; set; }
        public string article { get; set; }
        public string size { get; set; }
    }

}
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
```

```
using System.Windows.Media;

namespace PipeConstructor
{
   static class DataGridHelper
   {
      static public DataGridCell GetCell(DataGrid dg, int row, int column)
      {
         DataGridRow rowContainer = GetRow(dg, row);

         if (rowContainer != null)
         {
            DataGridCellsPresenter presenter =
GetVisualChild<DataGridCellsPresenter>(rowContainer);

            // try to get the cell but it may possibly be virtualized
            DataGridCell cell =
(DataGridCell)presenter.ItemContainerGenerator.ContainerFromIndex(column);
            if (cell == null)
            {
               // now try to bring into view and retreive the cell
               dg.ScrollIntoView(rowContainer, dg.Columns[column]);
               cell =
(DataGridCell)presenter.ItemContainerGenerator.ContainerFromIndex(column);
            }
            return cell;
         }
         return null;
      }
```

```
        static public DataGridRow GetRow(DataGrid dg, int index)
        {
            DataGridRow row =
(DataGridRow)dg.ItemContainerGenerator.ContainerFromIndex(index);
            if (row == null)
            {
                // may be virtualized, bring into view and try again
                dg.ScrollIntoView(dg.Items[index]);
                row =
(DataGridRow)dg.ItemContainerGenerator.ContainerFromIndex(index);
            }
            return row;
        }

        static T GetVisualChild<T>(Visual parent) where T : Visual
        {
            T child = default(T);
            int numVisuals = VisualTreeHelper.GetChildrenCount(parent);
            for (int i = 0; i < numVisuals; i++)
            {
                Visual v = (Visual)VisualTreeHelper.GetChild(parent, i);
                child = v as T;
                if (child == null)
                {
                    child = GetVisualChild<T>(v);
                }
                if (child != null)
                {
```

```
                break;
            }
        }
        return child;
    }
  }
}
```

ДОДАТОК Г

Код програми-роботи з Excel

```
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows;
using System.Windows.Threading;
using Excel = Microsoft.Office.Interop.Excel;

namespace PipeConstructor
{
   public class ExcelReport
   {
      public static void CreateComercialProposalReport(DataBase db, string roomName,
List<Clutch> clutches, List<Tap> taps, List<Pipe> pipes,
System.Windows.Controls.ProgressBar pb)
      {
         Application.Current.Dispatcher.Invoke(
            DispatcherPriority.Background,
            new ThreadStart(delegate
            {
               pb.IsIndeterminate = true;
            }));
         //створення
```

```
        Excel.Application xlApp;
        Excel.Workbook xlWorkBook;
        Excel.Worksheet xlWorkSheet;
        object misValue = System.Reflection.Missing.Value;
        xlApp = new Excel.Application();
        string path = Path.Combine(Directory.GetCurrentDirectory(),
"Templates\\SpecificationTemplate.xls");
        xlWorkBook = xlApp.Workbooks.Open(path);
        xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
        Excel.Range _excelCells = (Excel.Range)xlWorkSheet.get_Range("A1",
"B1").Cells;

        xlWorkSheet.Cells[2, 2] = "Специфікація: №1 от " +
DateTime.Now.ToShortDateString();
        xlWorkSheet.Cells[4, 23] = roomName;

        int count = 0;
        List<HelperClutchCount> hcc = new List<HelperClutchCount>();
        for (int i = 0; i < clutches.Count; i++)
        {
            Clutch tmp = clutches[i];
            tmp.sizes = db.clutches.Where(c => c.id == clutches[i].id).First().sizes;
            string article;
            string size = "-";
            if (clutches[i].size == -1)
                article = "невідомо";
            else
            {
                article = tmp.sizes.Where(s => s.id == tmp.size).First().article;
```

```
            size = tmp.sizes.Where(s => s.id == tmp.size).First().size;
        }

        if (hcc.Where(h => h.id == clutches[i].id && h.article == article).ToList().Count == 0) // якщо є
            hcc.Add(new HelperClutchCount() { id = clutches[i].id, name = clutches[i].name, article = article, count = 1, size = size  }); // створити з кількістю 1
        else //інакше
            hcc.Where(h => h.id == clutches[i].id && h.article == article).First().count++; // збільшити на 1
    }
    List<HelperTapCount> htc = new List<HelperTapCount>();
    for (int i = 0; i < taps.Count; i++)
    {
        Tap tmp = taps[i];
        tmp.sizes = db.taps.Where(c => c.id == taps[i].id).First().sizes;
        string article;
        string size = "-";
        if (taps[i].size == -1)
            article = "невідомо";
        else
        {
            article = tmp.sizes.Where(s => s.id == tmp.size).First().article;
            size = tmp.sizes.Where(s => s.id == tmp.size).First().size;
        }

        if (htc.Where(h => h.id == taps[i].id && h.article == article).ToList().Count == 0)
```

```
            htc.Add(new HelperTapCount() { id = taps[i].id, name = taps[i].name,
article = article, count = 1, size = size });
            else
                htc.Where(h => h.id == taps[i].id && h.article == article).First().count++;
        }
        List<HelperPipeCount> hpc = new List<HelperPipeCount>();
        for (int i = 0; i < pipes.Count; i++)
        {
            Pipe tmp = pipes[i];
                tmp.sizes = db.pipes.Where(c => c.id == pipes[i].id).First().sizes;
            string article;
            string size = "-";
            if (pipes[i].size == -1)
                article = "невідомо";
            else
            {
                article = tmp.sizes.Where(s => s.id == tmp.size).First().article;
                size = tmp.sizes.Where(s => s.id == tmp.size).First().size;
            }

            if (hpc.Where(h => h.id == pipes[i].id && h.article == article).ToList().Count
== 0)
                hpc.Add(new HelperPipeCount() { id = pipes[i].id, name = pipes[i].name,
article = article, count = 1, size = size });
            else
                hpc.Where(h => h.id == pipes[i].id && h.article == article).First().count++;
        }

        int currRowIndex = 9;
```

```
        for (int i = 0; i < hcc.Count; i++)
        {
            _excelCells = (Excel.Range)xlWorkSheet.get_Range("B" + currRowIndex,
"C" + currRowIndex).Cells;
            _excelCells.Merge(Type.Missing);
            _excelCells = (Excel.Range)xlWorkSheet.get_Range("E" + currRowIndex,
"AA" + currRowIndex).Cells;
            _excelCells.Merge(Type.Missing);
            _excelCells = (Excel.Range)xlWorkSheet.get_Range("AC" + currRowIndex,
"AG" + currRowIndex).Cells;
            _excelCells.Merge(Type.Missing);

            _excelCells.RowHeight = 10;

            xlWorkSheet.Cells[currRowIndex, 2] = currRowIndex - 8;
            xlWorkSheet.Cells[currRowIndex, 4] = hcc[i].article;
            xlWorkSheet.Cells[currRowIndex, 5] = hcc[i].name;
            xlWorkSheet.Cells[currRowIndex, 28] = hcc[i].size;
            xlWorkSheet.Cells[currRowIndex, 29] = hcc[i].count.ToString() + " шт";

            count += hcc[i].count;

            currRowIndex++;

             Application.Current.Dispatcher.Invoke(
             DispatcherPriority.Background,
             new ThreadStart(delegate
             {
                pb.Tag = "Обробка фітінгів " + (i + 1).ToString() + " из " + hcc.Count;
```

```
            }));
        }
        for (int i = 0; i < htc.Count; i++)
        {
            _excelCells = (Excel.Range)xlWorkSheet.get_Range("B" + currRowIndex,
"C" + currRowIndex).Cells;
            _excelCells.Merge(Type.Missing);
            _excelCells = (Excel.Range)xlWorkSheet.get_Range("E" + currRowIndex,
"AA" + currRowIndex).Cells;
            _excelCells.Merge(Type.Missing);
            _excelCells = (Excel.Range)xlWorkSheet.get_Range("AC" + currRowIndex,
"AG" + currRowIndex).Cells;
            _excelCells.Merge(Type.Missing);

            _excelCells.RowHeight = 10;

            xlWorkSheet.Cells[currRowIndex, 2] = currRowIndex - 8;
            xlWorkSheet.Cells[currRowIndex, 4] = htc[i].article;
            xlWorkSheet.Cells[currRowIndex, 5] = htc[i].name;
            xlWorkSheet.Cells[currRowIndex, 28] = htc[i].size;
            xlWorkSheet.Cells[currRowIndex, 29] = htc[i].count.ToString() + " шт";


            count += htc[i].count;

            currRowIndex++;

            Application.Current.Dispatcher.Invoke(
             DispatcherPriority.Background,
```

```
            new ThreadStart(delegate
            {
                pb.Tag = "Обробка кранів " + (i + 1).ToString() + " из " + htc.Count;
            }));
        }
        for (int i = 0; i < hpc.Count; i++)
        {
            _excelCells = (Excel.Range)xlWorkSheet.get_Range("B" + currRowIndex,
"C" + currRowIndex).Cells;
            _excelCells.Merge(Type.Missing);
            _excelCells = (Excel.Range)xlWorkSheet.get_Range("E" + currRowIndex,
"AA" + currRowIndex).Cells;
            _excelCells.Merge(Type.Missing);
            _excelCells = (Excel.Range)xlWorkSheet.get_Range("AC" + currRowIndex,
"AG" + currRowIndex).Cells;
            _excelCells.Merge(Type.Missing);

            _excelCells.RowHeight = 10;

            xlWorkSheet.Cells[currRowIndex, 2] = currRowIndex - 8;
            xlWorkSheet.Cells[currRowIndex, 4] = hpc[i].article;
            xlWorkSheet.Cells[currRowIndex, 5] = hpc[i].name;
            xlWorkSheet.Cells[currRowIndex, 28] = hpc[i].size;
            xlWorkSheet.Cells[currRowIndex, 29] = hpc[i].count.ToString() + " шт";


            count += hpc[i].count;

            currRowIndex++;
```

```
            Application.Current.Dispatcher.Invoke(
             DispatcherPriority.Background,
             new ThreadStart(delegate
             {
                pb.Tag = "Обробка труб " + (i + 1).ToString() + " из " + hpc.Count;
             }));
        }


             Application.Current.Dispatcher.Invoke(
             DispatcherPriority.Background,
             new ThreadStart(delegate
             {
                pb.Tag = "Оформлення...";
             }));

        _excelCells = (Excel.Range)xlWorkSheet.get_Range("B9", "AG" +
(currRowIndex-1).ToString()).Cells;
        _excelCells.Cells.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;
        _excelCells.WrapText = true;

        _excelCells = (Excel.Range)xlWorkSheet.get_Range("D9", "AA" +
(currRowIndex - 1).ToString()).Cells;
        _excelCells.HorizontalAlignment = Excel.XlHAlign.xlHAlignLeft;
        _excelCells = (Excel.Range)xlWorkSheet.get_Range("AB9", "AB" +
(currRowIndex - 1).ToString()).Cells;
        _excelCells.HorizontalAlignment = Excel.XlHAlign.xlHAlignRight;
```

```
        _excelCells = (Excel.Range)xlWorkSheet.get_Range("AC9", "AG" +
(currRowIndex - 1).ToString()).Cells;
        _excelCells.HorizontalAlignment = Excel.XlHAlign.xlHAlignLeft;

        //подвал
        currRowIndex++;

        _excelCells = (Excel.Range)xlWorkSheet.get_Range("X" + currRowIndex, "Z"
+ currRowIndex).Cells;
        _excelCells.Merge(Type.Missing);
        _excelCells.RowHeight = 10;
        xlWorkSheet.Cells[currRowIndex, 24] = "Разом:";
        xlWorkSheet.Cells[currRowIndex, 29] = count.ToString();
        xlWorkSheet.Cells[currRowIndex, 30] = "шт";

        _excelCells.Font.Size = 9;
        _excelCells.Font.Bold = true;

        currRowIndex += 2;

        _excelCells = (Excel.Range)xlWorkSheet.get_Range("B" + currRowIndex,
"AE" + currRowIndex).Cells;
        _excelCells.Merge(Type.Missing);
        _excelCells.RowHeight = 12;

        xlWorkSheet.Cells[currRowIndex, 2] = "Всього найменувань " + (hcc.Count +
htc.Count + hpc.Count).ToString();

        currRowIndex += 2;
```

```
        _excelCells = (Excel.Range)xlWorkSheet.get_Range("B" + currRowIndex,
"AE" + currRowIndex).Cells;

        _excelCells.Borders[Excel.XlBordersIndex.xlEdgeTop].LineStyle =
Excel.XlLineStyle.xlContinuous;
        _excelCells.Borders[Excel.XlBordersIndex.xlEdgeTop].Weight =
Excel.XlBorderWeight.xlThick;

        Application.Current.Dispatcher.Invoke(
            DispatcherPriority.Background,
            new ThreadStart(delegate
            {
                pb.Tag = "Збереження";
                //збереження
                SaveFileDialog savefile = new SaveFileDialog();
                savefile.FileName = "Специфікація на "+ roomName + " від " +
DateTime.Now.ToShortDateString() + ".xlsx";
                savefile.Filter = "Excel files (*.xlsx)|*.xlsx|All files (*.*)|*.*";

                if (savefile.ShowDialog() == true)
                {
                    if (File.Exists(savefile.FileName)) File.Delete(savefile.FileName);
                    xlWorkBook.SaveAs(savefile.FileName,
Excel.XlFileFormat.xlWorkbookDefault, misValue, misValue, misValue, misValue,
Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue, misValue, misValue,
misValue);
                    xlWorkBook.Close(true, misValue, misValue);
                    xlApp.Quit();
```

```
                }

                releaseObject(xlWorkSheet);
                releaseObject(xlWorkBook);
                releaseObject(xlApp);

            }));
        }
        private static void releaseObject(object obj)
        {
            try
            {
                System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
                obj = null;
            }
            catch (Exception ex)
            {
                obj = null;
            }
            finally
            {
                GC.Collect();
            }
        }

    }
}
```

ДОДАТОК Д

Код програми – взаємодії об’єктів

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Controls;
using System.Windows.Shapes;

namespace PipeConstructor
{
    public class HelperClutchCount
    {
        public int id;
        public string name;
        public int count;
        public string article;
        public string size;
    }
    public class HelperTapCount
    {
        public int id;
        public string name;
        public int count;
        public string article;
        public string size;
    }
    public class HelperPipeCount
    {
```

```
        public int id;
        public string name;
        public int count;
        public string article;
        public string size;
    }
    public class HelpWorkSpase
    {
        public Image image;
        public Border border;
        public object o;
        public byte lvl;
        public byte orientation; // 1-0, 2-90, 3-180, 4-270
        public bool mirror; // true - зеркально, false - нормально
    }
    public class HelpWorkSpaseLine
    {
        public Line line;
        public Border border;
        public object o;
        public byte lvl;

    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace PipeConstructor
{
    public class Room
    {

        public string name;
        public bool canCreate = false;
        public List<HelpWorkSpase> listhws = new List<HelpWorkSpase>();
        public List<HelpWorkSpaseLine> listhwsLine = new
List<HelpWorkSpaseLine>();
        public List<Clutch> clutches = new List<Clutch>();
        public List<Tap> taps = new List<Tap>();
        public List<Pipe> pipes = new List<Pipe>();
        public Room()
        {
            listhws = new List<HelpWorkSpase>();
            listhwsLine = new List<HelpWorkSpaseLine>();
            clutches = new List<Clutch>();
            taps = new List<Tap>();
            pipes = new List<Pipe>();
        }
    }
    public class RoomForFile
    {
        Core core = new Core();
        public string name { get; set; }
        public List<ClutchForFile> clutches { get; set; }
        public List<TapForFile> taps { get; set; }
        public List<PipeForFile> pipes { get; set; }
```

```
public List<double> wallLen { get; set; }
public RoomForFile()
{
    clutches = new List<ClutchForFile>();
    taps = new List<TapForFile>();
    pipes = new List<PipeForFile>();
}
public Room ToRoom(DataBase db)
{
    Room room = new Room();
    room.clutches = new List<Clutch>();
    room.taps = new List<Tap>();
    room.pipes = new List<Pipe>();
    room.listhws = new List<HelpWorkSpase>();
    room.listhwsLine = new List<HelpWorkSpaseLine>();
    //логика создания класа для работы с класа для сериализации
    room.name = name;
    for (int i = 0; i < clutches.Count; i++)
    {
        Clutch c = db.clutches.Where(cl=>cl.id== clutches[i].id).First();
        c.size = clutches[i].size;
        room.clutches.Add(core.copyClutch(c));
    }
    for (int i = 0; i < taps.Count; i++)
    {
        Tap t = db.taps.Where(tp => tp.id == taps[i].id).First();
        t.size = taps[i].size;
        room.taps.Add(core.copyTap(t));
    }
```

```
    for (int i = 0; i < pipes.Count; i++)
    {
        Pipe p = db.pipes.Where(pipe => pipe.id == pipes[i].id).First();
        p.size = pipes[i].size;
        room.pipes.Add(core.copyPipe(p));
    }

    return room;
}
public RoomForFile(Room room)
{
    name = room.name;
    clutches = new List<ClutchForFile>();
    taps = new List<TapForFile>();
    pipes = new List<PipeForFile>();
    wallLen = new List<double>();

    int id;
    int size;
    byte orientation;
    double left;
    double top;
    byte lvl;
    bool mirror;
    //логика создания класа для сериализации
    for (int i = 0; i < room.listhws.Count; i++)
    {
        string[] s = room.listhws[i].border.Tag.ToString().Split(' ');
```

```
if (s[0] == "1") //фитинги
{
    id = Convert.ToInt32(s[1]);
    size = Convert.ToInt32(s[2]);
    orientation = room.listhws[i].orientation;
    left = room.listhws[i].border.Margin.Left;
    top = room.listhws[i].border.Margin.Top;
    lvl = room.listhws[i].lvl;
    mirror = room.listhws[i].mirror;
    clutches.Add
        (
        new ClutchForFile()
        {
            id = id,
            size = size,
            left = left,
            top = top,
            orientation = orientation,
            lvl = lvl,
            mirror = mirror
        }
        );
}
else if (s[0] == "2")//краны
{
    id = Convert.ToInt32(s[1]);
    size = Convert.ToInt32(s[2]);
    orientation = room.listhws[i].orientation;
    left = room.listhws[i].border.Margin.Left;
```

```
            top = room.listhws[i].border.Margin.Top;
            lvl = room.listhws[i].lvl;
            mirror = room.listhws[i].mirror;

            taps.Add
            (
            new TapForFile()
            {
                id = id,
                size = size,
                left = left,
                top = top,
                orientation = orientation,
                lvl = lvl,
                mirror = mirror
            }
            );
        }
    }
    for (int i = 0; i < room.listhwsLine.Count; i++)
    {
        string[] s = room.listhwsLine[i].border.Tag.ToString().Split(' ');

        if (s[0] == "3")//трубы
        {
            id = Convert.ToInt32(s[1]);
            size = Convert.ToInt32(s[2]);
            left = room.listhwsLine[i].border.DesiredSize.Width;
            top = room.listhwsLine[i].border.DesiredSize.Height;
```

```
bool HorizontalOrVertical;
double len;
lvl = room.listhwsLine[i].lvl;
if (room.listhwsLine[i].line.Y1 == room.listhwsLine[i].line.Y2) //труба горизонтальная
{
   HorizontalOrVertical = true;
   len = Math.Abs(room.listhwsLine[i].line.X1 - room.listhwsLine[i].line.X2);
   left-=len;
}
else
{
   HorizontalOrVertical = false;
   len = Math.Abs(room.listhwsLine[i].line.Y1 - room.listhwsLine[i].line.Y2);
   top -= len;
}
pipes.Add
(
new PipeForFile()
{
   id = id,
   size = size,
   left = left,
   top = top,
   HorizontalOrVertical = HorizontalOrVertical,
   len = len,
   lvl = lvl
```

```
                }
                );
            }
        }

    }
}
public class ClutchForFile
{
    public int id { get; set; }
    public int size { get; set; }
    public double left { get; set; }
    public double top { get; set; }
    public byte lvl { get; set; }
    public byte orientation { get; set; } // 1-0, 2-90, 3-180, 4-270
    public bool mirror { get; set; }// true - зеркально, false - нормально

}
public class TapForFile
{
    public int id { get; set; }
    public int size { get; set; }
    public double left { get; set; }
    public double top { get; set; }
    public byte lvl { get; set; }
    public byte orientation { get; set; } // 1-0, 2-90, 3-180, 4-270
    public bool mirror { get; set; }// true - зеркально, false - нормально

}
```

```
    public class PipeForFile
    {
        public int id { get; set; }
        public int size { get; set; }
        public double left { get; set; }
        public double top { get; set; }
        public byte lvl { get; set; }
        public bool HorizontalOrVertical { get; set; } // true - горизонтально, false - вертикальна
        public double len { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;

namespace PipeConstructor
{
    public class StickyPoint
    {
        public double X;
        public double Y;
        public byte orientation; //1 -лево, 3 - право, 2 - верх, 4 - низ
    }
}
using System;
using System.Collections.Generic;
```

```
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;

namespace PipeConstructor
{
   public static class StringCipher
   {
      // This constant is used to determine the keysize of the encryption algorithm in bits.
      // We divide this by 8 within the code below to get the equivalent number of bytes.
      private const int Keysize = 256;

      // This constant determines the number of iterations for the password bytes generation function.
      private const int DerivationIterations = 1000;

      public static string Encrypt(string plainText, string passPhrase)
      {
         // Salt and IV is randomly generated each time, but is preprended to encrypted cipher text
         // so that the same Salt and IV values can be used when decrypting.
         var saltStringBytes = Generate256BitsOfRandomEntropy();
         var ivStringBytes = Generate256BitsOfRandomEntropy();
         var plainTextBytes = Encoding.UTF8.GetBytes(plainText);
         using (var password = new Rfc2898DeriveBytes(passPhrase, saltStringBytes, DerivationIterations))
         {
            var keyBytes = password.GetBytes(Keysize / 8);
```

```
            using (var symmetricKey = new RijndaelManaged())
            {
               symmetricKey.BlockSize = 256;
               symmetricKey.Mode = CipherMode.CBC;
               symmetricKey.Padding = PaddingMode.PKCS7;
               using (var encryptor = symmetricKey.CreateEncryptor(keyBytes,
ivStringBytes))
               {
                  using (var memoryStream = new MemoryStream())
                  {
                     using (var cryptoStream = new CryptoStream(memoryStream,
encryptor, CryptoStreamMode.Write))
                     {
                        cryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length);
                        cryptoStream.FlushFinalBlock();
                        // Create the final bytes as a concatenation of the random salt bytes,
the random iv bytes and the cipher bytes.
                        var cipherTextBytes = saltStringBytes;
                        cipherTextBytes                                              =
cipherTextBytes.Concat(ivStringBytes).ToArray();
                        cipherTextBytes                                              =
cipherTextBytes.Concat(memoryStream.ToArray()).ToArray();
                        memoryStream.Close();
                        cryptoStream.Close();
                        return Convert.ToBase64String(cipherTextBytes);
                     }
                  }
               }
            }
```

```
        }
    }

    public static string Decrypt(string cipherText, string passPhrase)
    {
        // Get the complete stream of bytes that represent:
        // [32 bytes of Salt] + [32 bytes of IV] + [n bytes of CipherText]
        var cipherTextBytesWithSaltAndIv = Convert.FromBase64String(cipherText);
        // Get the saltbytes by extracting the first 32 bytes from the supplied cipherText bytes.
        var saltStringBytes = cipherTextBytesWithSaltAndIv.Take(Keysize / 8).ToArray();
        // Get the IV bytes by extracting the next 32 bytes from the supplied cipherText bytes.
        var ivStringBytes = cipherTextBytesWithSaltAndIv.Skip(Keysize / 8).Take(Keysize / 8).ToArray();
        // Get the actual cipher text bytes by removing the first 64 bytes from the cipherText string.
        var cipherTextBytes = cipherTextBytesWithSaltAndIv.Skip((Keysize / 8) * 2).Take(cipherTextBytesWithSaltAndIv.Length - ((Keysize / 8) * 2)).ToArray();

        using (var password = new Rfc2898DeriveBytes(passPhrase, saltStringBytes, DerivationIterations))
        {
            var keyBytes = password.GetBytes(Keysize / 8);
            using (var symmetricKey = new RijndaelManaged())
            {
                symmetricKey.BlockSize = 256;
                symmetricKey.Mode = CipherMode.CBC;
```

```
            symmetricKey.Padding = PaddingMode.PKCS7;
            using (var decryptor = symmetricKey.CreateDecryptor(keyBytes, ivStringBytes))
            {
               using (var memoryStream = new MemoryStream(cipherTextBytes))
               {
                  using (var cryptoStream = new CryptoStream(memoryStream, decryptor, CryptoStreamMode.Read))
                  {
                     var plainTextBytes = new byte[cipherTextBytes.Length];
                     var decryptedByteCount = cryptoStream.Read(plainTextBytes, 0, plainTextBytes.Length);
                     memoryStream.Close();
                     cryptoStream.Close();
                     return Encoding.UTF8.GetString(plainTextBytes, 0, decryptedByteCount);
                  }
               }
            }
         }
      }

      private static byte[] Generate256BitsOfRandomEntropy()
      {
         var randomBytes = new byte[32]; // 32 Bytes will give us 256 bits.
         using (var rngCsp = new RNGCryptoServiceProvider())
         {
            // Fill the array with cryptographically secure random bytes.
```

```
            rngCsp.GetBytes(randomBytes);
        }
        return randomBytes;
    }
  }
}
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
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;

namespace PipeConstructor
{
  /// <summary>
  /// Interaction logic for WindowClutch.xaml
  /// </summary>
  public partial class WindowClutch : Window
  {
    public bool isNew;
```

```
public DataBase db;
public Core core = new Core();
public Clutch clutch;
ObservableCollection<string> ImgLen = new ObservableCollection<string>();
public WindowClutch()
{
    InitializeComponent();
}



private void Grid_Loaded(object sender, RoutedEventArgs e)
{
    ImgLen.Add("50");
    ImgLen.Add("100");
    ImgLen.Add("200");
    ImgLen.Add("300");

    ImgHeight.ItemsSource = ImgLen;
    ImgWidth.ItemsSource = ImgLen;

    ImgHeight.SelectedIndex = 1;
    ImgWidth.SelectedIndex = 1;
    if (isNew)
    {
        Do.Content = "Зберегти";
        clutch = new Clutch();
        if (db.clutches.Count == 0)
            clutch.id = 0;
        else
```

```
        {
            int max = db.clutches[0].id + 1;
            for (int i = 1; i < db.clutches.Count; i++)
            {
                if (db.clutches[i].id >= max)
                    max = db.clutches[i].id + 1;
            }
            clutch.id = max;
        }
    }
    else
    {
        ImgHeight.SelectedIndex = clutch.height / 100;
        ImgWidth.SelectedIndex = clutch.width / 100;
        Do.Content = "Редагувати";
        Name.Text = clutch.name;
        //              Size.Text = clutch.size.ToString();
        Carving.Text = clutch.carving;
        Img.Source = core.StringToImage(clutch.img);
    }
}

private void Button_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog OFD = new OpenFileDialog();
    OFD.Filter = "png files (*.png)|*.png|jpg files (*.jpg)|*.jpg|All files (*.*)|*.*";
    if (OFD.ShowDialog() == true)
    {
        try
```

```
            {
                Img.Source = new BitmapImage(new Uri(OFD.FileName));
            }
            catch
            {
                MessageBox.Show("Невірний формат", "Помилка",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }

    private void Do_Click(object sender, RoutedEventArgs e)
    {

        if (isNew)
        {
            try
            {
                clutch.name = Name.Text;
                clutch.img = core.ImageToString((BitmapImage)Img.Source);
                //                clutch.size = Convert.ToInt32(Size.Text);
                clutch.carving = Carving.Text;
                clutch.width = (int)Img.Width;
                clutch.height = (int)Img.Height;
            }
            catch { }
            db.clutches.Add(clutch);
            Close();
        }
```

```
        else
        {
            try
            {
                clutch.name = Name.Text;
                clutch.img = core.ImageToString((BitmapImage)Img.Source);
                //                  clutch.size = Convert.ToInt32(Size.Text);
                clutch.carving = Carving.Text;
                clutch.width = (int)Img.Width;
                clutch.height = (int)Img.Height;
            }
            catch { }
            Close();
        }
    }

    private void Sizes_Click(object sender, RoutedEventArgs e)
    {
        WindowSizes ws = new WindowSizes();
        ws.db = db;
        ws.clutch = clutch;
        ws.ShowDialog();

    }

    private void ImgWidth_SelectionChanged(object sender,
SelectionChangedEventArgs e)
    {
        try
```

```
            {
                Img.Height = Convert.ToDouble(ImgLen[ImgHeight.SelectedIndex]);
                Img.Width = Convert.ToDouble(ImgLen[ImgWidth.SelectedIndex]);
                Border.Height = Img.Height + 4;
                Border.Width = Img.Width + 4;
            }
            catch { }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace PipeConstructor
{
    /// <summary>
    /// Interaction logic for WindowClutches.xaml
    /// </summary>
    public partial class WindowClutches : Window
```

```
{
    public DataBase db;
    Core core = new Core();
    public WindowClutches()
    {
        InitializeComponent();
    }

    private void Grid_Loaded(object sender, RoutedEventArgs e)
    {
        ToForm();
    }
    public void ToForm()
    {
        DataGridClutches.ItemsSource = null;
        DataGridClutches.ItemsSource = db.clutches;
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            int index = DataGridClutches.SelectedIndex;
            string value = DataGridHelper.GetCell(DataGridClutches, index,
0).ToString().Substring(38);
            int id = Convert.ToInt32(value);
            Clutch clutch = db.clutches.Where(cl => cl.id == id).First();
            db.clutches.Remove(clutch);
            ToForm();
```

```
            core.SaveDB(db);
        }
        catch { }
    }

    private void Button_Click_1(object sender, RoutedEventArgs e)
    {
        WindowClutch wc = new WindowClutch();
        wc.isNew = true;
        wc.db = db;
        wc.ShowDialog();
        ToForm();
        core.SaveDB(db);
    }

    private void DataGridClutches_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
    {
        try
        {
            int index = DataGridClutches.SelectedIndex;
            string value = DataGridHelper.GetCell(DataGridClutches, index,
0).ToString().Substring(38);
            int id = Convert.ToInt32(value);
            WindowClutch wc = new WindowClutch();
            wc.isNew = false;
            wc.db = db;
            wc.clutch = db.clutches.Where(cl => cl.id == id).First();
            wc.ShowDialog();
```

```
            ToForm();
            core.SaveDB(db);
        }
        catch { }
    }
  }
}
```

ДОДАТОК Е

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:____Програмне забезпечення для проектування системи опалення будинку_____

___

Тип роботи:_____ бакалаврський дипломний проект _____

_____

__

(кваліфікаційна роботи, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ _____кафедра обчислювальної техніки_____

(кафедра, факультет (інститут), навчальна група)

Науковий керівник _____Черняк О. І., доцент кафедри ОТ_____

(прізвище, ініціали, посада)

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	77,8%
Тривога/Білі знаки	/	Схожість	22,2%

Аналіз звіту подібності (відмінити подібне)

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності і відсутності самостійності її автора. Робот направити на доопрацювання.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ _____

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

__ Ступінь оригінальності роботи відповідає вимогам, що висуваються до БР

_____

_____

Особа, відповідальна за перевірку_____ _Захарченко С.М.__

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали)