Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка інформаційної системи для обліку та контролю температури зерна на елеваторах»

Виконав: студент групи К21-1

Спеціальність 122 «Комп'ютерні науки» Просяник Ю. О.

(прізвище та ініціали)

Керівник к.т.н., доц. Ульяновська Ю. В.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент

(місце роботи)

(посада)

(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро 2025

АНОТАЦІЯ

Просяник Ю.О. Розробка інформаційної системи для обліку та контролю температури зерна на елеваторах.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 122 «Комп'ютерні наук». – Університет митної справи та фінансів, Дніпро, 2025.

Об'єктом дослідження є процес автоматизації управління термометрією на елеваторі.

Предмет дослідження – програмне забезпечення для вимірювання температури зерна на елеваторі.

Метою роботи є автоматизація моніторингу температури зерна на елеваторах.

Дана робота присвячена розробці програмного забезпечення для підвищення ефективності управління системи термометрії елеватора, що відбувається за рахунок алгоритмічної оптимізації задач вимірювання температури при зберіганні зерна в складах, елеваторах, силосах, а також розробленого алгоритму, оперативного вимірювання температури зерна за комплексним критерієм як складовою програмного забезпечення «Термометрія елеватора».

У процесі роботи проведено аналіз методів централізованої термометрії. Практична цінність роботи полягає у впровадженні інформаційних технологій у роботу зерносховищ і є критично важливою для сучасного аграрного сектору України. Дана робота сприятиме цифровій трансформації зернової галузі України, забезпечуючи сталість якості продукції, економію ресурсів та зростання експортного потенціалу.

Ключові слова: програмне забезпечення, термометрія, класифікація, C#, SQL, алгоритм, проектування.

ABSTRACT

Prosyanyk Y.O. Development of an information system for accounting and controlling grain temperature at elevators.

Qualification work for obtaining a bachelor's degree in specialty 122 "Computer Science". – University of Customs and Finance, Dnipro, 2025.

The object of the study is the process of automating the management of thermometry at the elevator.

The subject of the study is software for measuring the temperature of grain at the elevator.

The purpose of the work is to automate the monitoring of grain temperature at elevators.

This work is devoted to the development of software to improve the efficiency of the elevator thermometry system management, which occurs through algorithmic optimization of temperature measurement tasks during grain storage in warehouses, elevators, silos, as well as the developed algorithm for operational measurement of grain temperature by a complex criterion as a component of the “Elevator Thermometry” software.

In the process of work, an analysis of centralized thermometry methods was conducted. The practical value of the work lies in the introduction of information technologies into the operation of granaries and is critically important for the modern agricultural sector of Ukraine. This work will contribute to the digital transformation of the grain industry of Ukraine, ensuring the stability of product quality, saving resources and increasing export potential.

Keywords: software, thermometry, classification, C#, SQL, algorithm, design.

ЗМІСТ

ВСТУП

Сучасний аграрний сектор України, зокрема зернова галузь, відіграє ключову роль у забезпеченні продовольчої безпеки та формуванні експортного потенціалу країни. Однак ефективність виробництва та зберігання зерна обмежується низкою проблем, серед яких:

- застосування застарілих технологій;
- ручний контроль температурного режиму у зерносховищах;
- високі енерговитрати та ризики втрат продукції через самозайманння або псування.

Процес виробництва зернових культур характеризується чітко вираженою сезонністю, внаслідок чого значні обсяги зерна концентруються упродовж обмеженого часово періоду, що триває кілька діб. Створення великих запасів зерна напряму пов'язано з управлінням вимірювання температури зерна в сховищах. Існуючі підходи до управління, незалежно від того, старі чи це релейні системи, або сучасні з використанням промислових контролерів, використовують ручну оперативну настройку вимірювання температури зерна. Такий підхід затрудняє оперативний урахування стану механізмів елеватора за факторами зайнятості та справності, а також затрудняє визначення вартості транспортування за споживанням енергії маршрутом. Рішення зазначеної задачі вручну силами оператора або технології вносить в управління людський фактор, що приводить до небажаних простоїв і додаткових витрат електроенергії, що, у свою чергу, підвищує вартість зберігання зерна. Таким чином, актуальна задача побудови алгоритму вимірювання температури зерна при його зберіганні за комплексним критерієм справності, зайнятості та енергоспоживання вузлів транспортно-технологічної мережі елеватора [1].

Метою даної роботи є автоматизація моніторингу температури зерна на елеваторах, що дозволить:

- Оптимізувати витрати на енергію та обслуговування.

- Зменшити вплив людського фактора на процес контролю.
- Запобігти втратам зерна через перегрів або підвищену вологість.

Підвищення ефективності управління системи термометрії елеватора відбувається за рахунок алгоритмічної оптимізації задач вимірювання температури при зберіганні зерна в складах, елеваторах, силосах, а також розробленого алгоритму, оперативного вимірювання температури зерна за комплексним критерієм як складовою програмного забезпечення «Термометрія елеватора».

Об'єктом дослідження є процес автоматизації управління термометрією на елеваторі.

Предмет дослідження – програмне забезпечення для вимірювання температури зерна на елеваторі.

При рішенні поставлених завдань виконано аналіз літературних джерел за вихідними посиланнями досліджень, для оптимальної роботи системи термометрії зерна на елеваторі були досліджені існуючі методи вимірювання температури.

В процесі дослідження виконується аналіз методів централізованої термометрії.

Дана робота є актуальною бо впровадження інформаційних технологій у роботу зерносховища є критично важливим для:

- підвищення конкурентоспроможності українського аграрного сектору;
- відповідності міжнародним стандартам якості зберігання зерна;
- скорочення витрат на логістику та енергоресурси.

Запропоноване рішення відрізняється:

- комплексним підходом до управління термометрією з урахуванням справності обладнання, зайнятості вузлів елеватора та енергоефективності;

- інтеграцією з інтернетом речей, автоматизованими системами контролю та моніторингу та системами управління агропідприємствами для автоматизації збору даних та аналітики;
- використанням сучасних технологій (LoRaWAN, Python, React) для масштабованості та надійності;
- сучасним рівнем розв'язання проблеми, тому що існуючі системи термометрії часто:
  - вимагають ручного втручання, що збільшує витрати;
  - не мають ефективних механізмів прогнозування аномалій;
  - обмежені у можливостях інтеграції з іншими системами управління.

Очікуваний ефект від впровадження розробленого програмного забезпечення дозволить:

– знизити втрати зерна на 15–20% за рахунок оперативного виявлення перегріву;

– скоротити енерговитрати на 10–15% через оптимізацію роботи вентиляції;

– підвищити точність контролю за рахунок автоматизованої обробки даних.

Дана робота сприятиме цифровій трансформації зернової галузі України, забезпечуючи сталість якості продукції, економію ресурсів та зростання експортного потенціалу.

Робота складається зі вступу, трьох розділів, висновків, 19 використаних джерел та 1 додатку. Обсяг роботи становить 55 сторінок основного тексту, 20 рисунків.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Значення зернової галузі та її стан

Однією з найбільш важливих стратегічних галузей економіки України є зерновий сектор, який багато в чому визначає вартість продовольчих товарів у державі.

На жаль, матеріально-технічне забезпечення та ефективність роботи зернового сектора економіки не відповідає світовим стандартам і значно відстає від реальних потреб галузі.

Неналежне фінансування стримує впровадження новіших технологій, обмежує застосування інших ресурсів. На виробництво зерна істотно впливають погодні чинники. Через часте нецільове використання полів, вони засмічуються бур'янами. В результаті в процесі збору на елеватор може поступати пошкоджене зерно, що істотно відрізняється за своїми якісними показниками (вологість, засміченість). В той же час для забезпечення збереження зерна його доцільно сортувати. Сортують зерно за результатами лабораторного аналізу.

Зернове виробництво виступає фундаментальною складовою агропромислового комплексу та є ключовим чинником стабільного розвитку його основних секторів, а також формує основу аграрного експорту. У зв’язку з цим постає потреба в оперативному формуванні значних обсягів зернових запасів. Це, у свою чергу, зумовлює актуальність створення спеціалізованих комплексів, здатних забезпечити надійне та безпечне зберігання зерна, щоб це забезпечити, необхідно використовувати системи термометрії.

Існуючі підходи до управління вимірювання температури зерна, використовують ручну оперативну настройку систем термометрії. Це призводить до ускладнення процесів вимірювання температури зерна та підвищення її вартості. Присутність людського чинника в цьому процесі затрудняє визначення вартості процесу, а також виникає необхідність оперативного урахування стану механізмів елеватора по факторам зайнятості та

справності, що призводить до небажаних витрат енергії та простоям на виробництві. Це веде до небажаних витрат. Враховуючи вище перелічене, очевидно, що досить актуальною стає задача побудови алгоритму оперативного вимірювання температури зерна за комплексним критерієм справності, зайнятості та енергоспоживання вузлів транспортно-технологічної мережі елеватора.

1.2. Елеватор як об'єкт управління

Зернопереробна галузь в українському агропромисловому комплексі займає провідне місце по значущості позицій у вирішенні продовольчих проблем, пов'язаних із зберіганням і переробкою зерна. Це питання є стратегічним в аграрній політиці країни.

Основним ланцюгом даної галузі є комбінати хлібопродуктів. Організаційно-технологічна структура комбінатів хлібопродуктів обов'язково включає в себе зерновий елеватор, в якому і реалізований технологічний процес по збереженню і переробці зерна.

Головне призначення елеваторів – це зберігання зерна.

Зберігання зерна здійснюється на елеваторах — високотехнологічних механізованих об’єктах, призначених для тривалого збереження зернової продукції. Основні сховища представлені залізобетонними силосами циліндричної форми, встановленими у вертикальному положенні. Їхні типові розміри варіюються в межах 6–10 м у діаметрі та 15–30 м у висоту, а загальна кількість силосів на одному об’єкті може досягати кількох десятків. Підприємства елеваторної промисловості бувають:

- Заготовчими;
- Проміжними;
- виробничими.

Як показано на рис. 1.1 до заготовчих підприємств відносять хлібоприймальні; до проміжних – базисні, перевалочні, фондові; до виробничих – виробничі, портові, реалізаційні.

Рисунок 1.1 – Структурна схема елеваторної промисловості

Хлібоприймальні (заготівельні) підприємства (зерносховища) забезпечують прийом зерна від хлібоздавачів (з автотранспорту) та його первинну обробку (очистку, сушку і т. п.), а також зберігання та відвантаження за призначенням, в основному залізничним і водним транспортом.

Базисні служать для зберігання оперативних запасів зерна для поточного споживання. Основні операції, що виконуються на них: прийом зерна, очищення і сушка, довготривале зберігання, відвантаження на залізничний і водний транспорт. Базисні елеватори відрізняються великою місткістю і високою

продуктивністю обладнання. Розташовують їх на великих вузлових залізничних станціях, на пересіченні залізничних і водних шляхів [2].

Перевалочні елеватори призначені для перевантаження зерна з одного виду транспорту на інший, наприклад з водного на залізничний, або навпаки; з вузької колії на широку. Вони оснащуються транспортуючим обладнанням великої продуктивності.

Фондові елеватори використовують для зберігання державних зернових резервів. Зерно, закладене в державні резерви, зберігають протягом приблизно 3-4 років. Вони призначені для зберігання державних запасів зерна в обсязі 150-300 тис. тон.

Враховуючи необхідність зберігання великого об'єму зерна, застосовуються силоси великої місткості по 50 тис. тон кожен. Кількість ємностей – 3-6 штук.

Продуктивність прийомки зерна з автомобільного транспорту – 2 лінії по 150 т/год, продуктивність прийомки з ж/д транспорту та відвантаження в ж/д вагонів – 150 т/год, продуктивність сушарки – 50-150 т/год.

Передбачена система контролю температури і система активного вентилювання, яка забезпечує якісне зберігання зерна протягом 36 місяців.

Виробничі елеватори постачають зерно зернопереробним підприємствам (мель-, крупо- і комбікормовим заводам, олійнобійні, крохмалопатові і ін.) Призначення обробки зерна, крім загального, для всіх типів зерносховищ – підготовка сировини до переробки відповідно до вимог технології підприємства. Ці зерносховища повинні мати запас зерна, що забезпечить 3-6-місячну роботу підприємства.

Портові елеватори спеціалізуються на забезпеченні процесів експорту зернових культур шляхом їхнього завантаження на морські судна. Вони оснащені комплексом технологічного обладнання, що забезпечує приймання зерна з автомобільного та залізничного транспорту, його короткочасне зберігання, класифікацію, зважування, а також розвантаження та відвантаження як у залізничні вагони, так і в трюми суден. Типова загальна місткість таких

елеваторів становить 50–150 тис. тон, з рекомендованою ємністю окремих сховищ у межах 5–15 тис. тон і загальною кількістю силосів — 10–12 одиниць. Продуктивність приймання зерна з автомобільного транспорту забезпечується двома-чотирма лініями зі сукупною пропускною здатністю 300–600 т/год. Аналогічна продуктивність характерна для прийому та відвантаження зерна залізничним транспортом. Завантаження зерна в судна здійснюється зі швидкістю 600–1200 т/год, а його вивантаження з суден — до 300 т/год.

Реалізаційні елеватори призначені для приймання зерна та продуктів його переробки, переважно залізничним транспортом, з подальшим зберіганням, поліпшенням якісних характеристик та відвантаженням кінцевому споживачу, здебільшого за допомогою автомобільного транспорту.

Характерною особливістю структури елеваторної промисловості в Україні є інтеграція функціональних можливостей різних типів зерносховищ. Зернозберігальні комплекси, окрім основного функціонального призначення, часто виконують завдання, властиві іншим типам елеваторних об'єктів. Така функціональна універсальність сприяє скороченню логістичного шляху зерна від виробника до кінцевого споживача, оптимізує використання наявних сховищ і технічного оснащення, підвищує загальну ефективність праці та зменшує витрати обігу.

Елеватори – найбільш досконалий тип зерносховищ. Вони є комплексом наступних основних споруд: робочі будівлі, де розміщено основне технологічне та транспортне обладнання, силосні корпуси для зберігання зерна, приймальні та відпускні пристрої для різних видів транспорту, зерносушарки, цехи (склади) відходів.

Незважаючи на різноманіття типів елеваторів, основне їх призначення в принципі формулюється однаково: прийняти зерно, піддати його обробці (очищенню, сушінню, активній вентиляції та ін.), забезпечити надійне збереження, відвантажити користувачу. Переміщення зерна в елеваторі між його основними компонентами забезпечується різним транспортним обладнанням: конвеєрами, норіями та навантажувачами. Шлях переміщення зерна з одного

технологічного об'єкта на інший, за допомогою транспортного обладнання, називається транспортно-технологічним маршрутом.

Елеватор потрібен для навантаження, розвантаження, зберігання та сортування сировини. Дане виробництво є вибухо- та пожежонебезпечним виробництвом, внаслідок можливості самозаймання зерна, наявність вибухонебезпечного пилу.

У виробничому процесі елеватора зерно перебуватиме у постійному русі. Переміщення складається з великої кількості окремих етапів, які у свою чергу складаються з конкретних операцій. Для безпечного видалення вибухонебезпечного пилу використовують системи аерації, які безперервно стежать за кожним технологічним вузлом та всіма маршрутами [3].

На рис. 1.2 представлено технологічну схему хлібозаготівельного підприємства у загальному вигляді, на якій зазначено послідовність виконання можливих операцій.

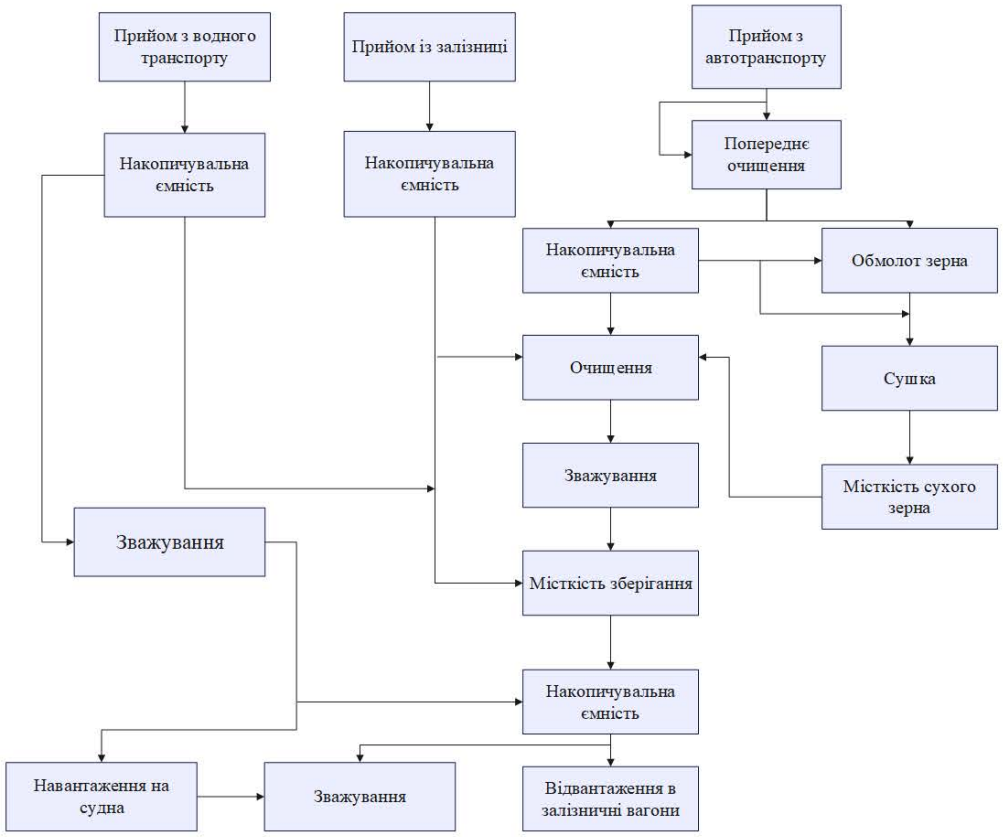


Рисунок 1.2 – Технологічна схема хлібозаготівельного підприємства

1.3. Актуальність кваліфікаційної роботи

Впровадження інформаційних технологій в аграрний сектор України є ключовим для забезпечення ефективного розвитку та конкурентоспроможності сільськогосподарського виробництва. Також впровадження інформаційних технологій в аграрний сектор України є важливим кроком до інноваційного розвитку та підвищення ефективності виробництва. Це допоможе забезпечити стале зростання та розвиток аграрної економіки країни. Дані аспекти розкривають актуальність цього процесу:

1. Підвищення ефективності: інформаційні технології дозволяють оптимізувати виробничі процеси, зменшувати витрати та підвищувати прибуток. Наприклад, технології точного сільського господарства допомагають точно визначати потреби рослин та ефективно використовувати ресурси.

2. Прогнозування та моніторинг: інформаційні технології дозволяють прогнозувати врожайність сільськогосподарських культур та моніторити стан полів в режимі реального часу. Це допомагає вчасно реагувати на потенційні проблеми та приймати обґрунтовані рішення.

3. Обробка полів: Використання дронів для обробки полів проти шкідників та бур'янів за технологією "Smart Spraying" значно підвищує ефективність і зменшує використання хімікатів.

4. Управління та прийняття рішень: інформаційні технології допомагають приймати обґрунтовані управлінські рішення на основі аналізу великих масивів даних. Це сприяє мінімізації витрат, максимізації прибутку та підвищенню конкурентоздатності.

5. Організаційно-правовий аспект: Впровадження інформаційно-комунікаційних технологій в аграрний сектор вимагає комплексного підходу, включаючи організаційне, інституційне, правове та інформаційне забезпечення. Це допомагає забезпечити ефективність та безпеку використання інформаційно-комунікаційних технологій.

6. Перспективи використання цифрових технологій: Цифрові технології можуть значно підвищити ефективність функціонування сільськогосподарських підприємств. Вони дозволяють зберігати великі масиви даних, проводити їх аналіз та приймати обґрунтовані рішення.

7. Інтернет-платформи та маркетплейси: Використання інформаційних технологій дозволяє створювати інтернет-платформи та маркетплейси, які сприяють взаємодії між агротоваровиробниками та споживачами. Це допомагає збільшити прозорість та ефективність торгівлі сільськогосподарською продукцією.

8. Технології точного сільського господарства: Ці технології дозволяють точно визначати потреби рослин та ефективно використовувати ресурси, що призводить до зниження витрат та підвищення врожайності.

9. Прогнозування та моніторинг: інформаційні технології дозволяють прогнозувати врожайність сільськогосподарських культур та моніторити стан полів в режимі реального часу. Це допомагає вчасно реагувати на потенційні проблеми та приймати обґрунтовані рішення.

Ці аспекти підкреслюють важливість впровадження інформаційних технологій в аграрний сектор України для підвищення ефективності та конкурентоспроможності виробництва [4].

1.4. Аналіз шляхів розв'язання впровадження інформаційних технологій в аграрний сектор України

Аграрний сектор посідає провідне місце в структурі національної економіки України, забезпечуючи вагомий внесок у формування валового внутрішнього продукту та обсягів експорту. Інтеграція інформаційних технологій у виробничі процеси сільського господарства сприяє зростанню продуктивності, оптимізації витрат і підвищенню якості сільськогосподарської продукції. Разом з тим, реалізація цифрових рішень супроводжується низкою

проблемних аспектів, що вимагають всебічного та системного підходу до їх вирішення. Виклики впровадження інформаційних технологій в аграрний сектор:

1. Низька цифрова грамотність фермерів – багато виробників не мають необхідних знань для використання сучасних технологій.
2. Фінансові обмеження – високі витрати на інфраструктуру та програмне забезпечення.
3. Недостатній рівень інтернет-покриття у сільській місцевості – обмежений доступ до хмарних сервісів та онлайн-платформ.
4. Слабка інтеграція рішень у традиційні аграрні процеси – відсутність стандартів і взаємодії між різними системами.
5. Кадровий дефіцит – нестача фахівців з аграрного консалтингу та обслуговування систем.

Шляхи розв'язання проблем впровадження інформаційних технологій:

1. Освітні програми та підвищення цифрової грамотності:

- Організація навчальних курсів для фермерів щодо використання цифрових технологій.
- Інтеграція предметів з аграрних технологій у навчальні програми університетів.
- Запуск державних та приватних ініціатив з підвищення кваліфікації спеціалістів.

2. Фінансові механізми підтримки:

- Надання державних та міжнародних грантів на цифровізацію фермерських господарств.
- Розвиток програм лізингу та кредитування на придбання агро-рішень.
- Податкові пільги для компаній, що інвестують у цифрові технології.

3. Розвиток інфраструктури:

- Розширення покриття широкосмугового інтернету в сільській місцевості.

- Впровадження супутникового інтернету для віддалених регіонів.
- Підтримка стартапів, що створюють мобільні додатки та платформи для агросектору.

4. Стандартизація та інтеграція систем:

- Впровадження єдиних стандартів обміну даними між аграрними підприємствами та державними органами.
- Розвиток платформ для інтеграції розумних систем управління сільським господарством.
- Використання великих даних для прогнозування врожайності та оптимізації витрат.

5. Підготовка кадрів та розвиток екосистеми:

- Створення центрів компетенцій з цифрового агровиробництва.
- Співпраця між аграрними університетами та компаніями для розробки нових технологій.
- Залучення міжнародних експертів та консалтингових компаній для адаптації найкращих практик.

Впровадження інформаційних технологій у аграрний сектор України є важливим напрямком для підвищення його конкурентоспроможності. Комплексний підхід, що включає освітні ініціативи, фінансову підтримку, розвиток інфраструктури, стандартизацію та кадрову підготовку, дозволить подолати існуючі бар'єри. Активна взаємодія держави, бізнесу та наукової спільноти допоможе зробити українське сільське господарство більш ефективним та стійким до сучасних викликів [5].

1.5. Постановка задачі

Завданням кваліфікаційної роботи бакалавра є підвищення ефективності керування системою термометрії елеватора. Це необхідно виконати за рахунок алгоритмічної оптимізації задачі оперативного вимірювання температури зерна

та підвищення ефективності керування системою термометрії елеватора за комплексним критерієм як складової програмного забезпечення автоматизація робочого місця оператора-технолога автоматизованими системами управління термометрією елеватора.

Створення програмного забезпечення, яке забезпечує автоматизований контроль температури зернових мас у силосах елеватора для попередження їх псування, самозайманння та інших негативних процесів, пов'язаних із перегрівом або підвищеною вологістю.

Основні завдання.

1. Збір даних:
    - Отримання даних у реальному часі або з заданою періодичністю.
2. Обробка та аналіз даних:
    - Визначення середніх, мінімальних та максимальних значень температури.
3. Візуалізація:
    - Відображення даних у вигляді графіків або таблиць.
4. Архівування та звітність:
    - Збереження історії показів для подальшого аналізу.
    - Формування звітів за заданими періодами (добових, тижневих, сезонних).

Вимоги до програмного забезпечення:

- Сумісність: Підтримка різних типів датчиів (Wi-Fi, LoRaWAN, Bluetooth тощо).
- Масштабованість: Можливість обробки даних з сотень/тисяч точок вимірювання.
- Надійність: Робота в умовах високого навантаження та при відсутності інтернет-з'єднання (офлайн-режим).

Програмний комплекс, який дозволяє оперативно виявляти та запобігати критичним змінам температури зерна, знижуючи ризики втрат і підвищуючи ефективність зберігання.

1.6. Аналіз літературних джерел

Впровадження інформаційних технологій в аграрний сектор є важливим напрямом розвитку сільського господарства України. Аналіз літературних джерел свідчить про активне використання таких рішень, як точне землеробство, автоматизація агровиробництва, використання дронів та супутникового моніторингу.

Наукові роботи та звіти міжнародних організацій (рис. 1.3) відзначають позитивний вплив цифровізації на продуктивність аграрного сектору.

Рисунок 1.3 – Стійкість українського агросектору [6]

Зокрема, у роботах українських та зарубіжних вчених розглядаються такі аспекти:

- Використання датчиків і інтернет речей для моніторингу стану ґрунту та рослин.

- Геоінформаційні системи для оптимізації управління ресурсами.
- Системи для планування та управління агропідприємствами.
- Використання великих даних та штучного інтелекту для прогнозування врожайності та оптимізації виробничих процесів.

Впровадження інформаційних технологій в аграрний сектор України є важливим аспектом для підвищення ефективності та продуктивності. Дані джерела, є ключовими для аналізу:

Особливості впровадження інформаційних технологій в аграрний сектор України - стаття, яка досліджує функціонування вітчизняних підприємств аграрного сектору в сучасних мовах цифровізації економіки. Вона також розглядає типи інформаційних технологій, що використовуються в сільському господарстві, такі як точне сільське господарство, прогнозування врожайності та моніторинг стану полів [6].

Цифрові технології в агрономії – від теорії до практики - монографія, яка розглядає технології, застосовувані в сільськогосподарській практиці, та надає рекомендації щодо їх впровадження [7].

Впровадження інформаційно-комунікаційних технологій в аграрний сектор економіки України: організаційно-правовий аспект - монографія, яка досліджує теоретичні, методологічні та практичні проблеми впровадження інформаційно-комунікаційних технологій в аграрний сектор України [8].

Дослідження також вказують на певні бар'єри у впровадженні рішень в аграрному секторі України, серед яких недостатнє фінансування, відсутність кваліфікованих спеціалістів та слабка цифрова інфраструктура. Однак, державні програми та ініціативи Євросоюзу сприяють впровадженню новітніх технологій, що дає перспективи для подальшого розвитку цифрового аграрного сектору в Україні.

1.7. Висновки до першого розділу

Зернова галузь є стратегічно важливою для економіки України, оскільки вона забезпечує продовольчу безпеку, формує аграрний експорт та є основою для розвитку агропромислового комплексу. Однак на сьогоднішній день стан галузі залишається незадовільним через:

- застарілу матеріально-технічну базу, яка не відповідає сучасним стандартам;
- недостатнє фінансування, що обмежує впровадження інноваційних технологій;
- вплив кліматичних чинників та неефективне використання земельних ресурсів, що призводить до погіршення якості зерна;
- недосконалість систем зберігання зерна, зокрема відсутність автоматизованого контролю температури, що збільшує ризики псування продукції.

Елеватори як ключова ланка зернової галузі відіграють вирішальну роль у зберіганні, обробці та транспортуванні зерна. Вони класифікуються за функціональним призначенням (заготівельні, базисні, перевалочні, фондові, виробничі, портові, реалізаційні), але їхня ефективність обмежується:

- ручним управлінням процесами, що збільшує витрати та ймовірність помилок;
- відсутністю інтегрованих систем моніторингу (наприклад, термометрії та вентиляції);
- необхідністю автоматизації для зменшення впливу людського чинника та оптимізації виробничих процесів.

Впровадження інформаційних технологій у зернову галузь є ключовим напрямком для підвищення ефективності. Серед основних технологічних рішень:

- Системи управління агропідприємствами – для управління ресурсами підприємств;
- Інтернет речей та автоматизовані системи контролю та моніторингу – для моніторингу стану зерна, контролю температури та вологості;
- хмарні технології та Big Data – для аналізу великих масивів даних;
- автоматизовані системи термометрії – для попередження псування зерна.

Однак основні виклики впровадження інформаційних технологій в аграрному секторі включають:

- високу вартість інноваційних рішень;
- недостатню цифрову грамотність серед працівників;
- недосконалу інфраструктуру (особливо в сільській місцевості);
- проблеми з інтеграцією нових систем у традиційні процеси.

Перспективи розвитку пов'язані з:

- державною підтримкою (гранти, субсидії, навчальні програми);
- розвитком інтернету речей та штучного інтелекту для предикативної аналітики;
- автоматизацією логістики та зберігання зерна;
- використанням дронів та супутникового моніторингу для оцінки стану полів.

Крім того, значна увага повинна приділятися стандартизації та уніфікації цифрових рішень у межах галузі, що сприятиме підвищенню сумісності обладнання та програмного забезпечення різних виробників, забезпеченню прозорості бізнес-процесів і формуванню єдиної цифрової екосистеми агропромислового виробництва. Удосконалення нормативно-правової бази щодо впровадження цифрових технологій у сфері зберігання та обігу зерна також є ключовим чинником для стимулювання інноваційної діяльності підприємств.

Для підвищення конкурентоспроможності українського зернового сектора необхідно активне впровадження сучасних рішень, які дозволять оптимізувати виробничі процеси, зменшити витрати та підвищити якість продукції.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ РОЗВ'ЯЗКІВ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1. Інформаційні та комп'ютерні технології

На підприємствах агропромислового комплексу потрібно активно впроваджувати сучасні інформаційні та комп'ютерні технології для підвищення ефективності виробництва, зменшення витрат і покращення управління ресурсами.

Основні технології, які потрібно застосовувати в зерновій галузі:

- Системи управління агропідприємствами – автоматизація управління фінансами, обліком ресурсів, виробничими процесами та логістикою.
- Інтернет речей – включає датчики для моніторингу вологості ґрунту, температури, рівня освітлення та інших параметрів.
- Автоматизовані системи контролю та моніторингу – використовуються для віддаленого управління зрошувальними системами, елеваторами та фермерським обладнанням, дозволяють ефективно використовувати ресурси та зменшувати витрати.
- Хмарні технології та Big Data – використовуються для зберігання та обробки великих обсягів даних.

Також потрібно використовувати різноманітне спеціалізоване програмне забезпечення не серійного виробництва для індивідуальних потреб підприємств агропромислового комплексу.

Програми комп'ютерної системи термометрії призначені для запуску дистанційного вимірювання температури в силосах зберігання зерна, а також збереження та перегляду результатів вимірювання у вигляді протоколу вимірювання, температурного поля або графіків.

Системи управління і контролю технологічним процесом сушки зерна виконують функції автоматизації контролю і управління процесом сушіння зерна [3].

2.2. Ефективність роботи програмного забезпечення

Попри значний прогрес у розвитку програмного забезпечення для аграрного сектору, він має певні слабкі сторони, які потребують вдосконалення. Автоматизація виробничих процесів, усунення проблем інтеграції та підвищення рівня безпеки сприяють підвищенню ефективності та прибутковості підприємств.

Сучасні інформаційні технології значно покращують управління аграрними підприємствами, проте мають певні недоліки та виклики, які потребують уваги. Основні слабкі сторони програмного забезпечення:

- Висока вартість впровадження – багато сучасних систем мають високу вартість, що робить їх недоступними для малих і середніх підприємств, додаткові фінансові вкладення у технічне обслуговування та оновлення.
- Складність у використанні та навчанні персоналу – підприємства стикаються з проблемою нестачі кваліфікованих спеціалістів, які можуть встановлювати та обслуговувати програмне забезпечення, персонал потребує навчання, що потребує додаткових ресурсів і часу.
- Проблеми з безпекою даних – хмарні сервіси, пристрої та мобільні додатки, які використовуються можуть бути вразливими до кібератак, втрата або крадіжка даних може призвести до серйозних фінансових та репутаційних ризиків.
- Застарілі системи та їхня низька продуктивність – застарілі програми, що знижує ефективність роботи, відсутність автоматизації та використання ручного введення даних спричиняє помилки та затримки у виробничих процесах.
- Проблеми з інтеграцією новітніх технологій – впровадження інноваційних рішень потребує значних змін у внутрішніх процесах компанії.
- Можливість автоматизації процесів – використання сучасного програмного забезпечення дозволяє зменшити вплив людського фактору,

підвищити швидкість обробки інформації та покращити прийняття рішень на основі даних.

Впровадження інформаційних технологій в аграрний сектор України сприятиме зростанню продуктивності праці та вирішенню багатьох актуальних проблем. Завдяки цим технологіям можна зберігати й аналізувати великі обсяги даних, що дозволяє оптимізувати витрати та підвищувати прибутковість аграрних підприємств.

Застосування інформаційних технологій зробить значний внесок у розвиток інформаційного забезпечення агропромислового комплексу України, що сприятиме підвищенню конкурентоспроможності сільськогосподарського виробництва, особливо в умовах розширення експорту. Подальші дослідження будуть зосереджені на використанні сучасних технологій для автоматизації управління фермерськими господарствами в Україні [9].

2.3. Загальні напрямки дослідження

Метою дослідження впровадження інформаційних технологій в аграрний сектор України є вивчення шляхів впровадження інформаційних технологій у аграрний сектор України, аналіз їхнього впливу на ефективність виробництва та визначення основних викликів і перспектив цифровізації сільського господарства, підвищення продуктивності та ефективності аграрного виробництва шляхом інтеграції сучасних цифрових рішень.

Основні завдання дослідження включають:

1. Аналіз існуючих рішень в аграрному секторі: вивчення вже використовуваних технологій, таких як точне землеробство, автоматизовані системи управління фермами, прогнозування врожайності та їх ефективність у сільському господарстві. Оцінка сучасного стану впровадження інформаційних технологій у аграрному секторі України.

2. Оцінка потенціалу нових технологій: дослідження нових і перспективних рішень, таких як дрони для моніторингу полів, сенсори для вимірювання вологості ґрунту та інші інновації.

3. Розробка стратегій впровадження: визначення найкращих підходів для впровадження інформаційних технологій в аграрні підприємства, враховуючи специфіку регіону та потреби фермерів. Визначити основні бар'єри, що заважають цифровізації агросфери.

4. Оцінка економічної ефективності: вивчення економічних аспектів впровадження інформаційних технологій, таких як зниження витрат, підвищення врожайності та покращення якості продукції.

5. Аналіз соціальних та екологічних впливів: вивчення впливу впровадження інформаційних технологій на соціальну структуру сільських громад та екологічні аспекти аграрного виробництва.

6. Аналіз міжнародного досвіду впровадження інформаційних технологій у аграрний сектор.

Це загальні напрямки дослідження, які можуть бути адаптовані відповідно до конкретних умов та потреб українського аграрного сектору [10].

2.4. Проектування й дослідження автоматизованих інформаційних систем в аграрному секторі

Автоматизовані інформаційні системи є невід'ємною складовою цифрової трансформації аграрного сектору. Проектування таких систем передбачає аналіз потреб агропідприємств, розробку програмного забезпечення, впровадження технологічних рішень та їх інтеграцію з існуючими системами управління.

Основними напрямами проектування автоматизованих інформаційних систем є:

- Розробка систем моніторингу стану посівів та ґрунту – використання датчиків та безпілотних літальних апаратів для збору даних у режимі реального часу.
- Інтеграція з геоінформаційними системами – забезпечення точного аналізу та прогнозування на основі просторових даних.
- Розробка систем підтримки прийняття рішень – використання аналітичних алгоритмів та штучного інтелекту для рекомендацій щодо оптимального внесення добрив, зрошення та захисту рослин.
- Автоматизація управління агропідприємствами – створення систем для планування ресурсів, ведення документації, аналізу економічних показників.
- Розробка мобільних додатків для аграріїв – забезпечення доступу до інформації про стан полів, погоду, ринкові ціни та агрономічні поради у зручному форматі.

Дослідження ефективності автоматизованих інформаційних систем дозволяє оцінити їх вплив на продуктивність та рентабельність аграрного виробництва. Аналіз існуючих рішень та тестування нових технологій сприяє їх удосконаленню та адаптації до специфіки українського сільського господарства. Завдяки впровадженню автоматизованої інформаційної системи аграрії можуть більш ефективно управляти ресурсами, знижувати витрати та підвищувати врожайність, що в результаті сприяє сталому розвитку аграрного сектору України [8].

2.5. Автоматизація технічного обслуговування

Автоматизація технічного обслуговування в аграрному секторі має великий потенціал для підвищення ефективності та зниження витрат. Ось кілька аспектів, які можна враховувати при розробці таких систем:

1. Моніторинг стану техніки.

- Датчики та інтернет речей: Встановлення датчиків на сільськогосподарську техніку для моніторингу її стану в режимі реального часу.
- Прогнозування поломок: Аналіз даних, зібраних датчиками, для прогнозування можливих поломок і запобігання аварій.

2. Автоматизовані системи обслуговування.

- Планування обслуговування: Розробка програмного забезпечення для автоматичного планування технічного обслуговування на основі даних про використання техніки.
- Автономні дрони та роботи: Використання дронів та роботів для виконання завдань обслуговування, таких як змащування або заміна деталей.

3. Інтеграція з системами управління агропідприємствами.

- Управління запасами: Інтеграція з системами для автоматичного замовлення запчастин та матеріалів на основі даних про витрати та запаси.
- Аналіз ефективності: Використання даних для аналізу ефективності технічного обслуговування та виявлення областей для покращення.

4. Візуалізація та звітність.

- Дашборди та звіти: Розробка інтуїтивно зрозумілих дашбордів для візуалізації стану техніки та звітів про проведені обслуговування.
- Мобільні додатки: Створення мобільних додатків для доступу до інформації про технічне обслуговування з будь-якого місця.

5. Використання штучного інтелекту та машинного навчання.

- Аналіз даних: Використання алгоритмів машинного навчання для аналізу великих обсягів даних про техніку та обслуговування.
- Прогнозування потреб: Алгоритми штучного інтелекту для прогнозування майбутніх потреб в обслуговуванні на основі історичних даних.
- Автоматизація технічного обслуговування може значно покращити надійність та ефективність сільськогосподарської техніки, знижуючи витрати на ремонт та простої [10].

2.6. Огляд програмного забезпечення систем термометрії зерносховищ

Програмне забезпечення системи термометрії призначено для моніторингу температури зернових культур під час зберігання (рис. 2.1). Воно дозволяє запобігати самозайманню, псуванню зерна та підвищує ефективність управління складськими ресурсами.

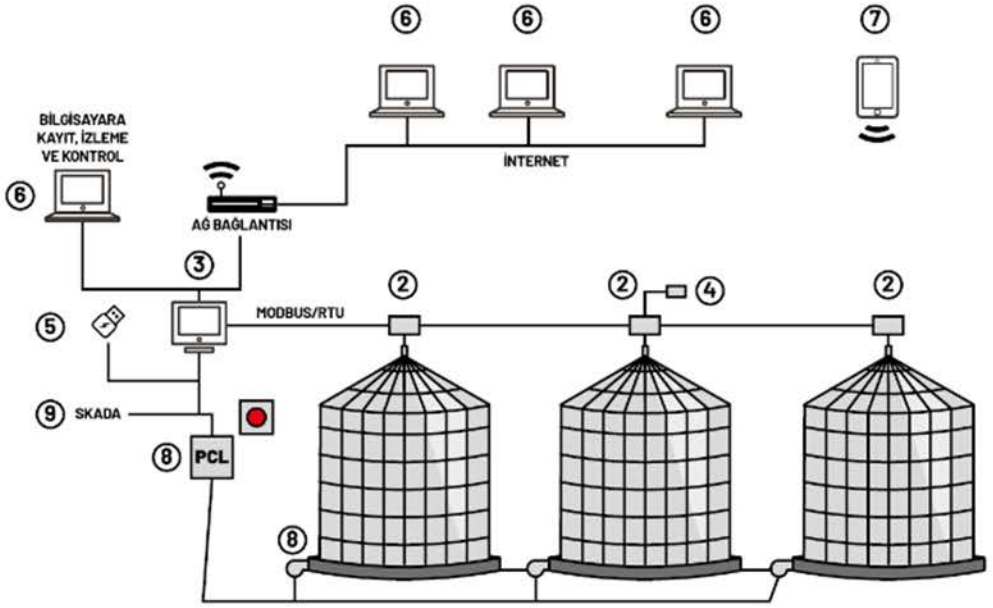


Рисунок 2.1 – Система моніторингу і контроля температури силосу

В залежності від типу зерносховища програмне забезпечення має вирішувати такі цілі і завдання:

Цілі:

- програмне забезпечення для автоматизованого контролю температури на елеваторі;
- підвищення рівня безпеки зберігання зернових культур;
- автоматизація збору та аналізу даних.

Завдання:

- реалізація збору даних з датчиків температури;
- візуалізація температурних показників;
- автоматичне оповіщення у разі перевищення допустимих значень;
- збереження історії вимірювань та формування звітності.

Архітектура систем термометрії складається з наступних основних компонентів:

- датчики температури (підключені через проводову або бездротову мережу);
- контролер збору даних (мікроконтролер або промисловий контролер);
- сервер обробки (локальний або хмарний сервер для аналізу даних);
- клієнтський додаток (веб- або мобільний додаток для моніторингу та управління).

Функціональні можливості включають в себе:

- збір даних – періодичне отримання температурних показників з датчиків;
- аналіз даних – побудова графіків, обчислення середніх та граничних значень;
- оповіщення – SMS, e-mail або push-сповіщення у разі аварійних ситуацій;
- інтерфейс користувача – інтуїтивно зрозуміла панель керування з відображенням температурних зон;
- інтеграція – можливість інтеграції з іншими системами автоматизації елеватора.

Технології та засоби реалізації притаманне різноманіття, наприклад:

- мова програмування: C#, Python (для збору та аналізу даних);
- база даних: PostgreSQL / MySQL;
- фронтенд: React, Angular (веб-інтерфейс);
- зв'язок з датчиками: MQTT, Modbus, LoRaWAN.

Результатом використання програмного забезпечення є:

- підвищення ефективності зберігання зерна за рахунок оперативного моніторингу температури;

- мінімізація ризику псування зернових культур;
- автоматизація звітності та управлінських процесів.

Програмне забезпечення систем термометрії зерносховищ дозволяє значно підвищити ефективність управління процесами зберігання зерна, зменшити ризики втрат та покращити контроль якості продукції [11].

2.7. Опис алгоритму і програмного забезпечення

Функціональна модель завдання для проекту програмного забезпечення системи термометрії елеватора зображена на рисунку 2.2. Вона ілюструє основні процеси, включаючи збір даних, їх аналіз, візуалізацію, оповіщення та інтеграцію з іншими системами.

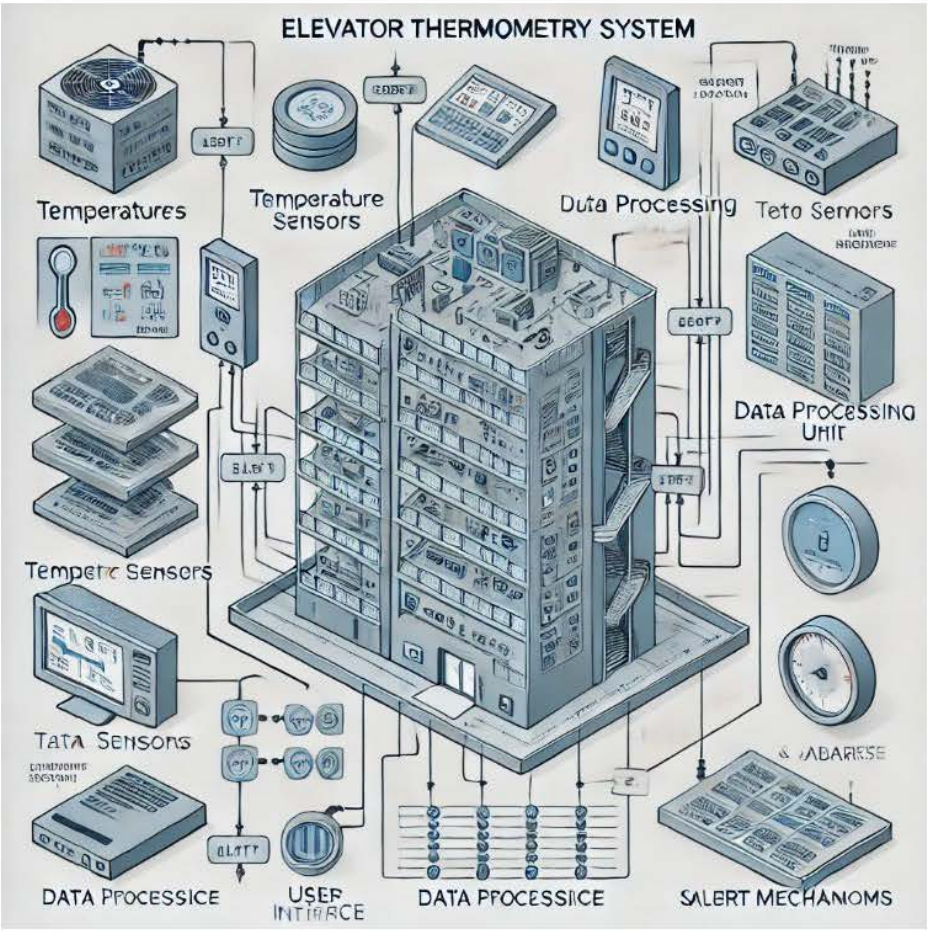


Рисунок 2.2 – Функціональна модель [1]

1. Збір та передача даних:
   - Моніторинг температури в реальному часі;
   - Збір даних із датчиків у різних зонах елеватора;
   - Фільтрація та первинна обробка показників;
   - Передача даних на сервер або у хмарне сховище.
2. Обробка та аналіз інформації:
   - Набір середньої, мінімальної та максимальної температури;
   - Виявлення аномальних значень;
   - Прогнозування можливих відхилень за допомогою алгоритмів штучного інтелекту;
   - Формування звітності та логування історичних даних.
3. Візуалізація та управління:
   - Відображення температурного поля елеватора у вигляді графіків та карт;
   - Генерація динамічних звітів та графіків;
   - Налаштування інтерфейсу користувача для аналізу показників;
   - Надання доступу до даних.
4. Сповіщення та реагування:
   - Автоматичне повідомлення операторів про критичні зміни температури;
   - Генерація аварійних сигналів при перевищенні допустимих норм.
5. Інтеграція з іншими системами:
   - Підключення до системи управління агропідприємства;
   - Обмін даними з системами контролю вологості та вентиляції;
   - Інтеграція з метеостанціями для аналізу зовнішніх впливів.

Ця модель показує ключові завдання системи та забезпечує її ефективне функціонування.

Згідно моделі розроблено алгоритм роботи програмного забезпечення, що зображений на рисунку 2.3, який включає в себе:

1. Запуск програми:
   - Ініціалізація системи: запуск основного програмного середовища, ініціалізація змінних, ресурсів і компонентів.

- Перевірка підключених датчиків температури: опитування доступних інтерфейсів, виявлення несправностей або відсутності з'єднання.

2. Зчитування температурних даних:
   - Утримання поточних показників температури з датчиків: періодичне або безперервне опитування всіх підключених сенсорів.
   - Фільтрація та обробка даних: виключення аномальних значень, згладжування коливань, застосування алгоритмів усереднення або фільтрів (наприклад, фільтр Калмана або медіанний фільтр).
3. Аналіз отриманих даних:
   - Порівняння з допустимими значеннями температури: перевірка кожного значення на відповідність нормам (мінімальний та максимальний пороги).
   - Визначення критичних зон: виявлення ділянок або пристроїв, де температура вийшла за межі допустимого діапазону.
4. Відображення інформації:
   - Візуалізація даних на екрані у вигляді графіків або таблиць: відображення поточних температур у зрозумілому для користувача вигляді.
   - Оновлення температурного поля в реальному часі: динамічне оновлення інтерфейсу зі змінами температур.
5. Формування звітів:
   - Збереження історичних даних у базі: фіксація усіх показників із зазначенням часу для подальшого аналізу.
   - Генерація аналітичних звітів: створення щоденних, тижневих або аварійних звітів із графіками, тенденціями та висновками.
6. Реакція на перевищення температури:
   - Сповіщення оператора: подача сигналів тривоги, відображення спливаючих повідомлень, надсилання SMS/електронних листів.
   - Активація системи охолодження або вентиляції (якщо передбачено): автоматичне керування виконавчими пристроями для стабілізації температури.
7. Циклічне оновлення:

- Програма повертається до етапу зчитування температури: постійний цикл контролю системи.
- Повторення процесу у встановленому часовому інтервалі: виконання циклу з заданою частотою (наприклад, кожні 10 секунд).

8. Завершення роботи:
   - Вимкнення системи за командою оператора або при аварійній ситуації: контрольоване завершення роботи з виведенням відповідного повідомлення.
   - Збереження останніх зібраних даних: архівація поточних даних для забезпечення цілісності при повторному запуску.

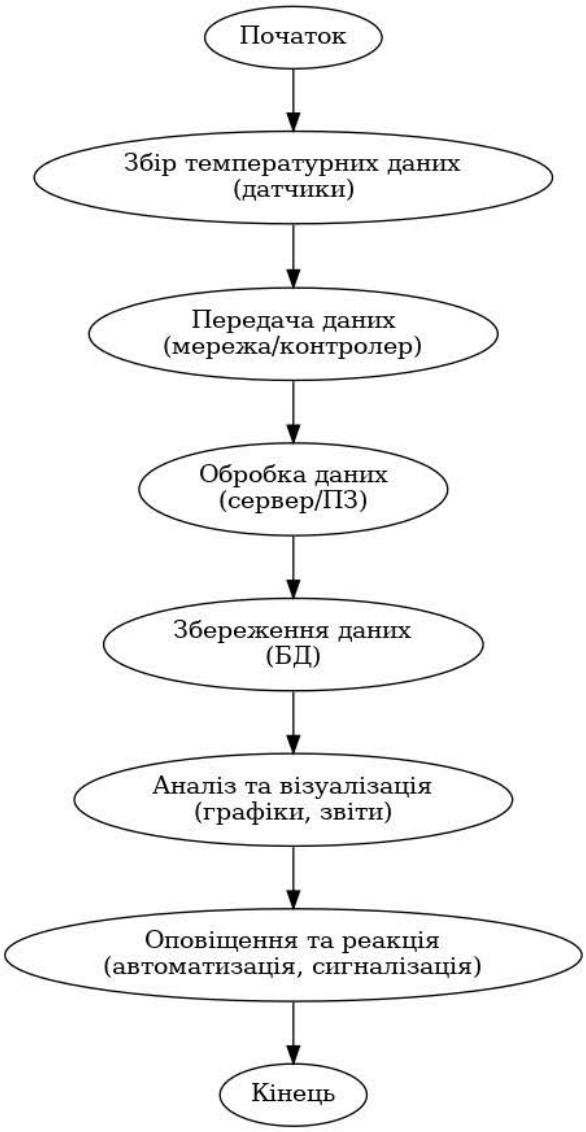


Рисунок 2.3 – Алгоритм роботи програмного забезпечення

Цей алгоритм забезпечує безперервний моніторинг температури в елеваторі та оперативне реагування на критичні зміни [9].

2.8. Висновки до другого розділу

Дослідження впровадження інформаційних технологій у аграрний сектор України підтверджує їх ключову роль у підвищенні ефективності, продуктивності та конкурентоспроможності сільського господарства. Аналіз літературних джерел та сучасних практик свідчить про активне використання таких інноваційних рішень, як:

- Точне землеробство;
- Системи управління агропідприємства;
- Великі дані та штучний інтелект для прогнозування врожайності;
- Системи термометрії зерносховищ, що запобігають втратам зерна.

Основні результати дослідження:

1. Переваги інформаційних технологій у сільському господарстві:
    - Зниження витрат за рахунок оптимізації ресурсів (води, добрив, палива);
    - Підвищення врожайності через точний моніторинг стану ґрунту та рослин;
    - Автоматизація звітності та управління, що зменшує вплив людського фактору.
2. Виклики впровадження:
    - Висока вартість рішень, особливо для малих фермерських господарств;
    - Дефіцит кваліфікованих кадрів для обслуговування складних систем;
    - Кібербезпека даних у хмарних рішеннях.
3. Перспективні напрямки розвитку:

- Розширення використання штучного інтелекту та машинного навчання для прогнозування ризиків (погода, шкідники);
- Інтеграція з міжнародними системами (наприклад, європейські цифрові платформи для агроекспорту);
- Розробка доступних мобільних додатків для малих ферм.

4. Автоматизовані системи термометрії зерносховищ як приклад успішного застосування інформаційних технологій:
   - Реалізація реального часу моніторингу температури запобігає псуванню зерна;
   - Автоматичні сповіщення та інтеграція з системами управління агропідприємств зменшують ризики втрат;
   - Використання сучасних технологій (LoRaWAN, Python, React) робить систему масштабованою.

Впровадження інформаційних технологій у аграрний сектор України є необхідним кроком для забезпечення сталого розвитку галузі, зростання експортного потенціалу та адаптації до глобальних викликів, таких як кліматичні зміни. Подолання бар'єрів (фінансування, інфраструктура, кадри) потребує спільних зусиль держави, бізнесу та наукового середовища. Майбутні дослідження мають зосередитись на розробці економічно ефективних рішень для малих господарств та впровадженні цифрових інновацій на всіх рівнях агровиробництва.

Таким чином, цифровізація аграрного сектору – це стратегічний шлях до його модернізації та глобальної конкурентоспроможності.

РОЗДІЛ 3. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОБЛІКУ ТА КОНТРОЛЮ ТЕМПЕРАТУРИ ЗЕРНА НА ЕЛЕВАТОРАХ

3.1. Опис алгоритму і програмного забезпечення

Для розробки поставленої задачі будемо використовувати середовище розробки програмного забезпечення Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення та інші інструменти. Розробку будемо виконувати засобами мови C# та структурно-орієнтованого підходу програмування.

C# – це сучасна, об'єктно-орієнтована мова програмування, розроблена компанією Microsoft у межах платформи .NET. Вона поєднує простоту мов високого рівня з потужними інструментами для розробки додатків різного призначення: від десктопних до веб, мобільних, хмарних та ігрових.

Основні характеристики мови C#:

- Об'єктно-орієнтованість: підтримка інкапсуляції, спадкування, поліморфізму;
- Сильна типізація: забезпечує більшу безпеку та контроль над помилками;
- Автоматичне керування пам’яттю: за допомогою garbage collector, що зменшує ризик витоків пам’яті;
- Підтримка LINQ: дозволяє зручно працювати з колекціями та базами даних у вигляді SQL-подібних запитів;
- Асинхронне програмування (async/await): ефективна робота з потоками та задачами;
- Інтеграція з платформою .NET: надає доступ до великої бібліотеки класів, API та фреймворків;
- Мультиплатформеність: завдяки .NET Core/.NET 5+ C# підтримується на Windows, macOS, Linux.

Переваги C#:

1. Простота та читабельність:

- Синтаксис C# схожий на Java та C++, але більш сучасний і лаконічний;
- Використовує об'єктно-орієнтовне програмування (класи, інтерфейси, успадкування), що полегшує написання структурованого коду;
- Має автоматичне керування пам'яттю (GC – Garbage Collector).

2. Крос-платформенність (.NET Core / .NET 5+):

- C# працює на Windows, Linux, macOS завдяки .NET Core та .NET 5/6/7+;
- Підтримує Docker, Kubernetes, що робить його ідеальним для сучасних хмарних рішень.

3. Велика екосистема (.NET Framework / .NET):

- Доступ до потужних бібліотек для роботи з: базами даних (Entity Framework, Dapper);
- Мережевими технологіями (ASP.NET Core, gRPC, SignalR);
- Графікою та іграми (Unity3D);
- Машинним навчанням (ML.NET);

4. Висока продуктивність:

- JIT-компіляція (Just-In-Time) для швидкої роботи;
- AOT-компіляція (Ahead-Of-Time) у .NET Native та Blazor WASM;
- Підтримка асинхронного програмування (async/await) для ефективної роботи з I/O.

5. Підтримка сучасних парадигм програмування:

- Асинхронність (Task, async/await);
- LINQ (Language Integrated Query) – потужний інструмент для роботи з колекціями;
- Pattern matching (з C# 7.0+) для зручного аналізу даних;
- Records (C# 9.0+) для створення immutable-об'єктів.

6. Розробка під різні сфери:

- Веб (ASP.NET Core, Blazor);
- Десктоп (WPF, WinForms, MAUI);
- Мобільні додатки (Xamarin / .NET MAUI);

- Ігри (Unity3D);
- Cloud & Microservices (Azure, AWS, Docker);
- Інтернет речей (Raspberry Pi, ARM-пристрої).

7. Сильна спільнота та підтримка Microsoft:

- Microsoft активно розвиває C# та .NET;
- Велика кількість документації, форумів (Stack Overflow, GitHub);
- Безкоштовні інструменти (Visual Studio Code, Rider, VS Community).

C# – це ефективна, безпечна та продуктивна мова для створення сучасного програмного забезпечення будь-якої складності. Вона підходить як для початківців, так і для професійних розробників [12].

3.2. Проектування структури програми

Відповідно до принципів системного аналізу, основним етапом життєвого циклу розробки програмного забезпечення є проєктування його архітектури. Цей процес включає:

- декомпозицію функціональних вимог, що забезпечує чітке структурування задач і сприяє ефективному розподілу відповідальностей між компонентами системи;
- проєктування допоміжних модулів, які підтримують основну логіку системи, забезпечуючи її масштабованість, повторне використання коду та спрощення супроводу;
- імплементацію механізмів обробки помилок з метою гарантування стабільності роботи системи в умовах некоректного вводу або збоїв, що є критичним для забезпечення надійної та безперебійної інтерактивної взаємодії з користувачем;
- визначення взаємозв'язків між підсистемами, що дозволяє формалізувати потоки даних і керуючих сигналів, а також забезпечити логічну узгодженість архітектурного рішення;

- вибір архітектурного стилю відповідно до технічних вимог, масштабів проєкту та очікуваних навантажень;
- аналіз варіантів розгортання системи, з урахуванням хмарних, гібридних або локальних середовищ виконання;
- застосування принципів модульності, інкапсуляції та слабкого зв'язування, що забезпечує високий рівень гнучкості, адаптивності та тестованості розроблюваного програмного забезпечення.

Результати моделювання алгоритмічної структури представлені на рис. 3.1.

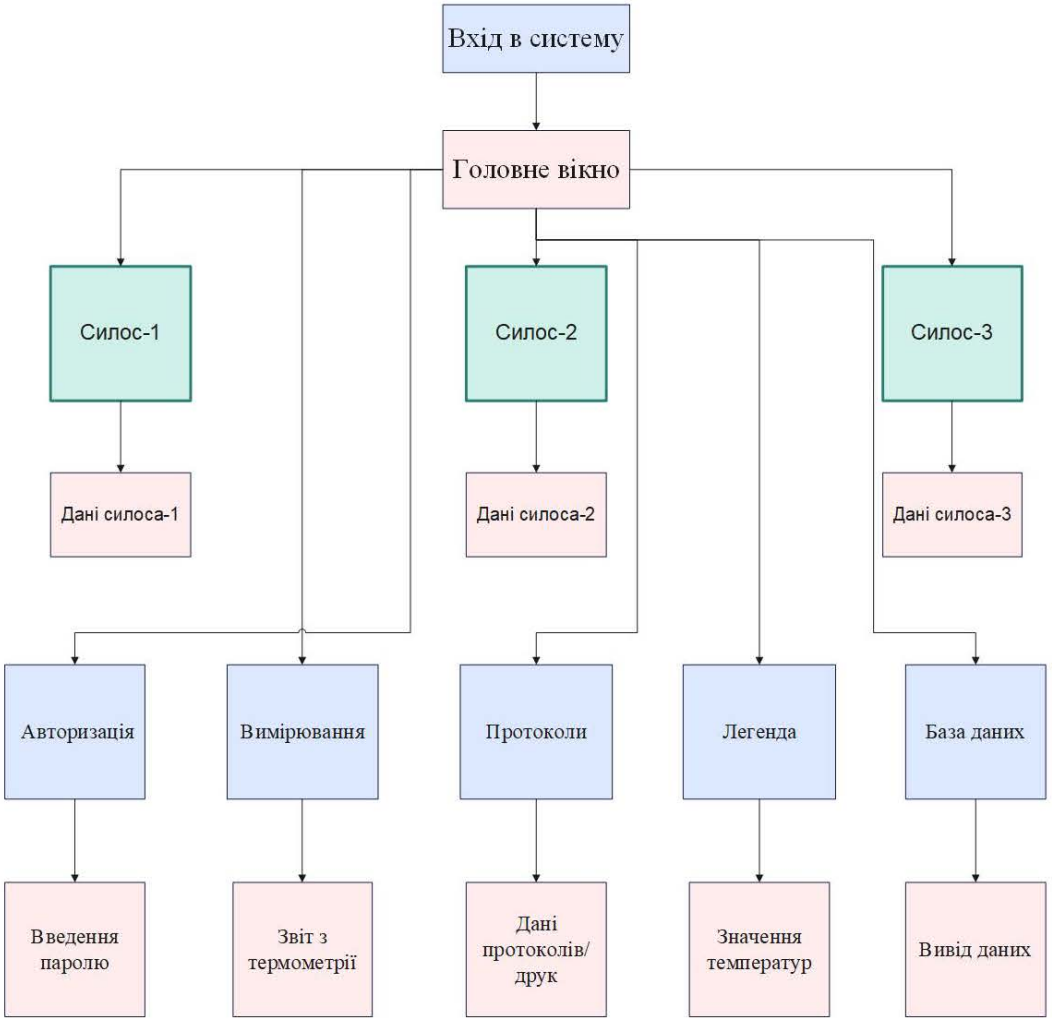


Рисунок 3.1 – Структура роботи програми

3.3. Проектування інтерфейсу користувача

Для забезпечення ефективної взаємодії користувача з програмним продуктом важливим етапом є ретельне проектування інтерфейсу, що сприяє зниженню когнітивного навантаження та підвищенню ефективності роботи.

Графічна концепція є невід'ємною складовою загальної концепції програмного забезпечення, оскільки визначає візуальні принципи представлення інформації, структуру елементів інтерфейсу користувача та логіку взаємодії між користувачем і системою. У контексті системи термометрії (рис.3.2), де передбачено моніторинг температурного стану об'єктів з високою точністю та в реальному часі, графічна концепція відіграє вирішальну роль у забезпеченні зручності експлуатації, оперативного реагування та інтуїтивного сприйняття даних [13].

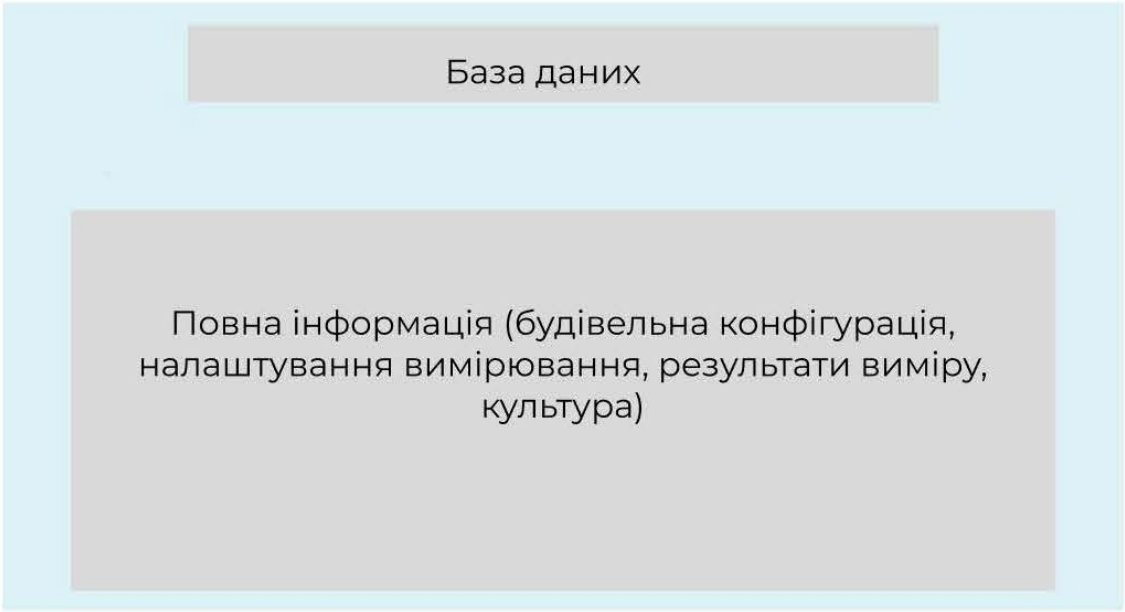


Рисунок 3.2 – Макет вікна «База даних»

Згідно з принципами UX/UI-дизайну, графічна концепція повинна враховувати специфіку цільової аудиторії — операторів елеваторів, інженерів з автоматизації, технічного персоналу. Вона повинна мінімізувати когнітивне навантаження та помилки користувачів, забезпечуючи інтуїтивно зрозумілу взаємодію з системою. Особливо важливим є використання візуальної аналітики,

для відображення змін температури в часі або в просторі по секціях елеватора (рис. 3.3).

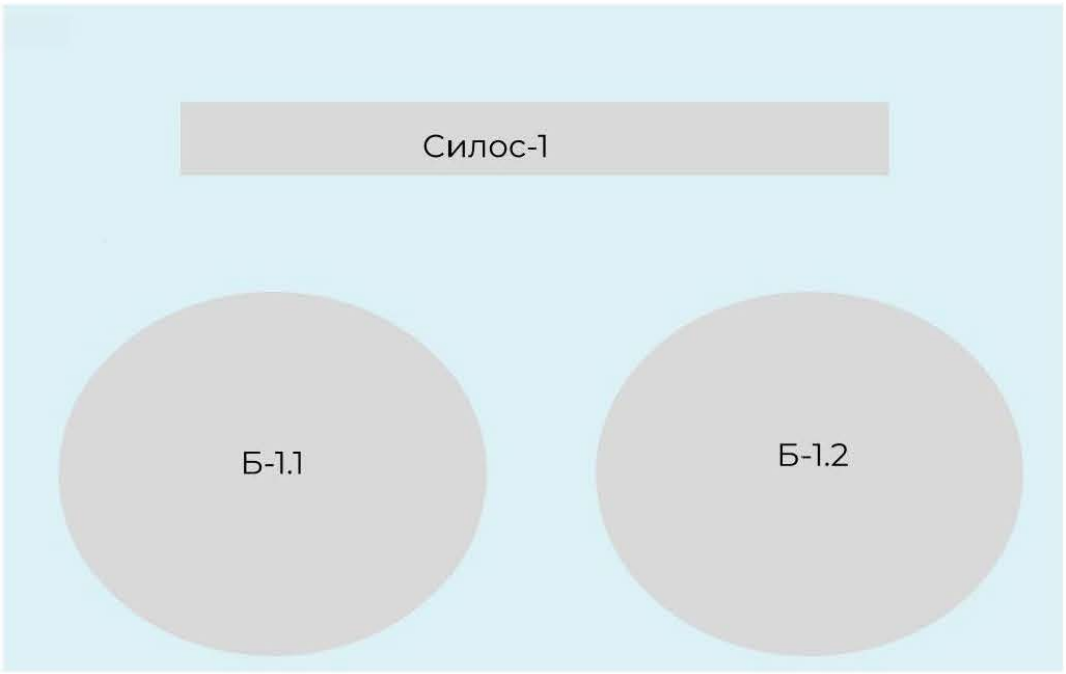


Рисунок 3.3 – Макет вікна «Силос»

Графічна концепція програмного забезпечення формується на етапі проєктування інтерфейсу користувача у вигляді макетів або інтерактивних прототипів, створених за допомогою спеціалізованих інструментів UI/UX-дизайну, таких як Figma, Adobe XD, Sketch тощо. На цьому етапі визначаються та узгоджуються ключові компоненти інтерфейсу, включаючи логіку взаємодії користувача з системою, структуру навігації, стилістичне оформлення та інформаційну ієрархію. Ці елементи є критично важливими для забезпечення інтуїтивності, доступності та ефективності користування програмним продуктом.

Після затвердження графічної концепції здійснюється її інтеграція у програмне середовище з урахуванням обраної архітектури, цільової платформи (десктопна, веб, мобільна) та технологій реалізації. Такий підхід забезпечує не лише візуальну цілісність продукту, а й адаптивність до різних сценаріїв використання.

Завдяки якісно спроєктованій графічній концепції програмне забезпечення отримує не лише функціональну завершеність, але й високий рівень ергономічності та користувацької зручності, що є особливо критичним для систем моніторингу та контролю, де швидкість і точність візуального сприйняття інформації безпосередньо впливають на ефективність прийняття рішень, мінімізацію людського фактору, збереження матеріальних ресурсів і забезпечення стабільності технологічних процесів (рис. 3.4).

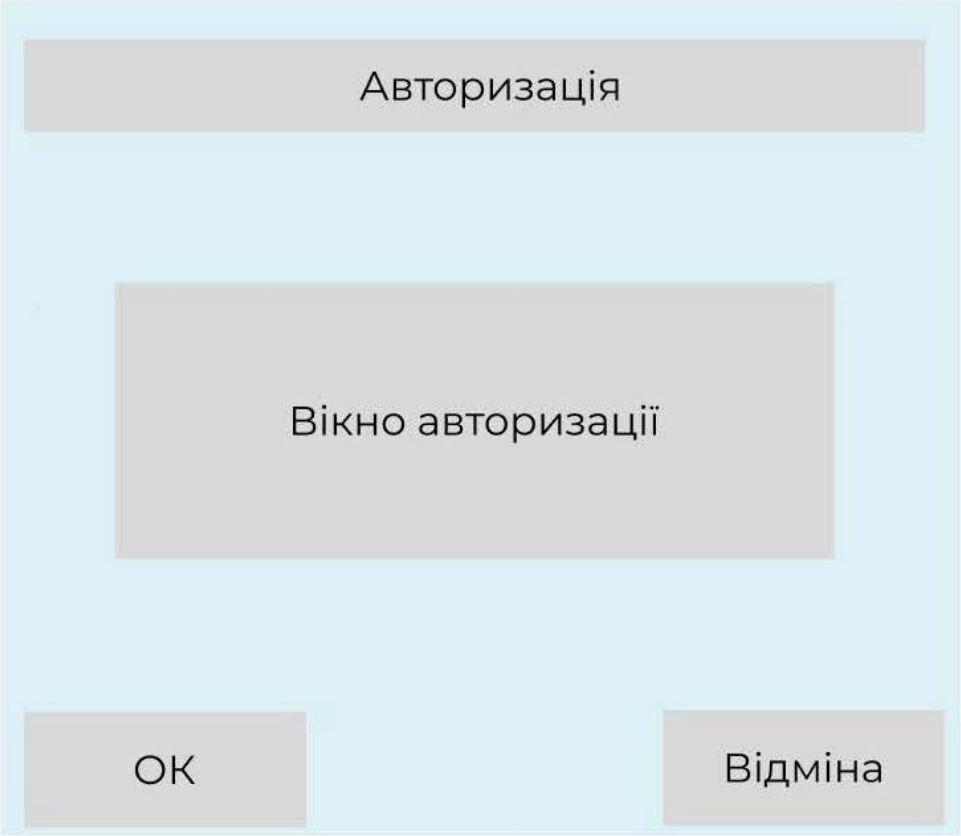


Рисунок 3.4 – Макет вікна «Авторизація»

У системі термометрії, де передбачено моніторинг температурного стану об'єктів з високою точністю та в реальному часі, функціональний модуль вимірювання температури є ключовим компонентом, що забезпечує оперативне отримання даних із сенсорних пристроїв, обробку показників та їхнє представлення в зручній формі для користувача.

На представленому прототипі інтерфейсу (рис.3.5) реалізовано базову структуру взаємодії користувача з системою:

Рисунок 3.5 – Макет вікна «Вимірювання»

У складі програмного забезпечення системи термометрії суттєве значення мають функціональні модулі, призначені для документування, зберігання, обробки та архівування результатів температурного моніторингу. Їх наявність є необхідною умовою для забезпечення простежуваності технологічних процесів, уніфікації звітної документації відповідно до стандартів якості, а також для реалізації механізмів регламентованого та нерегламентованого аудиту температурного режиму зберігання сільськогосподарської продукції (рис. 3.6).

Такі модулі забезпечують автоматизовану генерацію звітів, фіксацію критичних температурних подій із часовими мітками, а також підтримку історичних баз даних, що дозволяє здійснювати ретроспективний аналіз ефективності функціонування систем охолодження та вентиляції. Крім того, реалізація функцій експорту даних у стандартизованих форматах полегшує інтеграцію з зовнішніми інформаційними системами, включно з державними платформами контролю якості та внутрішніми модулями управління аграрними підприємствами.

Інформаційна відкритість, забезпечена цими модулями, сприяє підвищенню прозорості технологічних процесів, забезпеченню відповідності чинним нормативам у сфері зберігання продукції, а також підвищенню рівня довіри з боку партнерів, інспекційних органів та споживачів. Отже, функціональні модулі документування є інтегральною складовою цифрової інфраструктури сучасного елеваторного господарства та вагомим чинником підвищення операційної ефективності аграрного сектору.

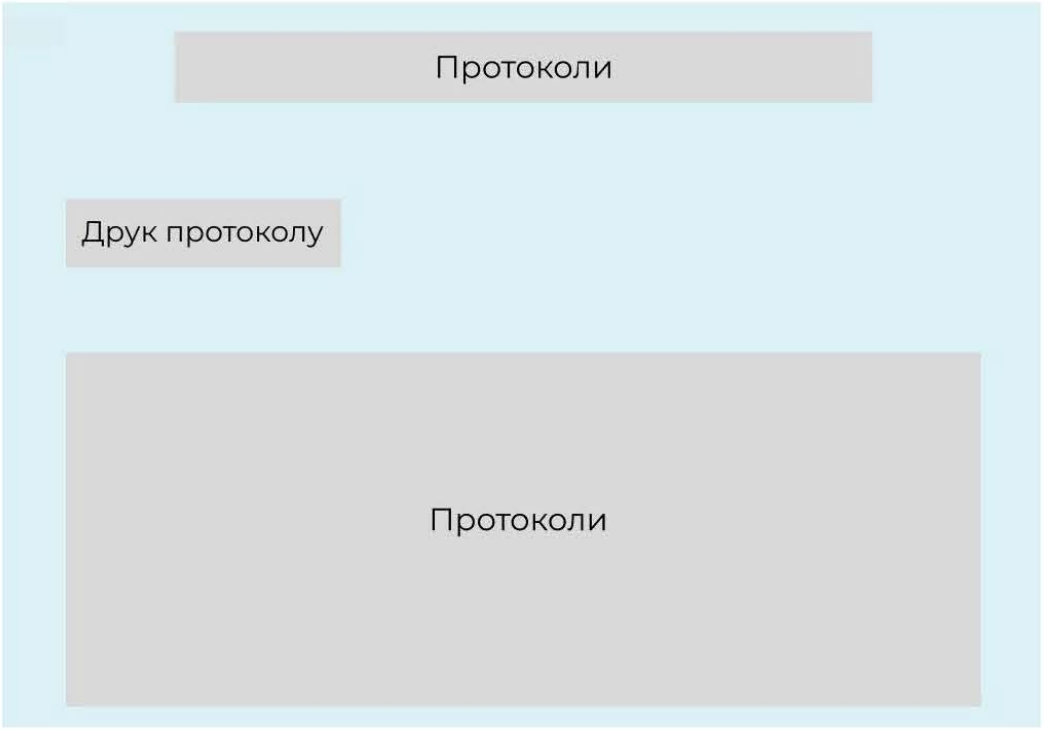


Рисунок 3.6 – Макет вікна «Протоколи»

У структурі програмного комплексу термометрії ключову роль відіграє модуль візуалізації температурних даних, який забезпечує інтерпретацію показників через кольорову картограму, де кожен спектр відповідає певному діапазону температур (рис. 3.7). Цей інструмент не лише спрощує інтерактивну роботу з даними, але й дозволяє оперативно аналізувати просторовий розподіл теплових параметрів у моніторинговому середовищі. Завдяки інтуїтивно зрозумілій кольоровій легенді користувач може ідентифікувати аномалії (локальні перегріви або гіпотермічні зони), що є вирішальним фактором для

підтримки оптимальних умов зберігання матеріалів та продукції. Впровадження подібних візуальних механізмів підвищує ефективність превентивного контролю, зменшуючи ймовірність технологічних ризиків, пов’язаних із температурними відхиленнями [15].

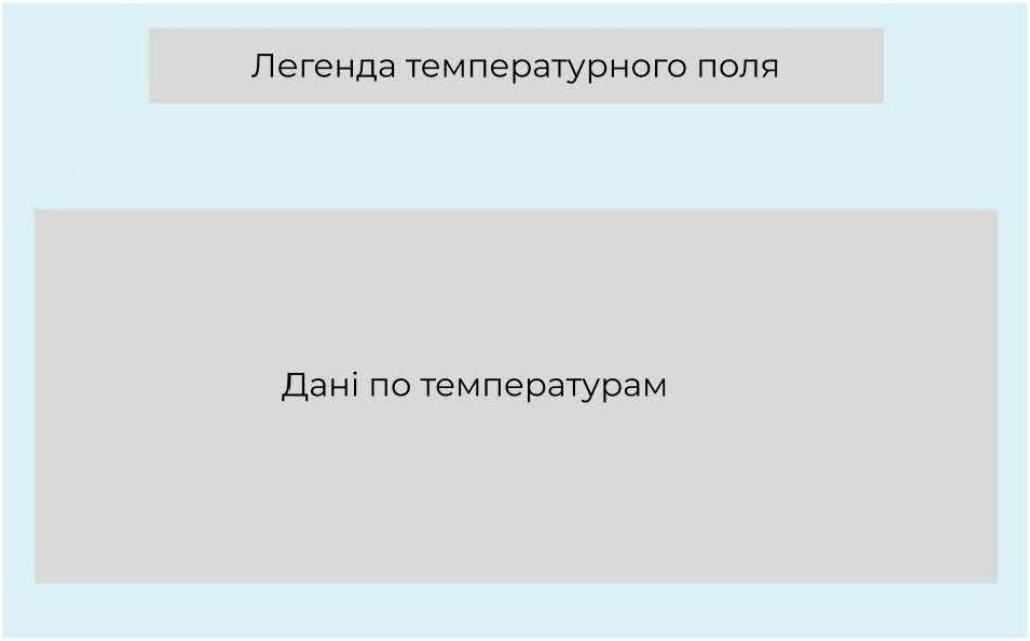


Рисунок 3.7 – Макет вікна «Протоколи»

3.4. Розробка програми та її опис

- Назва програми: Термометрія.
- Призначення програми: Програма комп'ютерної системи термометрії призначена для запуску дистанційного вимірювання температури у силосах зберігання зерна, а також збереження та перегляду результатів вимірювань у вигляді протоколу вимірювання, температурного поля або графіки.
- Мова програмування: C#.
- Логічна структура програми: Програма розроблена для роботи в середовищі Windows та має стандартний для цієї системи інтерфейс. Вікно Програми містить головне меню, через яке доступні всі основні команди Програми, панелі інструментів, що містять найчастіше використовувані команди, робочої області, зміст якої залежить від виконуваних команд

оператора, та рядки стану, що відображає стан найважливіших параметрів роботи програми [16].

- Вхідні дані:

- Показники температури у реальному часі;
- Частота оновлення даних;
- Ідентифікатор датчика для прив'язки до конкретної зони моніторингу;
- Додаткові параметри.
- Діапазони норми/аварії для температури (мінімальні та максимальні граничні значення);
- Налаштування кольорової шкали для візуалізації;
- Географічне розташування датчиків.

- Вихідні дані:

- Теплова шкала зміни температури у часі.
- Інтерактивна легенда з кольоровою шкалою.
- Звіти та статистика:
- Середні, мінімальні та максимальні значення за вибраний період.
- Автоматичні повідомлення для відповідальних осіб.
- PDF-звіти для документації.

- Програмні засоби, необхідні для реалізації програми: Windows Forms, pictureBox, richTextBox

- Технічні засоби, необхідні для реалізації програми: Персональний комп'ютер, Visual Studio, SQL Server.

- Вимоги до пристрою користувача програми: Встановлений Windows 10 та Visual Studio 2019

3.5. Інструкція користувачу

Програма комп'ютерної системи термометрії призначена для запуску дистанційного вимірювання температури у силосах зберігання зерна, а також

збереження та перегляду результатів вимірювань у вигляді протоколу вимірювання, температурного поля або графіки.

Програма розроблена для роботи в середовищі Windows та має стандартний для цієї системи інтерфейс. Вікно Програми містить головне меню, через яке доступні всі основні команди Програми, панелі інструментів, що містять найчастіше використовувані команди, робочої області, зміст якої залежить від виконуваних команд оператора, та рядки стану, що відображає стан найважливіших параметрів роботи програми. Відразу після запуску Програма має такий вигляд (рис. 3.8).

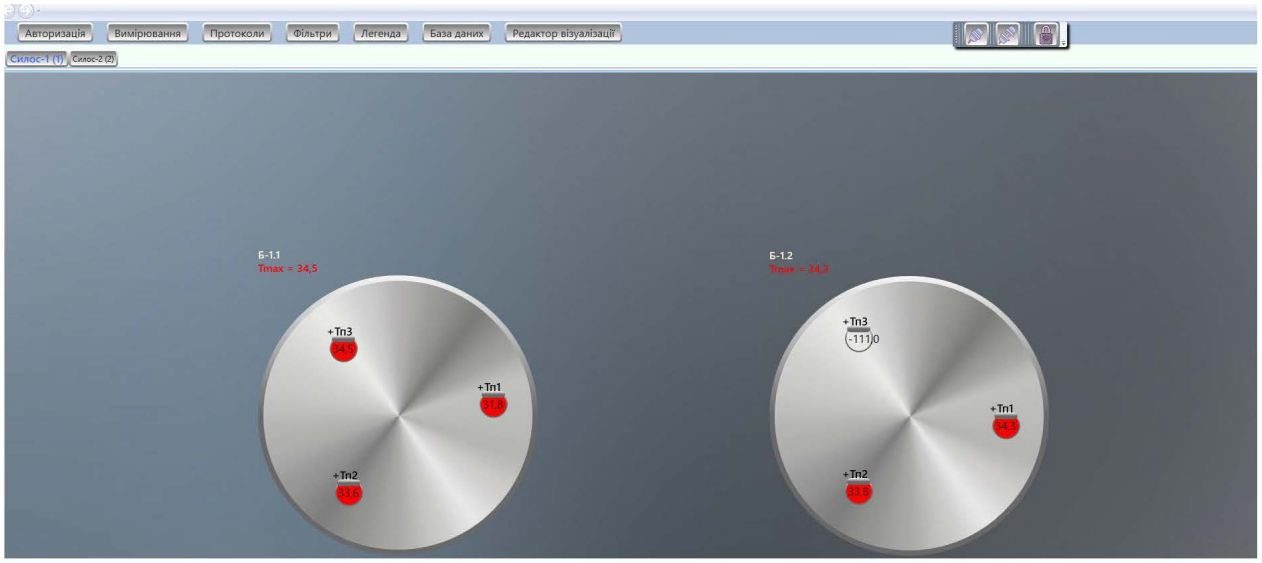


Рисунок 3.8 – Головне вікно програми

Для забезпечення повного доступу до функціональних можливостей програмного забезпечення системи термометрії користувач повинен пройти процедуру авторизації. Цей етап є обов'язковим для ідентифікації та перевірки прав доступу, що дозволяє реалізувати диференційований підхід до управління доступом залежно від ролі користувача. Авторизація не лише підвищує рівень інформаційної безпеки, але й запобігає несанкціонованому втручанню в критичні компоненти системи, зокрема зміні параметрів моніторингу, редагуванню протоколів чи маніпуляціям із даними вимірювань (рис. 3.9).

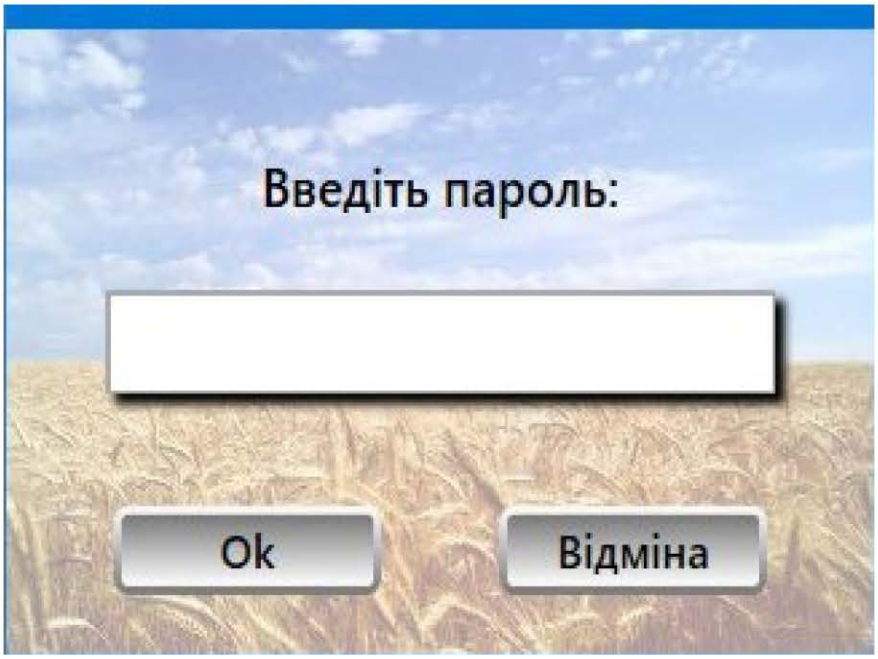


Рисунок 3.9 – Вікно «Авторизація»

Після успішного проходження процедури авторизації користувач отримує доступ до функціоналу програмного забезпечення, зокрема можливість ініціювати процес температурного моніторингу. Для цього необхідно перейти до інтерфейсного модуля «Вимірювання», де реалізовано запуск процедури зчитування температурних показників у режимі реального часу. Всі отримані значення автоматично фіксуються у відповідному протоколі з прив'язкою до точних часових міток та дати здійснення виміру. Такий підхід забезпечує безперервний контроль температурного режиму, дозволяє формувати хронологічну історію спостережень та є необхідною умовою для подальшого аналізу динаміки змін у контрольованому середовищі (рис. 3.10).

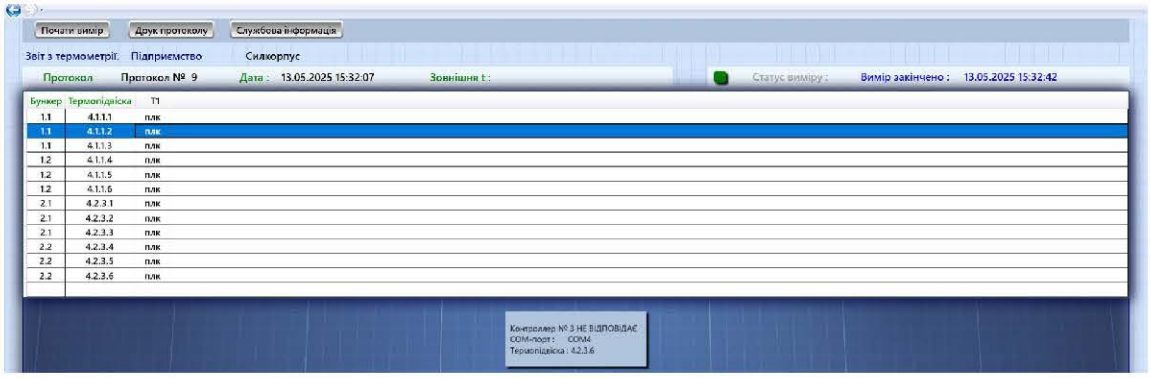


Рисунок 3.10 – Вікно «Вимірювання»

У програмному забезпеченні системи термометрії реалізовано механізм збереження результатів усіх температурних вимірювань (за винятком окремих службових режимів) у модулі «Протоколи». Цей функціональний блок забезпечує централізоване зберігання, систематизацію та доступ до історичних даних, що є критично важливим для забезпечення простежуваності процесів моніторингу. Для перегляду раніше зафіксованих даних передбачено спеціальний режим візуалізації протоколів. У цьому режимі на екран виводиться структурований список усіх записів, здійснених під час попередніх сеансів вимірювань. Кожен протокол містить назву, точну дату й час здійснення виміру, значення зовнішньої температури, зафіксоване у відповідний момент, а також кількість записаних температурних значень. Такий підхід забезпечує ефективну навігацію користувача по історичних даних і створює умови для аналітичної обробки температурної інформації. Крім того, збережені протоколи можуть використовуватись як джерело для формування статистичних звітів та виявлення довготривалих тенденцій у зміні температурного режиму, що особливо актуально для контролю умов зберігання сільськогосподарської продукції чи інших чутливих до температури об’єктів. Таким чином, модуль «Протоколи» виступає не лише засобом збереження історії вимірювань, але і важливим елементом інформаційно-аналітичної підтримки прийняття рішень у рамках системи термометричного моніторингу (рис. 3.11).

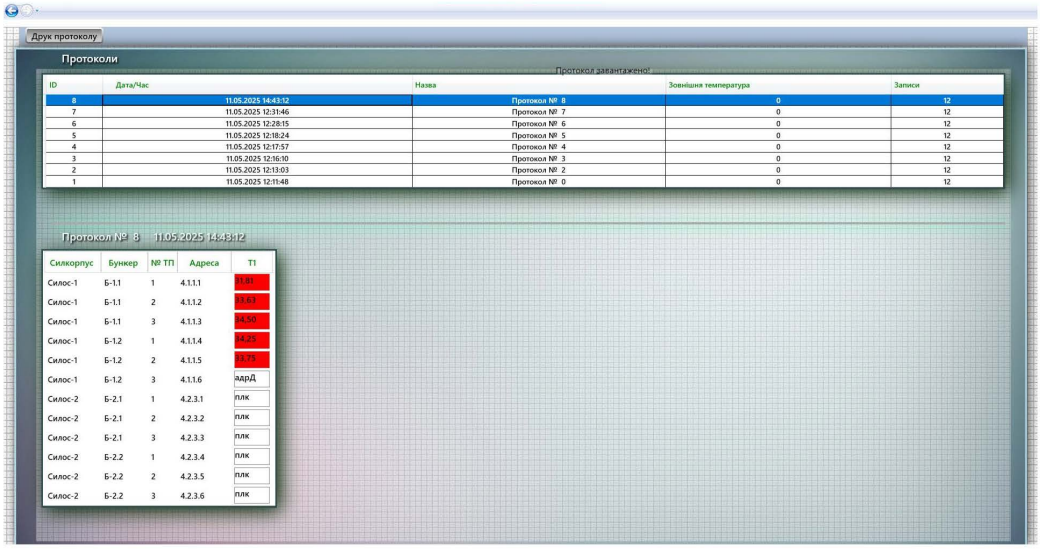


Рисунок 3.11 – Вікно «Протоколи»

У вікні «Легенда температурного поля» представлено інтерпретацію температурних даних за допомогою кольорової шкали, що дозволяє користувачу швидко ідентифікувати критичні зони в межах контрольованого середовища. Зокрема, вказуються значення мінімальної та максимальної температур, які автоматично фіксуються системою під час моніторингу. Кожному діапазону температур відповідає певний колір, що забезпечує візуальну диференціацію температурних зон (рис. 3.12).

Рисунок 3.12 – Вікно «Легенда температурного поля»

Такий підхід до візуалізації дозволяє оперативно виявляти аномалії температурного режиму (зони перегріву або переохолодження), що має суттєве значення для забезпечення нормативних умов зберігання, транспортування або експлуатації температурно-чутливих об'єктів. Графічна легенда служить орієнтиром для інтерпретації температурного розподілу, підвищуючи інформативність та ефективність прийняття рішень у процесі моніторингу [17].

Крім того, у складі програмного забезпечення системи термометрії передбачено функціональний модуль «База даних», який забезпечує централізоване зберігання та управління ключовою інформацією, необхідною для ефективного моніторингу температурного режиму. У цьому вікні зосереджено структуровані відомості про конфігурацію силосних ємностей, характеристики зернових культур, що зберігаються, а також параметри проведених вимірювань (рис. 3.13).

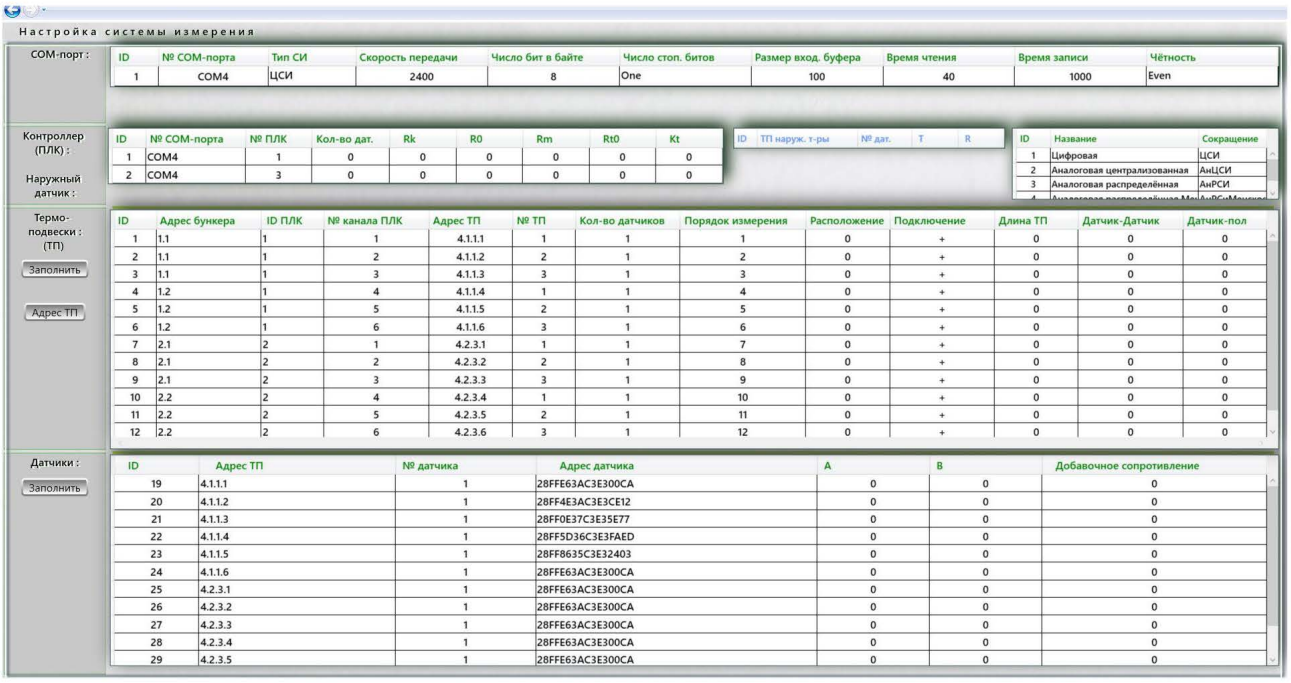
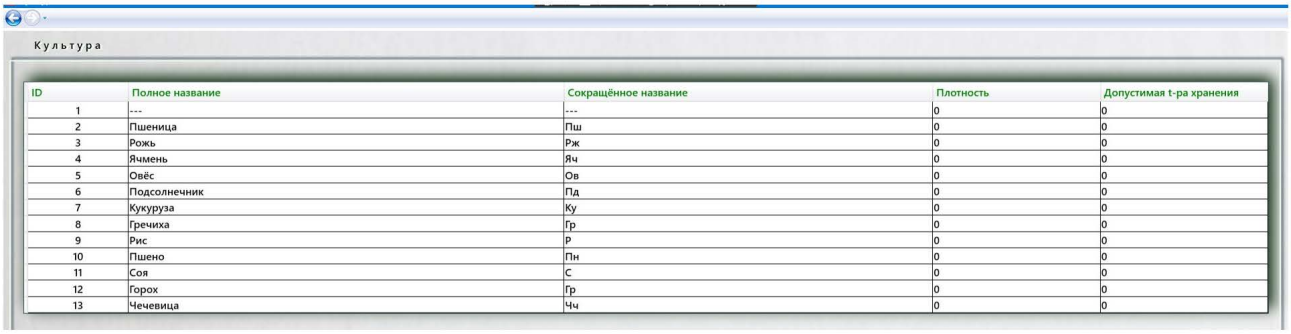


Рисунок 3.13 – Вікно «База даних. Система вимірювання»

Інтерфейс модуля дозволяє фахівцям виконувати перегляд, оновлення та редагування відповідних записів із дотриманням вимог достовірності та актуальності даних. Така функціональність є критично важливою для адаптації системи до змін у структурі зберігання, типах культур або параметрах технологічного процесу. Можливість ручного коригування інформації підвищує гнучкість системи, забезпечує її відповідність поточним умовам експлуатації та сприяє підвищенню загальної ефективності контролю за температурним режимом.

Функціонал редагування надає можливість авторизованим користувачам змінювати існуючі записи, додавати нові об’єкти зберігання або оновлювати інформацію про врожай, що надходить. Така модифікація даних виконується з обов’язковою фіксацією змін, що є важливим елементом забезпечення відповідності стандартам безпеки аграрного зберігання(рис. 3.13).



ID	Полное название	Сокращённое название	Плотность	Допустимая t-ра хранения
1	---	---	0	0
2	Пшеница	Пш	0	0
3	Рожь	Рж	0	0
4	Ячмень	Яч	0	0
5	Овёс	Ов	0	0
6	Подсолнечник	Пд	0	0
7	Кукуруза	Ку	0	0
8	Гречиха	Гр	0	0
9	Рис	Р	0	0
10	Пшено	Пн	0	0
11	Соя	С	0	0
12	Горох	Гр	0	0
13	Чечевица	Чч	0	0

Рисунок 3.14 – Вікно «База даних. Культура»

3.6. Аналіз результатів комп'ютерної реалізації програми та пропозиції щодо вдосконалення програмного продукту

Розроблений програмний продукт «Термометрія» успішно виконує поставлені завдання з моніторингу температурного режиму зберігання сільськогосподарської продукції, забезпечуючи високу точність вимірювань, оперативність обробки інформації та зручність взаємодії з користувачем. У результаті комп'ютерної реалізації було протестовано ключові функціональні модулі, включаючи:

- модуль авторизації, що гарантує контроль доступу до системи та захист даних;
- модуль вимірювання, який забезпечує зчитування температури в реальному часі та автоматичну фіксацію показників із прив'язкою до дати та часу;
- інтерфейс протоколювання, що дозволяє зберігати, переглядати й друкувати результати вимірювань;
- систему візуалізації температурного поля, яка надає кольорову інтерпретацію даних з урахуванням граничних температур;
- модуль бази даних, у якому зберігається повна інформація про структуру силосів, типи зернових культур та відповідні агротехнічні параметри [18].

Аналіз функціонування програмного забезпечення засвідчив високу стабільність роботи та коректність обчислювальних алгоритмів. Завдяки наочній

графічній інтерпретації та структурованому представленню протоколів користувач отримує змогу оперативно реагувати на відхилення температурного режиму, що має критичне значення для збереження якості продукції.

Однак у ході експлуатаційного тестування було виявлено низку напрямів, які можуть бути покращені з метою підвищення ефективності та масштабованості системи:

1. Реалізація функцій автоматичного сповіщення (SMS/E-mail) при виявленні критичних температур або аварійних ситуацій, що дозволить оперативно інформувати відповідальних осіб.
2. Інтеграція з мобільними платформами для забезпечення віддаленого доступу до температурних даних у режимі реального часу.
3. Можливість побудови графіків змін температури з історичних даних, що забезпечить аналітичний інструментарій для виявлення довготривалих тенденцій.
4. Додавання багатомовного інтерфейсу для використання системи в міжнародному контексті.
5. Розширення модуля бази даних шляхом введення автоматичного імпорту даних з польових пристроїв (наприклад, через бездротові сенсори).

Узагальнюючи вищенаведене, можна стверджувати, що програмний продукт «Термометрія» є ефективним інструментом цифрового контролю температурного режиму зберігання сільськогосподарської продукції. Його подальший розвиток у напрямі автоматизації, мобільності та аналітики дозволить істотно підвищити рівень технологічної підтримки агропромислових підприємств [19].

3.7. Висновки до третього розділу

У ході розробки програмного забезпечення «Термометрія» було реалізовано ефективний інструмент для моніторингу температурного режиму в силосах зберігання зерна. Використання сучасних технологій, дозволило

створити стабільну, масштабовану та зручну для користувача систему, яка відповідає основним вимогам до автоматизації процесів температурного контролю.

Основні досягнення:

Реалізація ключових функціональних модулів:

- модуль авторизації для забезпечення безпеки даних;
- модуль вимірювання температури в реальному часі;
- система протоколювання та зберігання історії вимірювань;
- візуалізація температурного поля за допомогою кольорової шкали;
- модуль бази даних для управління інформацією про силоси та зернові культури.

Зручний інтерфейс користувача:

- інтуїтивна графічна оболонка з елементами візуалізації даних;
- можливість генерування звітів у форматі PDF;
- автоматичне фіксування температурних показників із часовими мітками.

Технологічні переваги:

- використання C#, як основної мови програмування забезпечило високу продуктивність, безпеку коду та легкість підтримки;
- інтеграція з Windows Forms дозволила створити зручний десктопний інтерфейс;
- можливість подальшого масштабування за рахунок підтримки .NET Core/.NET 5+.

Перспективи вдосконалення:

Розширення функціоналу:

- додавання SMS/E-mail сповіщень при критичних температурних відхиленнях;
- інтеграція з мобільними додатками для віддаленого моніторингу;
- впровадження аналітичних інструментів (графіки трендів, статистичний аналіз).

Покращення користувацького досвіду:

- додавання багатомовного інтерфейсу;
- оптимізація роботи з великими масивами даних;
- підтримка хмарного зберігання для резервування даних.

Загальний висновок:

Програмний продукт «Система термометрії» успішно вирішує поставлені завдання, забезпечуючи точний контроль температурного режиму та зручний інструментарій для аналізу даних. Подальший розвиток системи дозволить ще більше підвищити її ефективність, автоматизацію та інтеграцію з сучасними технологіями, що робить її перспективним рішенням для агропромислового сектору.

ВИСНОВКИ

Зернова галузь виступає ключовою ланкою агропромислового комплексу України, забезпечуючи не лише продовольчу безпеку держави, але й формуючи значну частку валютних надходжень через експорт аграрної продукції. Її стратегічне значення обумовлює потребу в системному модернізаційному підході, спрямованому на подолання існуючих структурних, технологічних та організаційних обмежень.

Станом на сьогодні зернова галузь стикається з низкою критичних проблем, серед яких варто виокремити:

- застарілість матеріально-технічної бази, що не відповідає вимогам сучасного агровиробництва;
- недостатній рівень фінансового забезпечення, який обмежує здатність підприємств до впровадження інновацій;
- негативний вплив кліматичних змін у поєднанні з неефективним використанням земельних ресурсів;
- низьку ефективність зберігання продукції через технічну недосконалість елеваторних систем.

Елеватори як центральний компонент зернової інфраструктури відіграють визначальну роль у ланцюгу створення доданої вартості — від післязбиральної обробки до підготовки зерна до експорту. Їхня ефективність значною мірою залежить від рівня автоматизації процесів, наявності цифрових систем моніторингу та управління, а також мінімізації впливу людського чинника. Проте більшість елеваторних господарств функціонують на основі застарілих технологій із ручним управлінням, що знижує загальну продуктивність та підвищує ризики втрат.

Цифрова трансформація зернової галузі має стати основою її подальшого розвитку. До перспективних технологічних напрямів належать:

- впровадження системи управління агропідприємствами для централізованого управління ресурсами;
- використання пристроїв і систем для контролю параметрів зберігання;
- застосування хмарних платформ і аналітики Big Data для прийняття управлінських рішень;
- автоматизація термометрії як ключового елемента контролю якості продукції.

Водночас цифровізація стикається з рядом бар'єрів: високою вартістю впровадження, низьким рівнем цифрової грамотності серед персоналу, недосконалістю інфраструктури сільських територій та складністю інтеграції новітніх технологій у традиційні виробничі процеси.

У межах даного дослідження було розроблено програмний продукт – автоматизовану систему термометрії, призначену для безперервного моніторингу температурного режиму в зерносховищах. Система реалізована на базі мови C# з використанням модульної архітектури та підтримує такі функції:

- збирання даних у реальному часі;
- візуалізація температурного поля за допомогою кольорових індикаторів;
- генерація звітів у форматі PDF із фіксацією часових міток;
- ведення бази даних температурних значень.

Аналіз результатів тестування виявив низку напрямів для подальшого вдосконалення:

1. Функціональні розширення – впровадження сповіщень при виявленні критичних температур, можливість роботи з мобільними додатками, формування графіків історичних змін.
2. Покращення UX/UI – додавання багатомовного інтерфейсу, оптимізація роботи з великими обсягами даних, реалізація резервного зберігання в хмарі.
3. Технічна модернізація – використання бездротових сенсорних технологій (LoRaWAN, Bluetooth), інтеграція з системами активного вентилювання.

Система термометрії має потенціал до масштабування та може бути

інтегрована в ширший контекст цифрового агровиробництва. Її використання сприятиме:

- мінімізації втрат зерна внаслідок псування чи самозайманння;
- зменшенню витрат на енергоносії через оптимізацію вентиляції;
- забезпеченню прозорості та достовірності даних щодо умов зберігання.

Найбільш перспективними напрямами подальшого наукового дослідження є:

- створення інтегрованої платформи для комплексного моніторингу параметрів зерносховищ;
- застосування технологій машинного навчання для прогнозування температурних коливань і запобігання ризикам псування;
- розробка модулів інтеграції з державними реєстрами якості та безпечності продукції для забезпечення простежуваності по всьому ланцюгу постачання.

Цифрова трансформація зернової галузі України виступає ключовим чинником забезпечення її сталого розвитку та підвищення конкурентоспроможності як на внутрішньому, так і на глобальному агропродовольчому ринку. Інтеграція сучасних цифрових рішень, зокрема автоматизованих систем моніторингу, таких як система термометрії, відкриває нові можливості для оптимізації виробничих процесів, підвищення точності контролю за умовами зберігання та мінімізації втрат продукції. Ефективне впровадження таких інноваційних технологій сприяє не лише підвищенню якості зерна, а й раціоналізації використання ресурсів, зниженню операційних витрат та забезпеченню прозорості управлінських рішень. Водночас досягнення стійкого прогресу в цифровізації аграрного сектору потребує скоординованих дій з боку держави, бізнесу та наукової спільноти. Така міжсекторальна взаємодія має формувати сприятливе інституційне середовище, забезпечувати фінансову та освітню підтримку, а також сприяти впровадженню відкритих стандартів і платформ для масштабованого використання цифрових технологій у сільському господарстві.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сайт Elevatorist [Електронний ресурс] – Режим доступу: https://elevatorist.com/

2. Сайт АПК Інформ [Електронний ресурс] – Режим доступу: https://www.apk-inform.com/uk

3. Сайт Emerson [Електронний ресурс] – Режим доступу: https://www.emerson.com/en-ua

4. Присяжнюк О.І., Роїк М.В., Сінченко В.М., Черняк М.О., Маляренко О.А., Кононюк Н.О. Цифрові технології в агрономії – від теорії до практики [Текст] – Вінниця.: Інститут біоенергетичних культур і цукрових буряків НААН, 2024. – 211 с.

5. І. В. Арістова, В. І. Курило, О. Ю. Калугін; за заг. ред. Арістової І. В. Впровадження інформаційно-комунікаційних технологій в аграрний сектор економіки України: організаційно-правовий аспект [Текст] – Київ.: Інститут інформації, безпеки і права Національної академії правових наук України, 2014. – 193 с.

6. Сайт KSE [Електронний ресурс] – Режим доступу: https://kse.ua/ua/

7. Н. М. Горобець, Д. О. Хомякова, Д. О. Стариковська, Перспективи використання цифрових технологій в діяльності аграрних підприємств [Текст] – Дніпро.: Дніпровський державний аграрно-економічний університет, 2021. – 92

8. Шацька З.Я. Особливості впровадження інформаційних технологій в аграрному секторі України [Текст] – Київ.: Київський національний університет технологій та дизайну, 2022. – 5 с.

9. Сайт Qalight [Електронний ресурс] – Режим доступу: https://qalight.ua/baza-znan/

10. Сайт vseosvita.ua [Електронний ресурс] – Режим доступу: https://vseosvita.ua/library/embed/000h80-9259.doc.html

11. Сайт Adakurutma [Електронний ресурс] – Режим доступу: https://www.adakurutma.com.tr/en/

12. Соколова, Н. О., Бешта, Л. В., & Бешта, Д. О. (2024). ВІЗУАЛІЗАЦІЯ ІНФОРМАЦІЇ: РОЗКІШ ЧИ НЕОБХІДНІСТЬ?. Електротехнічні та інформаційні системи, (105), 10–13. https://doi.org/10.32782/EIS/2024-105-2

13. Firsov, O. D., & Marischuk, O. V. (2024). FUZZY SYSTEM FOR ANALYSIS OF BUSINESS RULES OF THE SUBJECT AREA DURING DATABASE CREATION. Systems and Technologies, 68(2), 48-62. https://doi.org/10.32782/2521-6643-2024-2-68.6

14. Погребняк А.В., Яковенко В.О., Клим В.Я., Яковенко Т.Ю. (2024). ПРОБЛЕМИ ІНТЕГРАЦІЇ ІОТ В КОМП'ЮТЕРНІ МЕРЕЖІ. Системи та технології , 68 (2), 32-38. https://doi.org/10.32782/2521-6643-2024-2-68.4

15. Чупілко Т.А. Актуальні проблеми високоефективної обробки даних. Моделювання показників за допомогою мови програмування Python / Т.А. Чупілко // Актуальні напрями розвитку технічного та виробничого потенціалу національної економіки: монографія/ за ред. В.О.Пінчук, Г.С. Прокудіна – Дніпро: Пороги.- 2021. - 536 с.

16. Ю.В. Ульяновська, Т.М. Рудянова, В.В. Костенко, А.О. Олещук, Булгакова О.Ф. Використання технологій Augmented Reality в програмному забезпеченні для інтерактивної візуалізації тривимірних об'єктів на промислових підприємствах. Системи та технології, №1 (61), 2021. - С.71-84..

17. EC-Council University [Електронний ресурс] – Режим доступу: https://www.eccu.edu/blog/cybersecurity/fundamentals-of-information-security/

18. ZenGRC, [Електронний ресурс] – Режим доступу: https://www.zengrc.com/blog/what-are-the-principles-of-information-security/

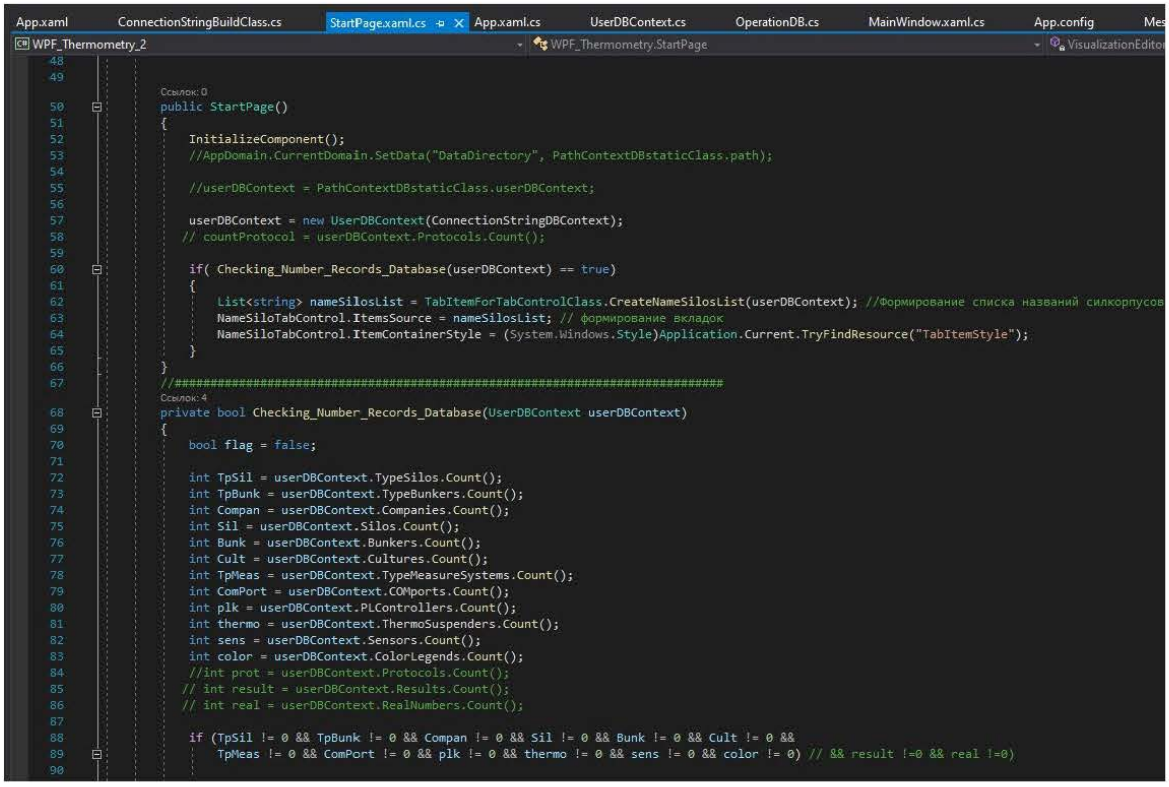
19. Infosecuritybasics, [Електронний ресурс] – Режим доступу: https://www.cisa.gov/sites/default/files/publications/infosecuritybasics.pdf

ДОДАТОК А

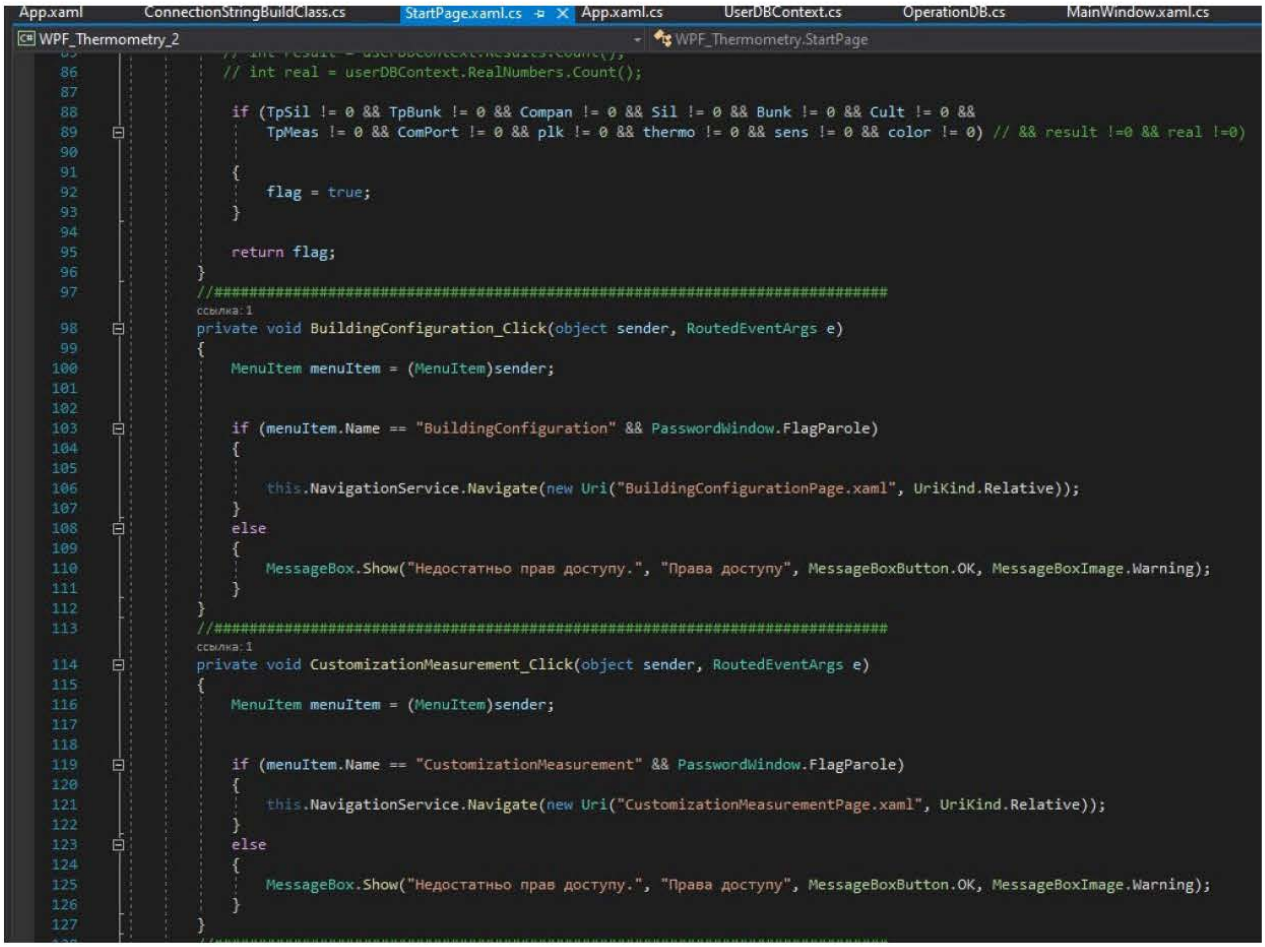
ЛІСТІНГ ПРОГРАМИ

App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs

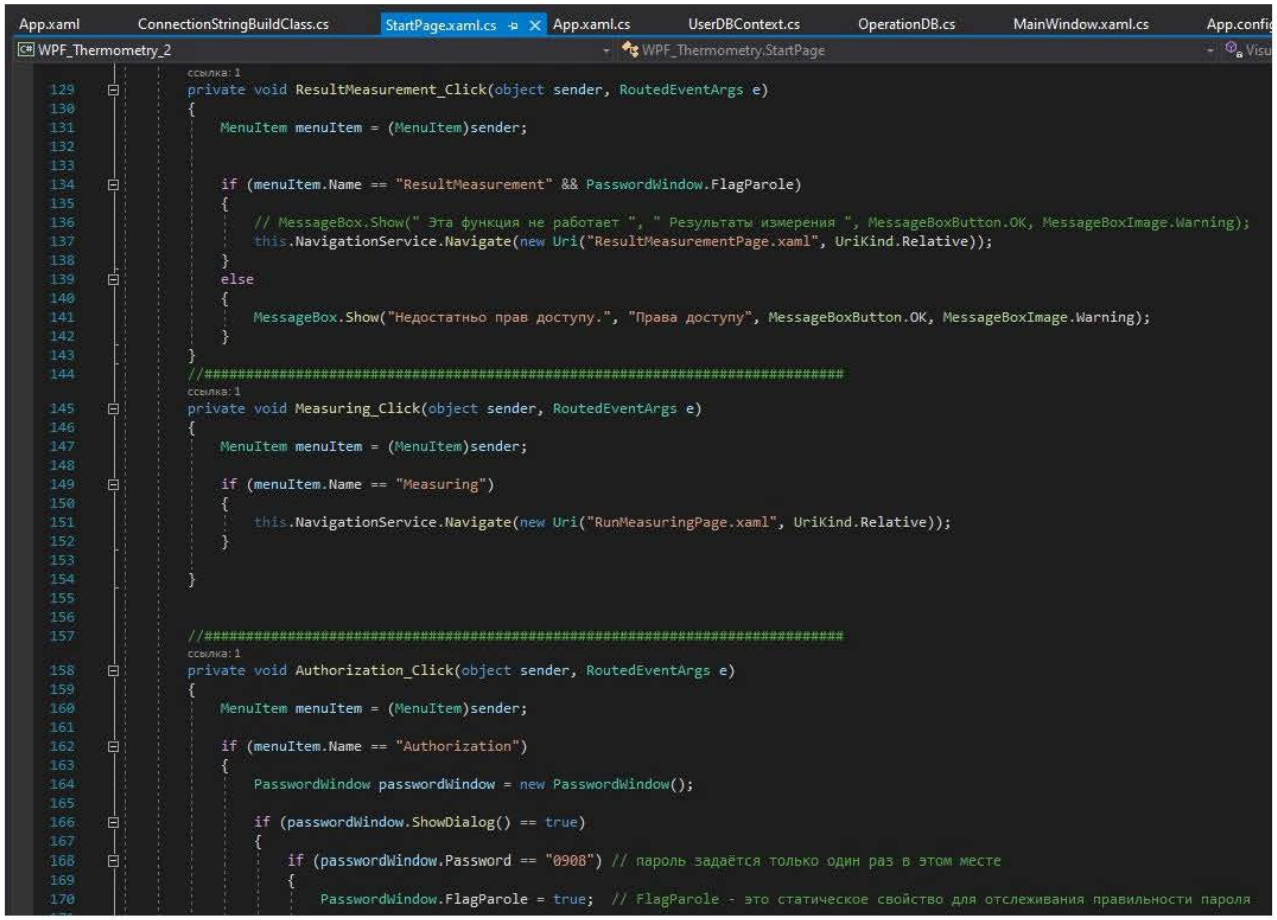
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Data.Entity;
using System.Reflection;
using System.Collections;
using System.ComponentModel;
using WPF_Thermometry.Model;
using WPF_Thermometry.Measuring;
using WPF_Thermometry.VisualEditor;
using System.Data;
using System.Diagnostics;


namespace WPF_Thermometry
{
    /// <summary>
    /// Логика взаимодействия для StartPage.xaml
    /// </summary>
    ссылки
    public partial class StartPage : Page
    {
        readonly UserDBContext userDBContext;
        readonly string ConnectionStringDBContext = ConnectionStringBuildClass.BuildConnectionString();

        int OldSelectedIndexTabControl;
        int countProtocol;

        // ThermoSuspender TSparam;
        //List<ThermoSuspender> TSparamList = new List<ThermoSuspender>();
        string adressTS;
        ThermoSuspender _TSParametr ;

        //ModelThermoSuspender ModelTS;
```



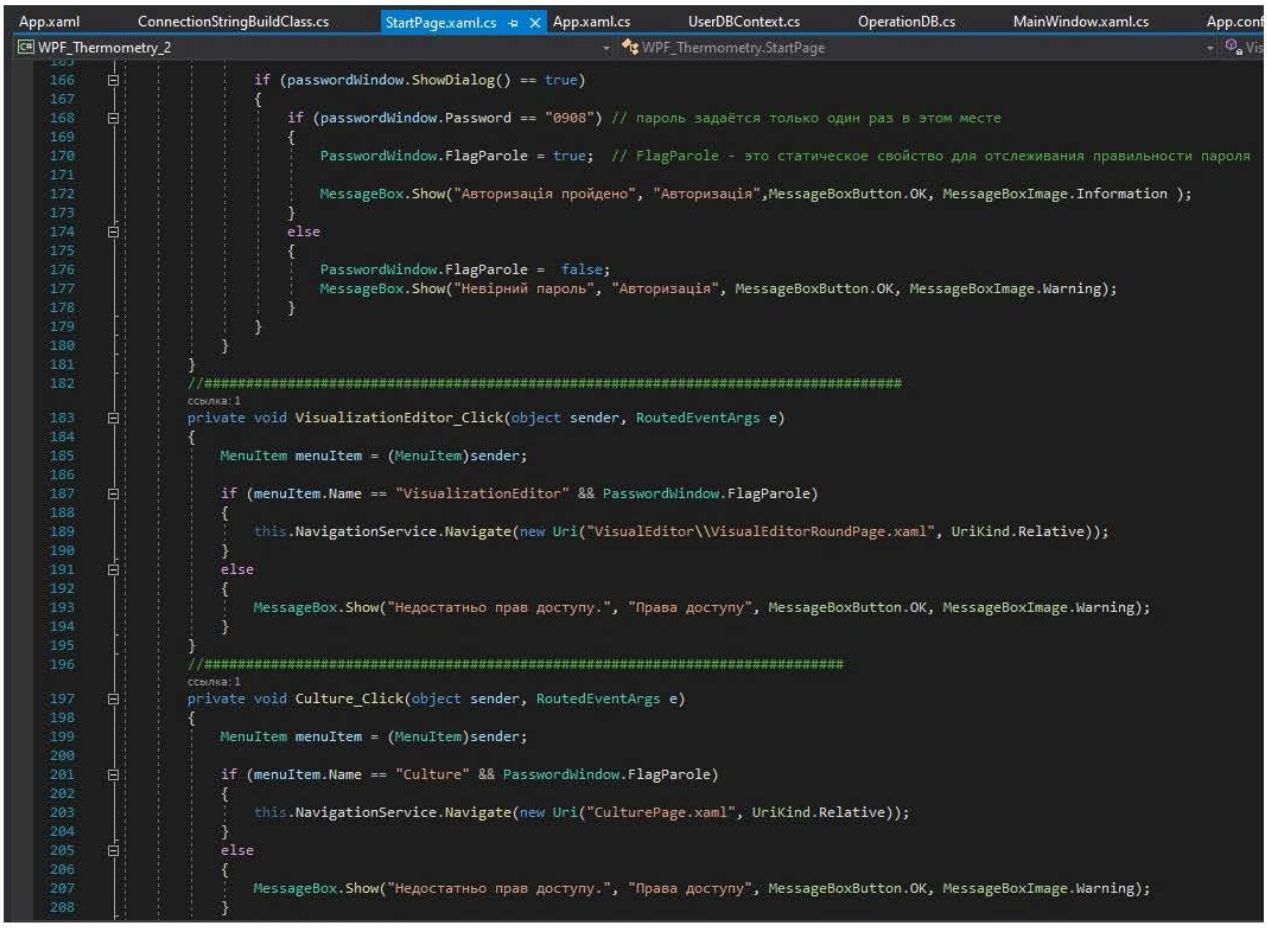
```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs | App.config
WPF_Thermometry_2 | WPF_Thermometry.StartPage

48
49
        Ссылок: 0
50      public StartPage()
51      {
52          InitializeComponent();
53          //AppDomain.CurrentDomain.SetData("DataDirectory", PathContextDBstaticClass.path);
54
55          //userDBContext = PathContextDBstaticClass.userDBContext;
56
57          userDBContext = new UserDBContext(ConnectionStringDBContext);
58         // countProtocol = userDBContext.Protocols.Count();
59
60          if( Checking_Number_Records_Database(userDBContext) == true)
61          {
62              List<string> nameSilosList = TabItemForTabControlClass.CreateNameSilosList(userDBContext); //Формирование списка названий силкорпусов
63              NameSiloTabControl.ItemsSource = nameSilosList; // формирование вкладок
64              NameSiloTabControl.ItemContainerStyle = (System.Windows.Style)Application.Current.TryFindResource("TabItemStyle");
65          }
66      }
67      //#########################################################################
        Ссылок: 4
68      private bool Checking_Number_Records_Database(UserDBContext userDBContext)
69      {
70          bool flag = false;
71
72          int TpSil = userDBContext.TypeSilos.Count();
73          int TpBunk = userDBContext.TypeBunkers.Count();
74          int Compan = userDBContext.Companies.Count();
75          int Sil = userDBContext.Silos.Count();
76          int Bunk = userDBContext.Bunkers.Count();
77          int Cult = userDBContext.Cultures.Count();
78          int TpMeas = userDBContext.TypeMeasureSystems.Count();
79          int ComPort = userDBContext.COMports.Count();
80          int plk = userDBContext.PLControllers.Count();
81          int thermo = userDBContext.ThermoSuspenders.Count();
82          int sens = userDBContext.Sensors.Count();
83          int color = userDBContext.ColorLegends.Count();
84          //int prot = userDBContext.Protocols.Count();
85         // int result = userDBContext.Results.Count();
86         // int real = userDBContext.RealNumbers.Count();
87
88          if (TpSil != 0 && TpBunk != 0 && Compan != 0 && Sil != 0 && Bunk != 0 && Cult != 0 &&
89              TpMeas != 0 && ComPort != 0 && plk != 0 && thermo != 0 && sens != 0 && color != 0) // && result !=0 && real !=0)
90
```



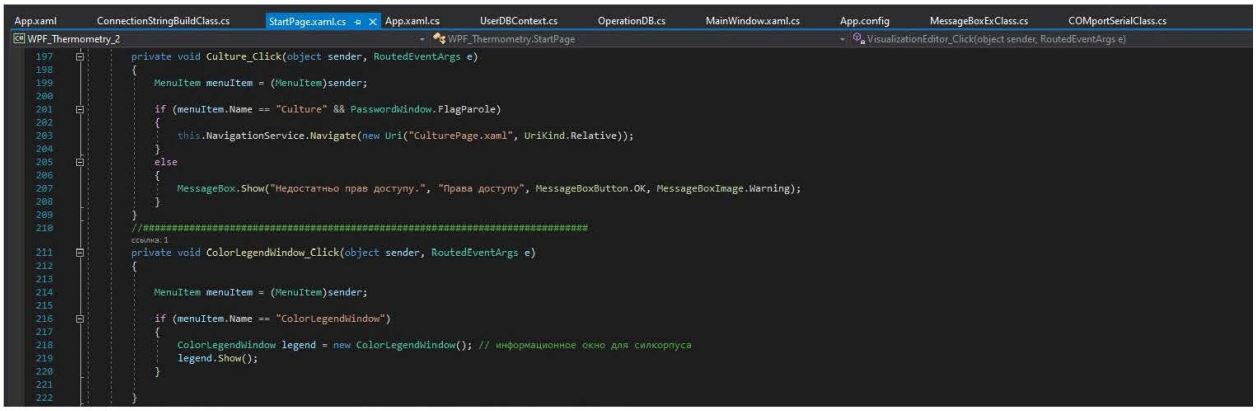
```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs
WPF_Thermometry_2 | WPF_Thermometry.StartPage

86          // int real = userDBContext.RealNumbers.Count();
87
88          if (TpSil != 0 && TpBunk != 0 && Compan != 0 && Sil != 0 && Bunk != 0 && Cult != 0 &&
89              TpMeas != 0 && ComPort != 0 && plk != 0 && thermo != 0 && sens != 0 && color != 0) // && result !=0 && real !=0)
90
91          {
92              flag = true;
93          }
94
95          return flag;
96      }
97      //#########################################################################
        ссылка: 1
98      private void BuildingConfiguration_Click(object sender, RoutedEventArgs e)
99      {
100         MenuItem menuItem = (MenuItem)sender;
101
102
103         if (menuItem.Name == "BuildingConfiguration" && PasswordWindow.FlagParole)
104         {
105
106             this.NavigationService.Navigate(new Uri("BuildingConfigurationPage.xaml", UriKind.Relative));
107         }
108         else
109         {
110             MessageBox.Show("Недостатньо прав доступу.", "Права доступу", MessageBoxButton.OK, MessageBoxImage.Warning);
111         }
112     }
113     //#########################################################################
        ссылка: 1
114     private void CustomizationMeasurement_Click(object sender, RoutedEventArgs e)
115     {
116         MenuItem menuItem = (MenuItem)sender;
117
118
119         if (menuItem.Name == "CustomizationMeasurement" && PasswordWindow.FlagParole)
120         {
121             this.NavigationService.Navigate(new Uri("CustomizationMeasurementPage.xaml", UriKind.Relative));
122         }
123         else
124         {
125             MessageBox.Show("Недостатньо прав доступу.", "Права доступу", MessageBoxButton.OK, MessageBoxImage.Warning);
126         }
127     }
```

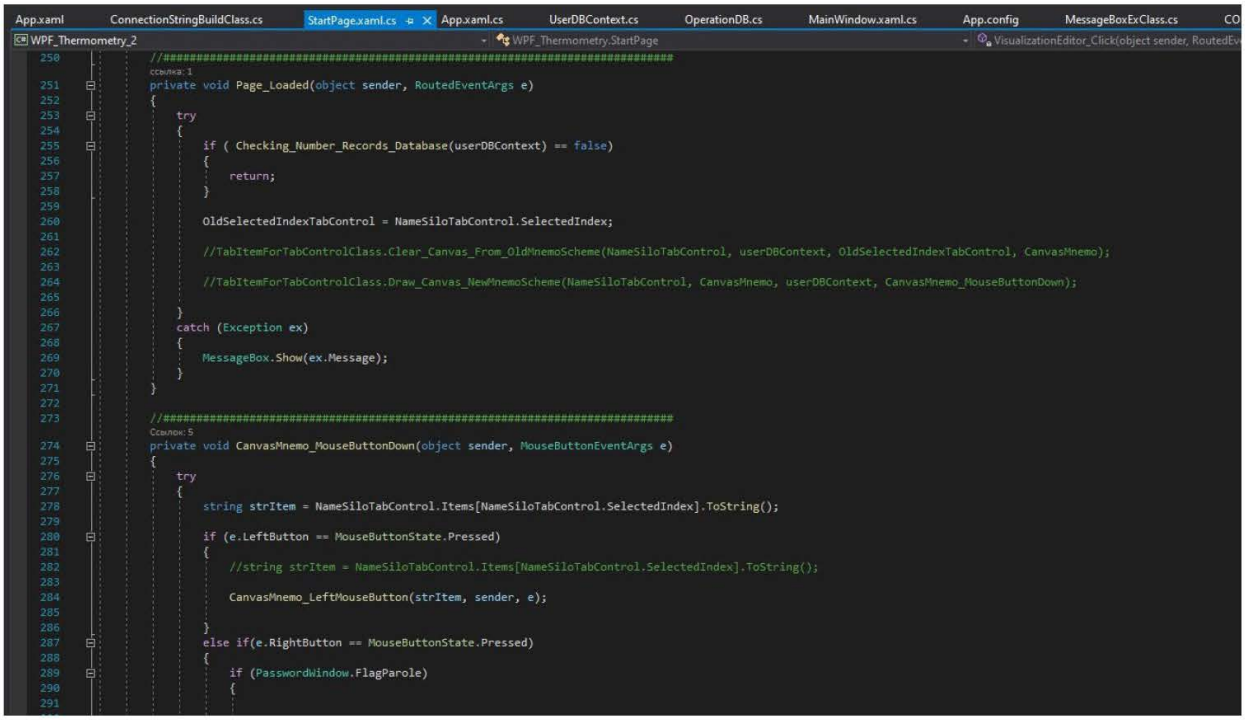


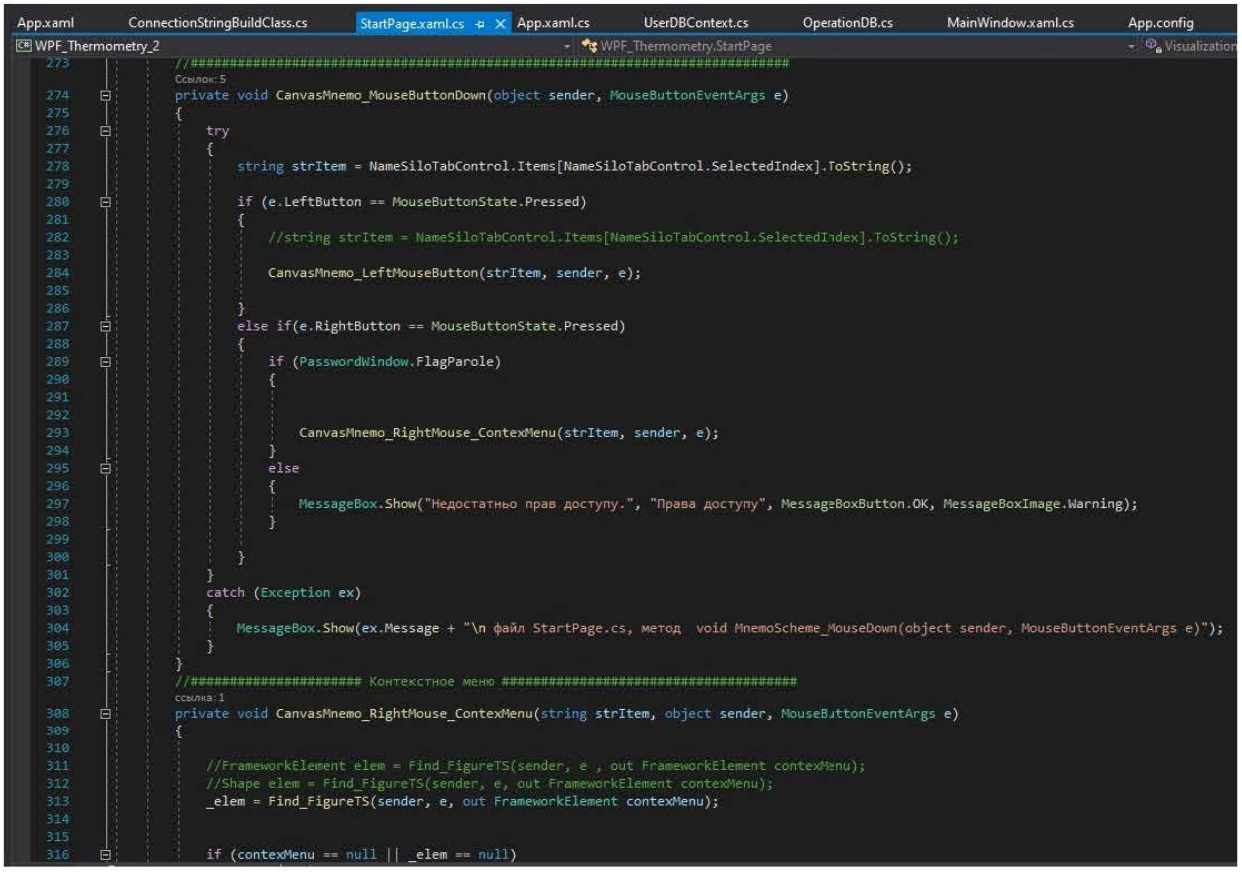
```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs | App.config
WPF_Thermometry_2 | WPF_Thermometry.StartPage

ссылка: 1
private void ResultMeasurement_Click(object sender, RoutedEventArgs e)
{
    MenuItem menuItem = (MenuItem)sender;


    if (menuItem.Name == "ResultMeasurement" && PasswordWindow.FlagParole)
    {
        // MessageBox.Show(" Эта функция не работает ", " Результаты измерения ", MessageBoxButton.OK, MessageBoxImage.Warning);
        this.NavigationService.Navigate(new Uri("ResultMeasurementPage.xaml", UriKind.Relative));
    }
    else
    {
        MessageBox.Show("Недостатньо прав доступу.", "Права доступу", MessageBoxButton.OK, MessageBoxImage.Warning);
    }
}
//#########################################################################
ссылка: 1
private void Measuring_Click(object sender, RoutedEventArgs e)
{
    MenuItem menuItem = (MenuItem)sender;

    if (menuItem.Name == "Measuring")
    {
        this.NavigationService.Navigate(new Uri("RunMeasuringPage.xaml", UriKind.Relative));
    }

}


//#########################################################################
ссылка: 1
private void Authorization_Click(object sender, RoutedEventArgs e)
{
    MenuItem menuItem = (MenuItem)sender;

    if (menuItem.Name == "Authorization")
    {
        PasswordWindow passwordWindow = new PasswordWindow();

        if (passwordWindow.ShowDialog() == true)
        {
            if (passwordWindow.Password == "0908") // пароль задаётся только один раз в этом месте
            {
                PasswordWindow.FlagParole = true;  // FlagParole - это статическое свойство для отслеживания правильности пароля
```



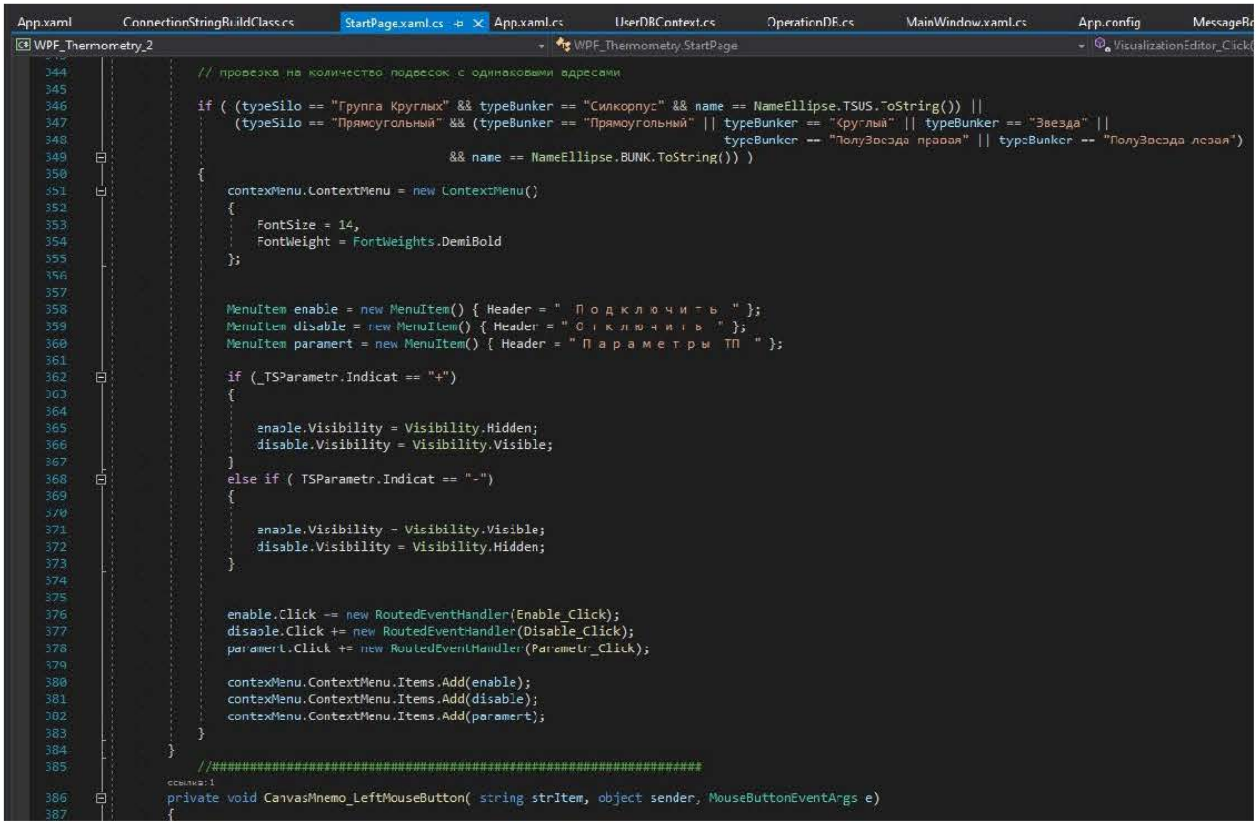
```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs | App.conf
WPF_Thermometry_2 | WPF_Thermometry.StartPage

        if (passwordWindow.ShowDialog() == true)
        {
            if (passwordWindow.Password == "0908") // пароль задаётся только один раз в этом месте
            {
                PasswordWindow.FlagParole = true;  // FlagParole - это статическое свойство для отслеживания правильности пароля

                MessageBox.Show("Авторизація пройдено", "Авторизація",MessageBoxButton.OK, MessageBoxImage.Information );
            }
            else
            {
                PasswordWindow.FlagParole =  false;
                MessageBox.Show("Невірний пароль", "Авторизація", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
        }
    }
}
//#########################################################################
ссылка: 1
private void VisualizationEditor_Click(object sender, RoutedEventArgs e)
{
    MenuItem menuItem = (MenuItem)sender;

    if (menuItem.Name == "VisualizationEditor" && PasswordWindow.FlagParole)
    {
        this.NavigationService.Navigate(new Uri("VisualEditor\\VisualEditorRoundPage.xaml", UriKind.Relative));
    }
    else
    {
        MessageBox.Show("Недостатньо прав доступу.", "Права доступу", MessageBoxButton.OK, MessageBoxImage.Warning);
    }
}
//#########################################################################
ссылка: 1
private void Culture_Click(object sender, RoutedEventArgs e)
{
    MenuItem menuItem = (MenuItem)sender;

    if (menuItem.Name == "Culture" && PasswordWindow.FlagParole)
    {
        this.NavigationService.Navigate(new Uri("CulturePage.xaml", UriKind.Relative));
    }
    else
    {
        MessageBox.Show("Недостатньо прав доступу.", "Права доступу", MessageBoxButton.OK, MessageBoxImage.Warning);
    }
```


App.xaml
ConnectionStringBuildClass.cs
StartPage.xaml.cs
App.xaml.cs
UserDBContext.cs
OperationDB.cs
MainWindow.xaml.cs
App.config
MessageBoxExClass.cs
COMportSerialClass.cs
WPF_Thermometry_2
WPF_Thermometry.StartPage
VisualizationEditor_Click(object sender, RoutedEventArgs e)
197 private void Culture_Click(object sender, RoutedEventArgs e)
198 {
199 MenuItem menuItem = (MenuItem)sender;
200
201 if (menuItem.Name == "Culture" && PasswordWindow.FlagParole)
202 {
203 this.NavigationService.Navigate(new Uri("CulturePage.xaml", UriKind.Relative));
204 }
205 else
206 {
207 MessageBox.Show("Недостатньо прав доступу.", "Права доступу", MessageBoxButton.OK, MessageBoxImage.Warning);
208 }
209 }
210 //###
ссылка: 1
211 private void ColorLegendWindow_Click(object sender, RoutedEventArgs e)
212 {
213
214 MenuItem menuItem = (MenuItem)sender;
215
216 if (menuItem.Name == "ColorLegendWindow")
217 {
218 ColorLegendWindow legend = new ColorLegendWindow(); // информационное окно для силкорпуса
219 legend.Show();
220 }
221
222 }

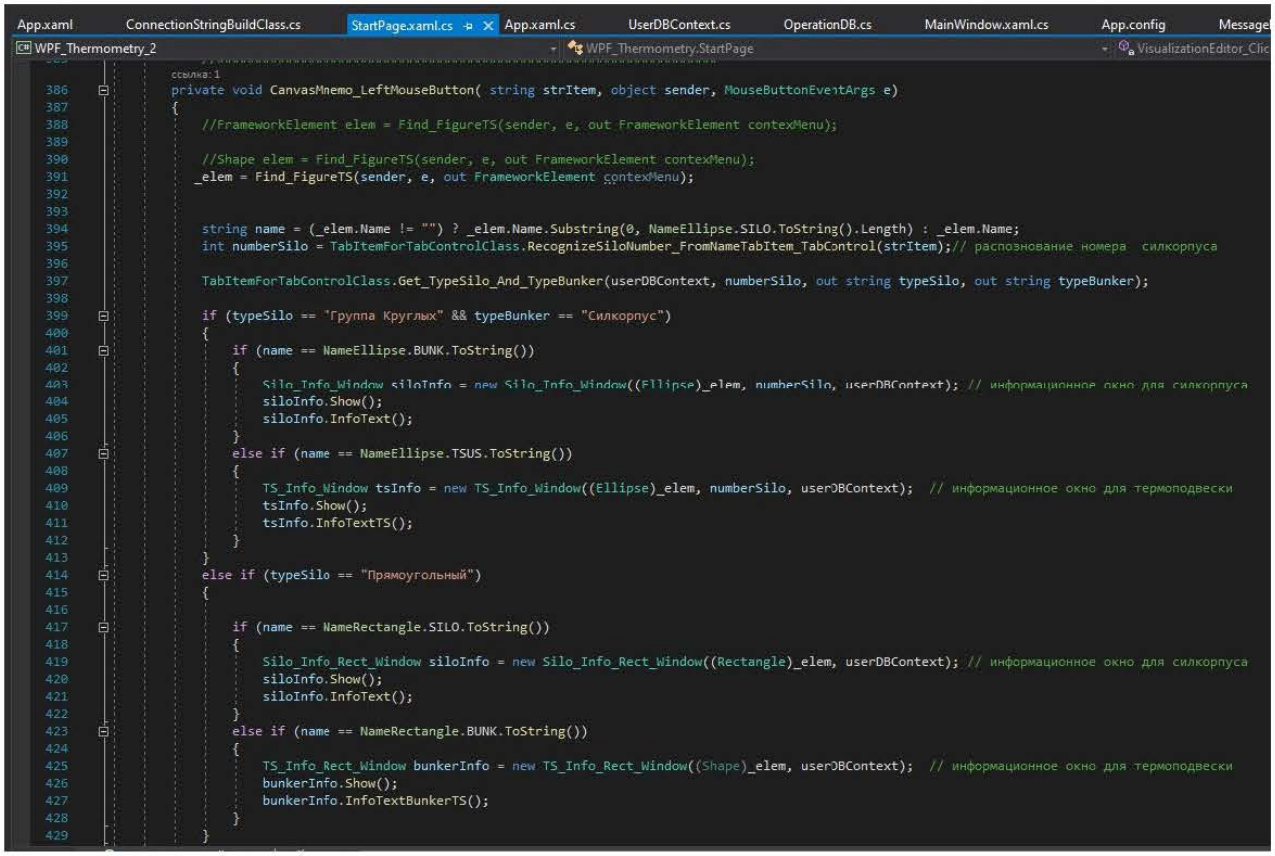

App.xaml
ConnectionStringBuildClass.cs
StartPage.xaml.cs
App.xaml.cs
UserDBContext.cs
OperationDB.cs
MainWindow.xaml.cs
App.config
MessageBoxExClass.cs
WPF_Thermometry_2
WPF_Thermometry.StartPage
250 //###
ссылка: 1
251 private void Page_Loaded(object sender, RoutedEventArgs e)
252 {
253 try
254 {
255 if (Checking_Number_Records_Database(userDBContext) == false)
256 {
257 return;
258 }
259
260 OldSelectedIndexTabControl = NameSiloTabControl.SelectedIndex;
261
262 //TabItemForTabControlClass.Clear_Canvas_From_OldMnemoScheme(NameSiloTabControl, userDBContext, OldSelectedIndexTabControl, CanvasMnemo);
263
264 //TabItemForTabControlClass.Draw_Canvas_NewMnemoScheme(NameSiloTabControl, CanvasMnemo, userDBContext, CanvasMnemo_MouseButtonDown);
265
266 }
267 catch (Exception ex)
268 {
269 MessageBox.Show(ex.Message);
270 }
271 }
272
273 //###
Ссылок: 5
274 private void CanvasMnemo_MouseButtonDown(object sender, MouseButtonEventArgs e)
275 {
276 try
277 {
278 string strItem = NameSiloTabControl.Items[NameSiloTabControl.SelectedIndex].ToString();
279
280 if (e.LeftButton == MouseButtonState.Pressed)
281 {
282 //string strItem = NameSiloTabControl.Items[NameSiloTabControl.SelectedIndex].ToString();
283
284 CanvasMnemo_LeftMouseButton(strItem, sender, e);
285
286 }
287 else if(e.RightButton == MouseButtonState.Pressed)
288 {
289 if (PasswordWindow.FlagParole)
290 {
291



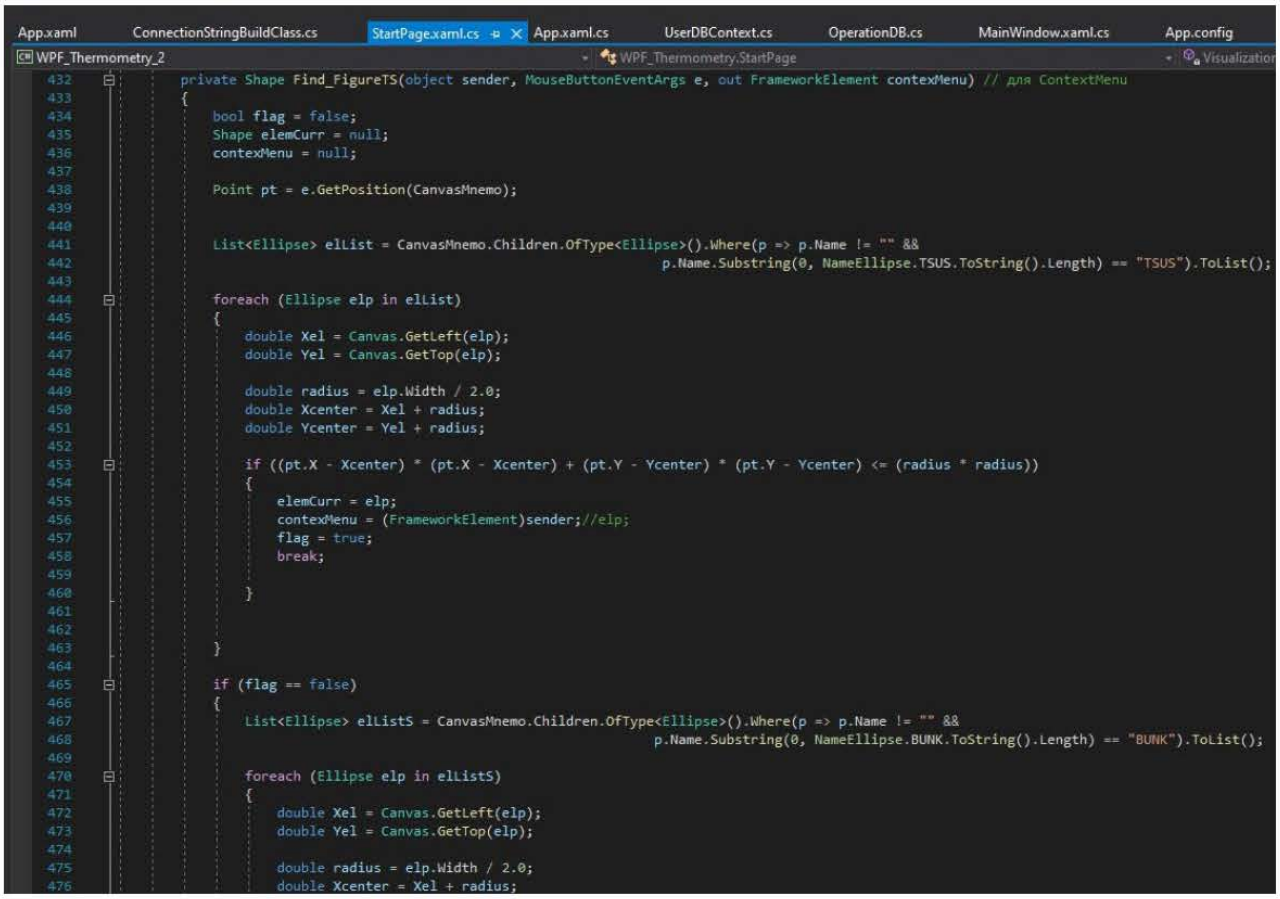
```
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.xaml.cs  App.config
WPF_Thermometry_2  WPF_Thermometry.StartPage  Visualization
273         //#########################################################################
            Ссылок: 5
274         private void CanvasMnemo_MouseButtonDown(object sender, MouseButtonEventArgs e)
275         {
276             try
277             {
278                 string strItem = NameSiloTabControl.Items[NameSiloTabControl.SelectedIndex].ToString();
279
280                 if (e.LeftButton == MouseButtonState.Pressed)
281                 {
282                     //string strItem = NameSiloTabControl.Items[NameSiloTabControl.SelectedIndex].ToString();
283
284                     CanvasMnemo_LeftMouseButton(strItem, sender, e);
285
286                 }
287                 else if(e.RightButton == MouseButtonState.Pressed)
288                 {
289                     if (PasswordWindow.FlagParole)
290                     {
291
292
293                         CanvasMnemo_RightMouse_ContexMenu(strItem, sender, e);
294                     }
295                     else
296                     {
297                         MessageBox.Show("Недостатньо прав доступу.", "Права доступу", MessageBoxButton.OK, MessageBoxImage.Warning);
298                     }
299
300                 }
301             }
302             catch (Exception ex)
303             {
304                 MessageBox.Show(ex.Message + "\n файл StartPage.cs, метод  void MnemoScheme_MouseDown(object sender, MouseButtonEventArgs e)");
305             }
306         }
307         //###################### Контекстное меню ###################################
            ссылка: 1
308         private void CanvasMnemo_RightMouse_ContexMenu(string strItem, object sender, MouseButtonEventArgs e)
309         {
310
311             //FrameworkElement elem = Find_FigureTS(sender, e , out FrameworkElement contexMenu);
312             //Shape elem = Find_FigureTS(sender, e, out FrameworkElement contexMenu);
313             _elem = Find_FigureTS(sender, e, out FrameworkElement contexMenu);
314
315
316             if (contexMenu == null || _elem == null)
```



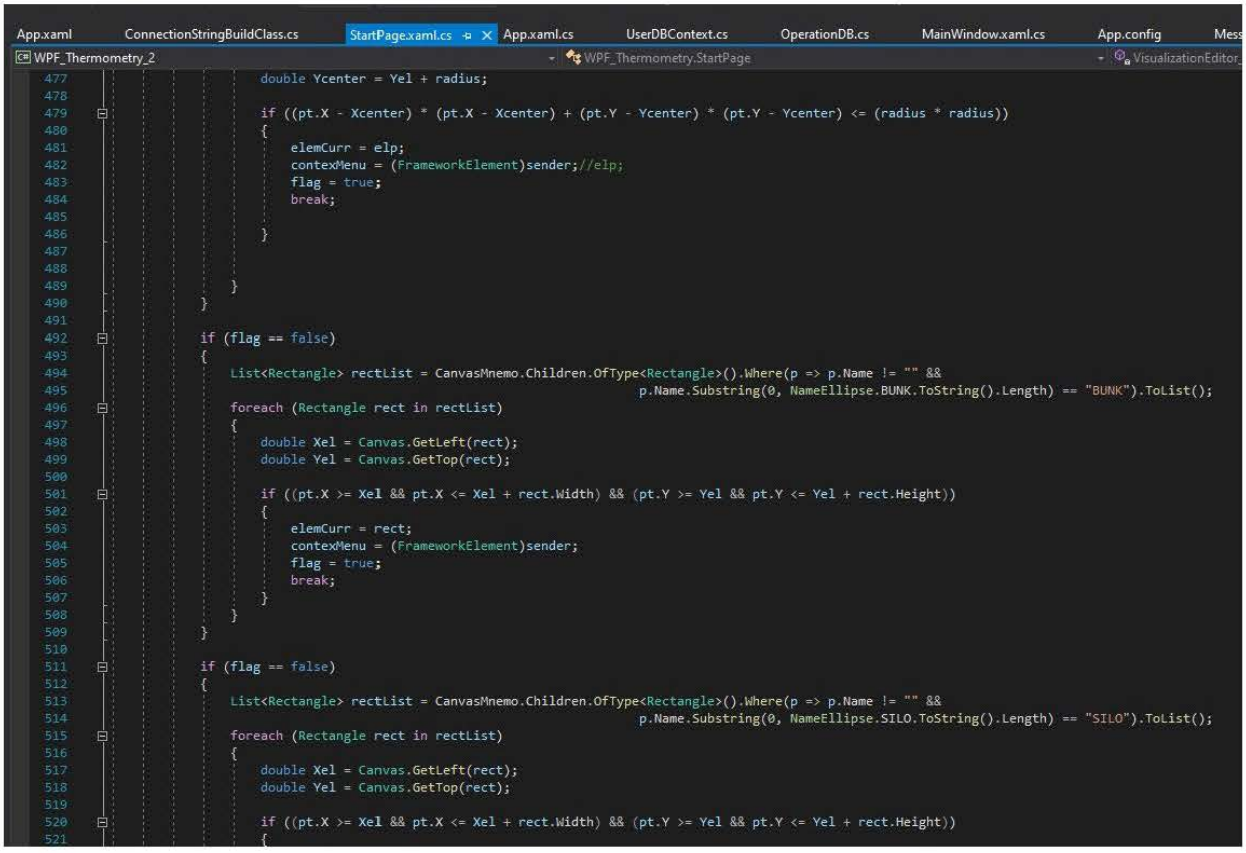
```
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.xaml.cs  App.config  MessageBo
WPF_Thermometry_2  WPF_Thermometry.StartPage  VisualizationEditor_Click(
344             // провезка на количество подвесок с одинаковыми адресами
345
346             if ( (typeSilo == "Группа Круглых" && typeBunker == "Силкорпус" && name == NameEllipse.TSUS.ToString()) ||
347                  (typeSilo == "Прямоугольный" && (typeBunker == "Прямоугольный" || typeBunker == "Круглый" || typeBunker == "Звезда" ||
348                                                   typeBunker == "ПолуЗвезда правая" || typeBunker == "ПолуЗвезда левая")
349                                               && name == NameEllipse.BUNK.ToString()) )
350             {
351                 contexMenu.ContextMenu = new ContextMenu()
352                 {
353                     FontSize = 14,
354                     FontWeight = FontWeights.DemiBold
355                 };
356
357
358                 MenuItem enable = new MenuItem() { Header = " П о д к л ю ч и т ь " };
359                 MenuItem disable = new MenuItem() { Header = " О т к л ю ч и т ь " };
360                 MenuItem paramert = new MenuItem() { Header = " П а р а м е т р ы  ТП " };
361
362                 if (_TSParametr.Indicat == "+")
363                 {
364
365                     enable.Visibility = Visibility.Hidden;
366                     disable.Visibility = Visibility.Visible;
367                 }
368                 else if ( TSParametr.Indicat == "-")
369                 {
370
371                     enable.Visibility = Visibility.Visible;
372                     disable.Visibility = Visibility.Hidden;
373                 }
374
375
376                 enable.Click -= new RoutedEventHandler(Enable_Click);
377                 disable.Click += new RoutedEventHandler(Disable_Click);
378                 paramert.Click += new RoutedEventHandler(Paramet_Click);
379
380                 contexMenu.ContextMenu.Items.Add(enable);
381                 contexMenu.ContextMenu.Items.Add(disable);
382                 contexMenu.ContextMenu.Items.Add(paramert);
383             }
384         }
385         //#############################################################
            ссылка: 1
386         private void CanvasMnemo_LeftMouseButton( string strItem, object sender, MouseButtonEventArgs e)
387         {
```



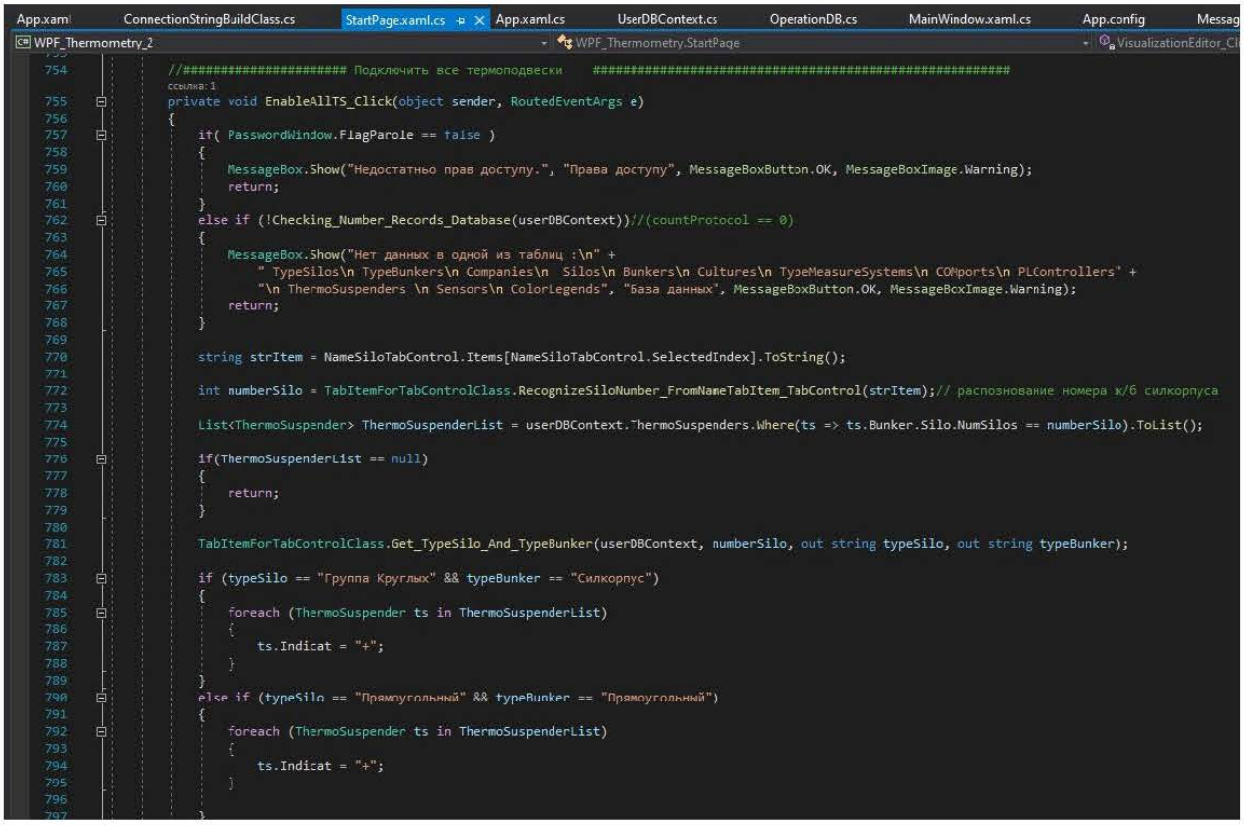
```
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.xaml.cs  App.config  Message
WPF_Thermometry_2  WPF_Thermometry.StartPage  VisualizationEditor_Clic
        ссылка:1
386     private void CanvasMnemo_LeftMouseButton( string strItem, object sender, MouseButtonEventArgs e)
387     {
388         //FrameworkElement elem = Find_FigureTS(sender, e, out FrameworkElement contexMenu);
389
390         //Shape elem = Find_FigureTS(sender, e, out FrameworkElement contexMenu);
391         _elem = Find_FigureTS(sender, e, out FrameworkElement contexMenu);
392
393
394         string name = (_elem.Name != "") ? _elem.Name.Substring(0, NameEllipse.SILO.ToString().Length) : _elem.Name;
395         int numberSilo = TabItemForTabControlClass.RecognizeSiloNumber_FromNameTabItem_TabControl(strItem);// распознование номера  силкорпуса
396
397         TabItemForTabControlClass.Get_TypeSilo_And_TypeBunker(userDBContext, numberSilo, out string typeSilo, out string typeBunker);
398
399         if (typeSilo == "Группа Круглых" && typeBunker == "Силкорпус")
400         {
401             if (name == NameEllipse.BUNK.ToString())
402             {
403                 Silo_Info_Window siloInfo = new Silo_Info_Window((Ellipse)_elem, numberSilo, userDBContext); // информационное окно для силкорпуса
404                 siloInfo.Show();
405                 siloInfo.InfoText();
406             }
407             else if (name == NameEllipse.TSUS.ToString())
408             {
409                 TS_Info_Window tsInfo = new TS_Info_Window((Ellipse)_elem, numberSilo, userDBContext);  // информационное окно для термоподвески
410                 tsInfo.Show();
411                 tsInfo.InfoTextTS();
412             }
413         }
414         else if (typeSilo == "Прямоугольный")
415         {
416
417             if (name == NameRectangle.SILO.ToString())
418             {
419                 Silo_Info_Rect_Window siloInfo = new Silo_Info_Rect_Window((Rectangle)_elem, userDBContext); // информационное окно для силкорпуса
420                 siloInfo.Show();
421                 siloInfo.InfoText();
422             }
423             else if (name == NameRectangle.BUNK.ToString())
424             {
425                 TS_Info_Rect_Window bunkerInfo = new TS_Info_Rect_Window((Shape)_elem, userDBContext);  // информационное окно для термоподвески
426                 bunkerInfo.Show();
427                 bunkerInfo.InfoTextBunkerTS();
428             }
429         }
```



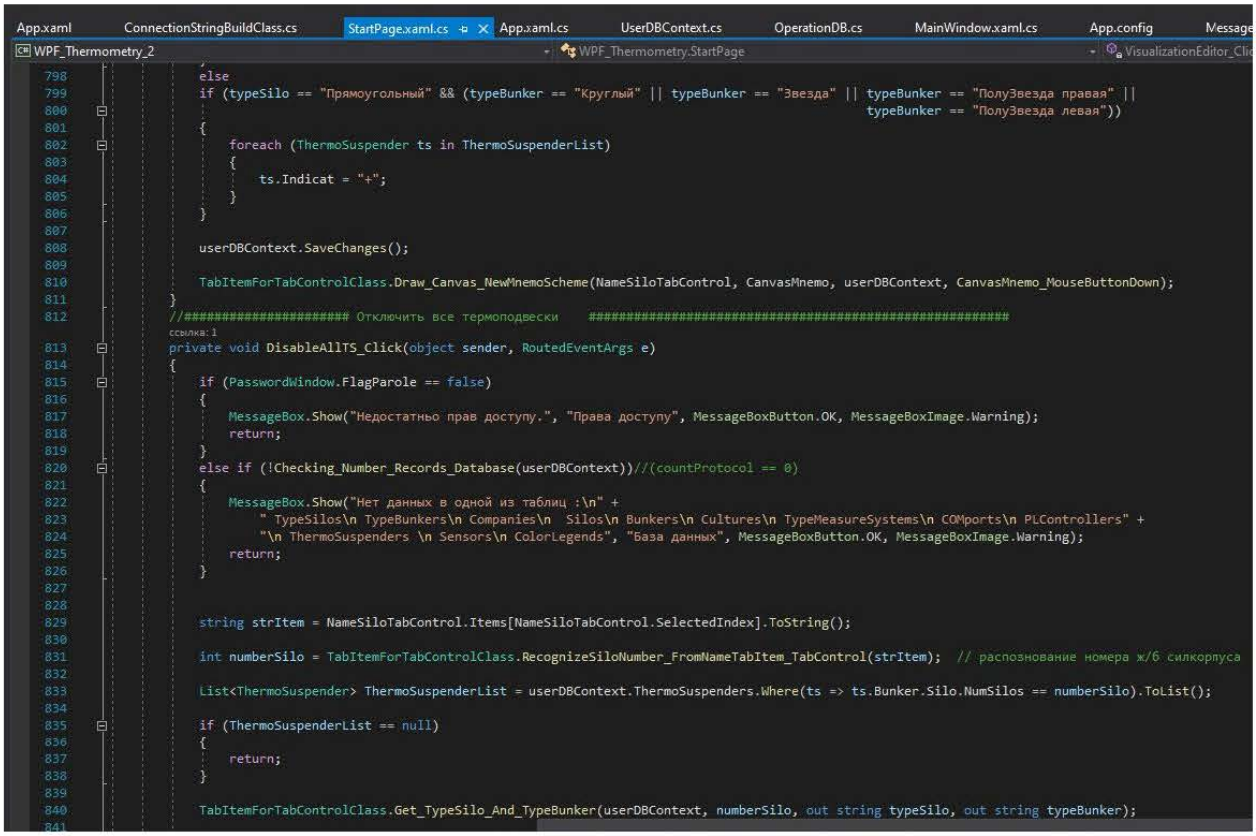
```
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.xaml.cs  App.config
WPF_Thermometry_2  WPF_Thermometry.StartPage  Visualizatior
432     private Shape Find_FigureTS(object sender, MouseButtonEventArgs e, out FrameworkElement contexMenu) // для ContextMenu
433     {
434         bool flag = false;
435         Shape elemCurr = null;
436         contexMenu = null;
437
438         Point pt = e.GetPosition(CanvasMnemo);
439
440
441         List<Ellipse> elList = CanvasMnemo.Children.OfType<Ellipse>().Where(p => p.Name != "" &&
442                                                                 p.Name.Substring(0, NameEllipse.TSUS.ToString().Length) == "TSUS").ToList();
443
444         foreach (Ellipse elp in elList)
445         {
446             double Xel = Canvas.GetLeft(elp);
447             double Yel = Canvas.GetTop(elp);
448
449             double radius = elp.Width / 2.0;
450             double Xcenter = Xel + radius;
451             double Ycenter = Yel + radius;
452
453             if ((pt.X - Xcenter) * (pt.X - Xcenter) + (pt.Y - Ycenter) * (pt.Y - Ycenter) <= (radius * radius))
454             {
455                 elemCurr = elp;
456                 contexMenu = (FrameworkElement)sender;//elp;
457                 flag = true;
458                 break;
459
460             }
461
462
463         }
464
465         if (flag == false)
466         {
467             List<Ellipse> elListS = CanvasMnemo.Children.OfType<Ellipse>().Where(p => p.Name != "" &&
468                                                                 p.Name.Substring(0, NameEllipse.BUNK.ToString().Length) == "BUNK").ToList();
469
470             foreach (Ellipse elp in elListS)
471             {
472                 double Xel = Canvas.GetLeft(elp);
473                 double Yel = Canvas.GetTop(elp);
474
475                 double radius = elp.Width / 2.0;
476                 double Xcenter = Xel + radius;
```



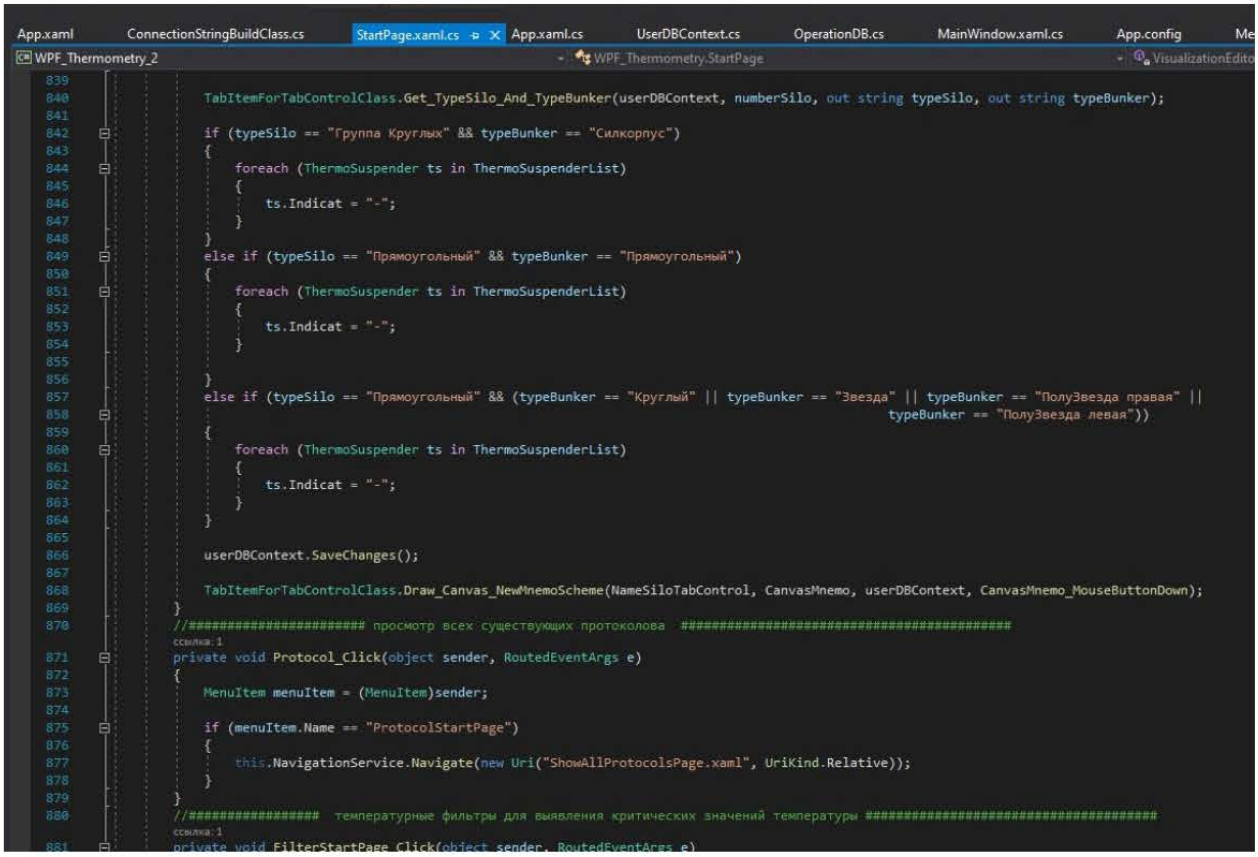
```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs | App.config
WPF_Thermometry_2 | WPF_Thermometry.StartPage | VisualizationEditor_

                    double Ycenter = Yel + radius;

                    if ((pt.X - Xcenter) * (pt.X - Xcenter) + (pt.Y - Ycenter) * (pt.Y - Ycenter) <= (radius * radius))
                    {
                        elemCurr = elp;
                        contexMenu = (FrameworkElement)sender;//elp;
                        flag = true;
                        break;

                    }


                }
            }

            if (flag == false)
            {
                List<Rectangle> rectList = CanvasMnemo.Children.OfType<Rectangle>().Where(p => p.Name != "" &&
                                                                    p.Name.Substring(0, NameEllipse.BUNK.ToString().Length) == "BUNK").ToList();
                foreach (Rectangle rect in rectList)
                {
                    double Xel = Canvas.GetLeft(rect);
                    double Yel = Canvas.GetTop(rect);

                    if ((pt.X >= Xel && pt.X <= Xel + rect.Width) && (pt.Y >= Yel && pt.Y <= Yel + rect.Height))
                    {
                        elemCurr = rect;
                        contexMenu = (FrameworkElement)sender;
                        flag = true;
                        break;
                    }
                }
            }

            if (flag == false)
            {
                List<Rectangle> rectList = CanvasMnemo.Children.OfType<Rectangle>().Where(p => p.Name != "" &&
                                                                    p.Name.Substring(0, NameEllipse.SILO.ToString().Length) == "SILO").ToList();
                foreach (Rectangle rect in rectList)
                {
                    double Xel = Canvas.GetLeft(rect);
                    double Yel = Canvas.GetTop(rect);

                    if ((pt.X >= Xel && pt.X <= Xel + rect.Width) && (pt.Y >= Yel && pt.Y <= Yel + rect.Height))
                    {
```



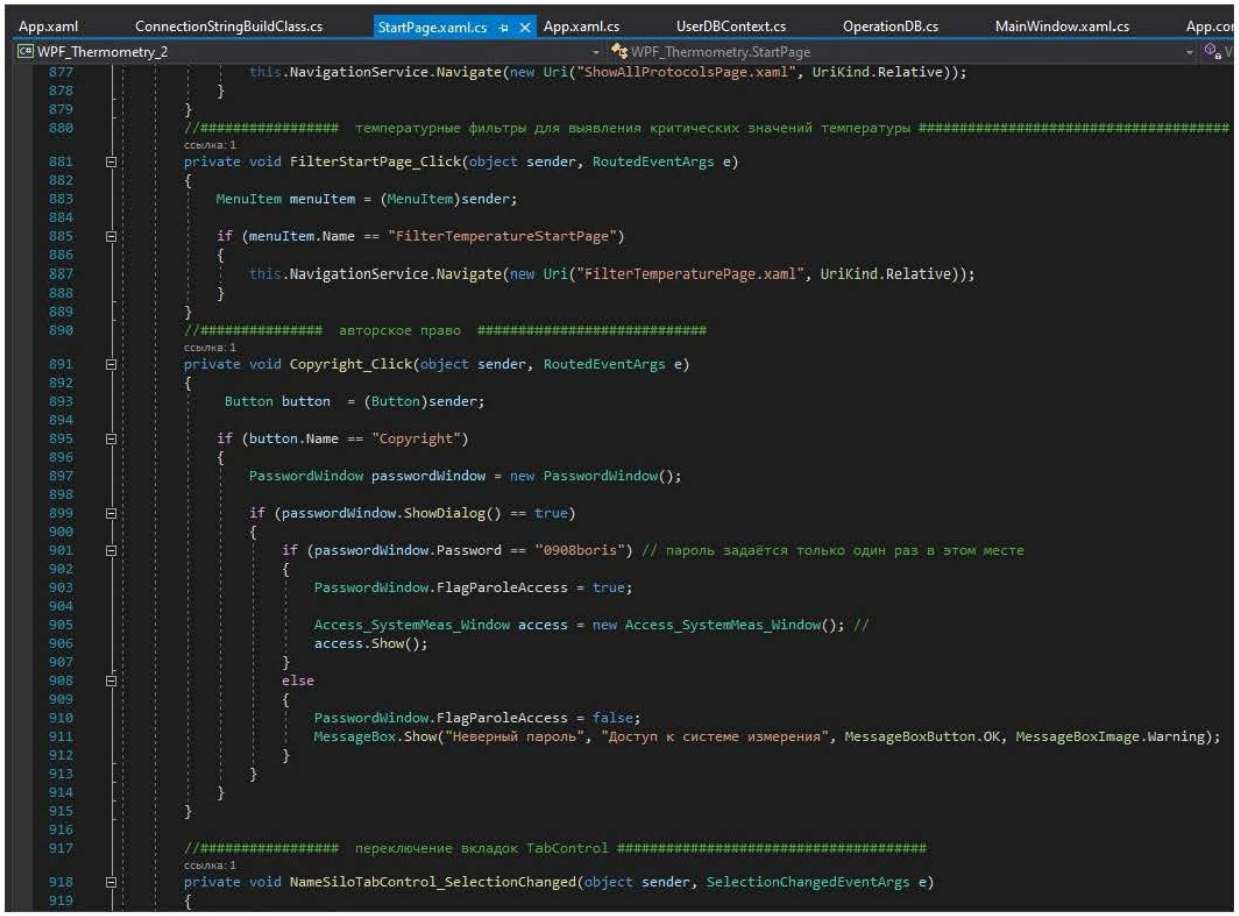
```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs | App.config
WPF_Thermometry_2 | WPF_Thermometry.StartPage | VisualizationEditor_Cl

        //#################### Подключить все термоподвески    ##################################################
        ссылка: 1
        private void EnableAllTS_Click(object sender, RoutedEventArgs e)
        {
            if( PasswordWindow.FlagParole == false )
            {
                MessageBox.Show("Недостатньо прав доступу.", "Права доступу", MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }
            else if (!Checking_Number_Records_Database(userDBContext))//(countProtocol == 0)
            {
                MessageBox.Show("Нет данных в одной из таблиц :\n" +
                    " TypeSilos\n TypeBunkers\n Companies\n  Silos\n Bunkers\n Cultures\n TypeMeasureSystems\n COMports\n PLControllers" +
                    "\n ThermoSuspenders \n Sensors\n ColorLegends", "База данных", MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            string strItem = NameSiloTabControl.Items[NameSiloTabControl.SelectedIndex].ToString();

            int numberSilo = TabItemForTabControlClass.RecognizeSiloNumber_FromNameTabItem_TabControl(strItem);// распознование номера к/б силкорпуса

            List<ThermoSuspender> ThermoSuspenderList = userDBContext.ThermoSuspenders.Where(ts => ts.Bunker.Silo.NumSilos == numberSilo).ToList();

            if(ThermoSuspenderList == null)
            {
                return;
            }

            TabItemForTabControlClass.Get_TypeSilo_And_TypeBunker(userDBContext, numberSilo, out string typeSilo, out string typeBunker);

            if (typeSilo == "Группа Круглых" && typeBunker == "Силкорпус")
            {
                foreach (ThermoSuspender ts in ThermoSuspenderList)
                {
                    ts.Indicat = "+";
                }
            }
            else if (typeSilo == "Прямоугольный" && typeBunker == "Прямоугольный")
            {
                foreach (ThermoSuspender ts in ThermoSuspenderList)
                {
                    ts.Indicat = "+";
                }

            }
```



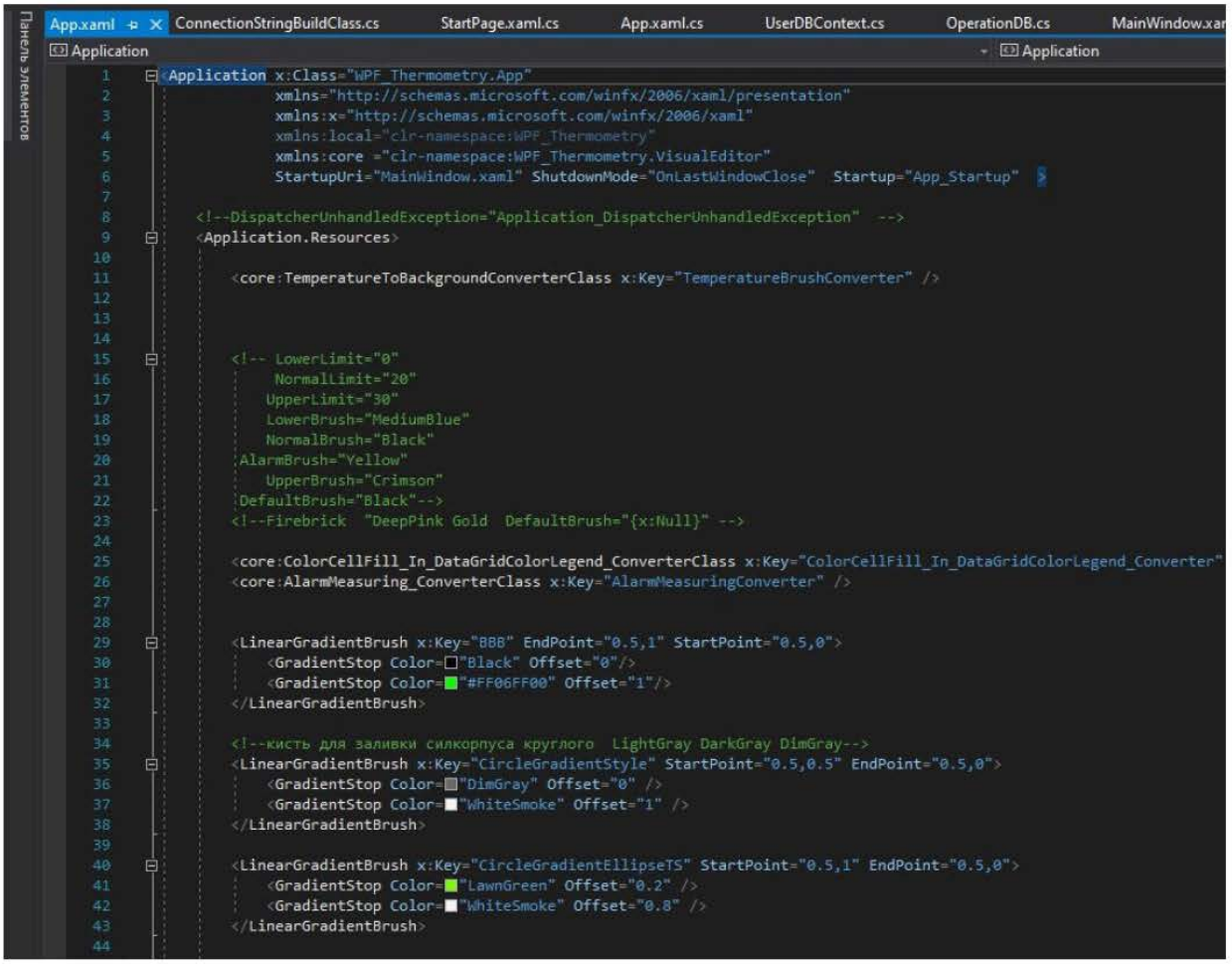
```
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.xaml.cs  App.config  Message
WPF_Thermometry_2  WPF_Thermometry.StartPage  VisualizationEditor_Clic
798         }
799         else
800         if (typeSilo == "Прямоугольный" && (typeBunker == "Круглый" || typeBunker == "Звезда" || typeBunker == "ПолуЗвезда правая" ||
801                                                                        typeBunker == "ПолуЗвезда левая"))
802         {
803             foreach (ThermoSuspender ts in ThermoSuspenderList)
804             {
805                 ts.Indicat = "+";
806             }
807         }
808
809         userDBContext.SaveChanges();
810
811         TabItemForTabControlClass.Draw_Canvas_NewMnemoScheme(NameSiloTabControl, CanvasMnemo, userDBContext, CanvasMnemo_MouseButtonDown);
812     }
        //###################### Отключить все термоподвески    ####################################################
        ссылка: 1
813     private void DisableAllTS_Click(object sender, RoutedEventArgs e)
814     {
815         if (PasswordWindow.FlagParole == false)
816         {
817             MessageBox.Show("Недостатньо прав доступу.", "Права доступу", MessageBoxButton.OK, MessageBoxImage.Warning);
818             return;
819         }
820         else if (!Checking_Number_Records_Database(userDBContext))//(countProtocol == 0)
821         {
822             MessageBox.Show("Нет данных в одной из таблиц :\n" +
823                 " TypeSilos\n TypeBunkers\n Companies\n  Silos\n Bunkers\n Cultures\n TypeMeasureSystems\n COMports\n PLControllers" +
824                 "\n ThermoSuspenders \n Sensors\n ColorLegends", "База данных", MessageBoxButton.OK, MessageBoxImage.Warning);
825             return;
826         }
827
828
829         string strItem = NameSiloTabControl.Items[NameSiloTabControl.SelectedIndex].ToString();
830
831         int numberSilo = TabItemForTabControlClass.RecognizeSiloNumber_FromNameTabItem_TabControl(strItem);  // распознование номера ж/б силкорпуса
832
833         List<ThermoSuspender> ThermoSuspenderList = userDBContext.ThermoSuspenders.Where(ts => ts.Bunker.Silo.NumSilos == numberSilo).ToList();
834
835         if (ThermoSuspenderList == null)
836         {
837             return;
838         }
839
840         TabItemForTabControlClass.Get_TypeSilo_And_TypeBunker(userDBContext, numberSilo, out string typeSilo, out string typeBunker);
841
```



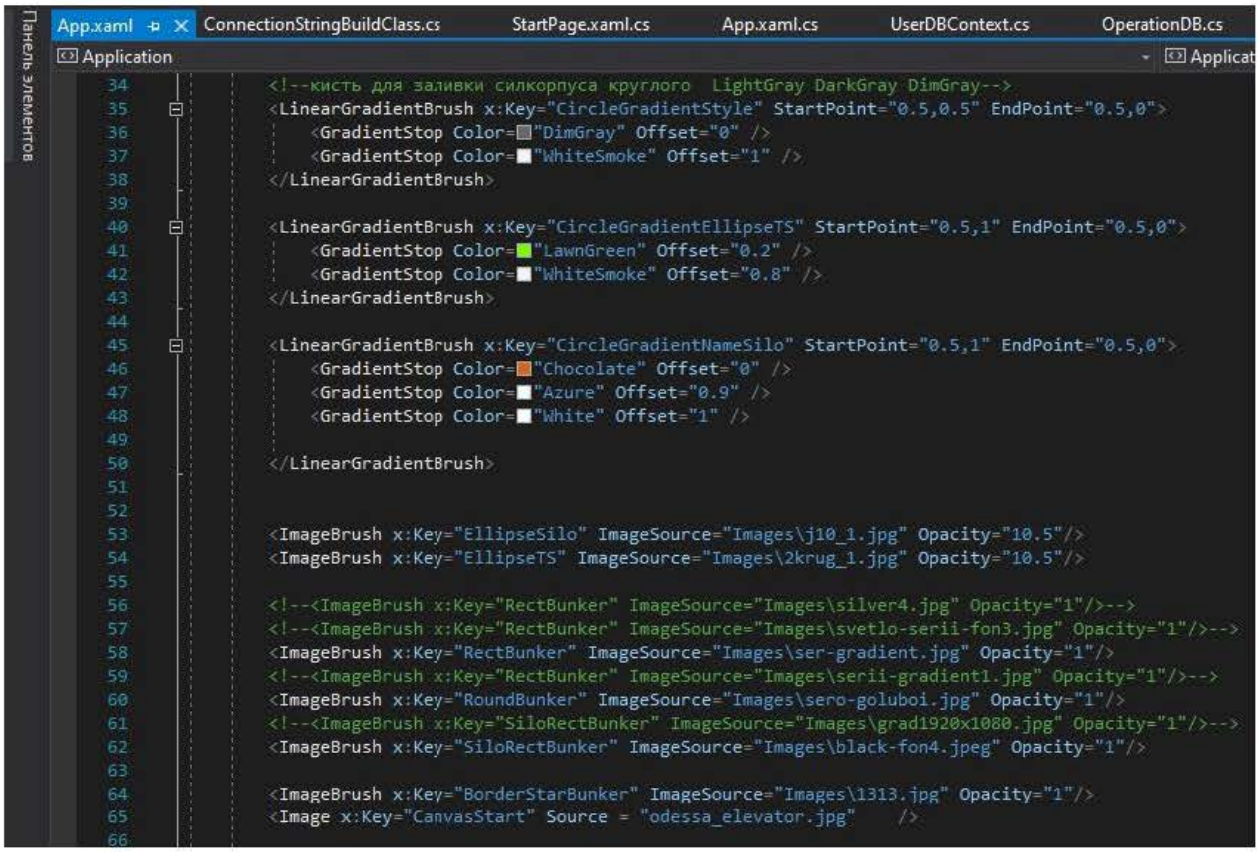
```
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.xaml.cs  App.config  Me
WPF_Thermometry_2  WPF_Thermometry.StartPage  VisualizationEdito
839
840         TabItemForTabControlClass.Get_TypeSilo_And_TypeBunker(userDBContext, numberSilo, out string typeSilo, out string typeBunker);
841
842         if (typeSilo == "Группа Круглых" && typeBunker == "Силкорпус")
843         {
844             foreach (ThermoSuspender ts in ThermoSuspenderList)
845             {
846                 ts.Indicat = "-";
847             }
848         }
849         else if (typeSilo == "Прямоугольный" && typeBunker == "Прямоугольный")
850         {
851             foreach (ThermoSuspender ts in ThermoSuspenderList)
852             {
853                 ts.Indicat = "-";
854             }
855
856         }
857         else if (typeSilo == "Прямоугольный" && (typeBunker == "Круглый" || typeBunker == "Звезда" || typeBunker == "ПолуЗвезда правая" ||
858                                                                        typeBunker == "ПолуЗвезда левая"))
859         {
860             foreach (ThermoSuspender ts in ThermoSuspenderList)
861             {
862                 ts.Indicat = "-";
863             }
864         }
865
866         userDBContext.SaveChanges();
867
868         TabItemForTabControlClass.Draw_Canvas_NewMnemoScheme(NameSiloTabControl, CanvasMnemo, userDBContext, CanvasMnemo_MouseButtonDown);
869     }
870     //###################### просмотр всех существующих протоколова  ##################################
        ссылка: 1
871     private void Protocol_Click(object sender, RoutedEventArgs e)
872     {
873         MenuItem menuItem = (MenuItem)sender;
874
875         if (menuItem.Name == "ProtocolStartPage")
876         {
877             this.NavigationService.Navigate(new Uri("ShowAllProtocolsPage.xaml", UriKind.Relative));
878         }
879     }
880     //################  температурные фильтры для выявления критических значений температуры ###################################
        ссылка: 1
881     private void FilterStartPage_Click(object sender, RoutedEventArgs e)
```

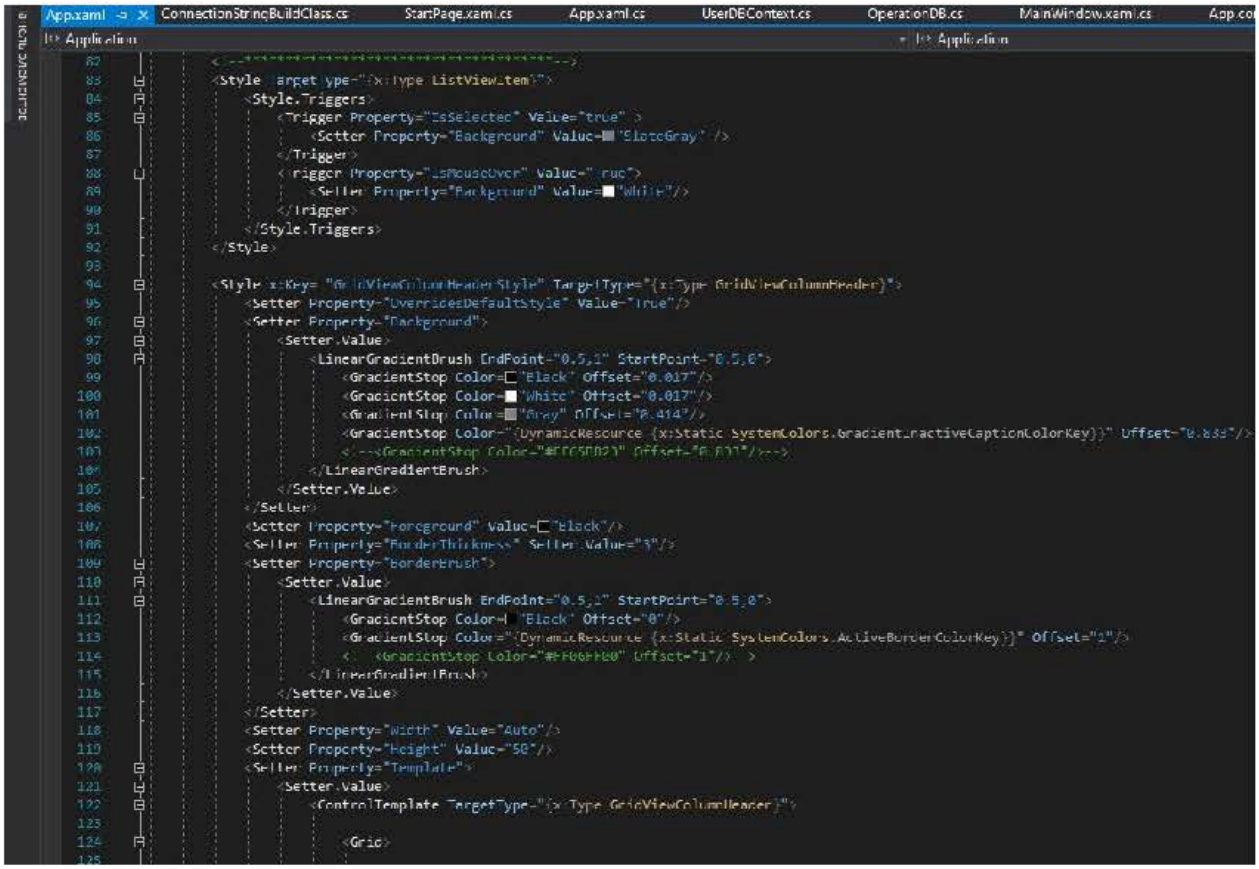


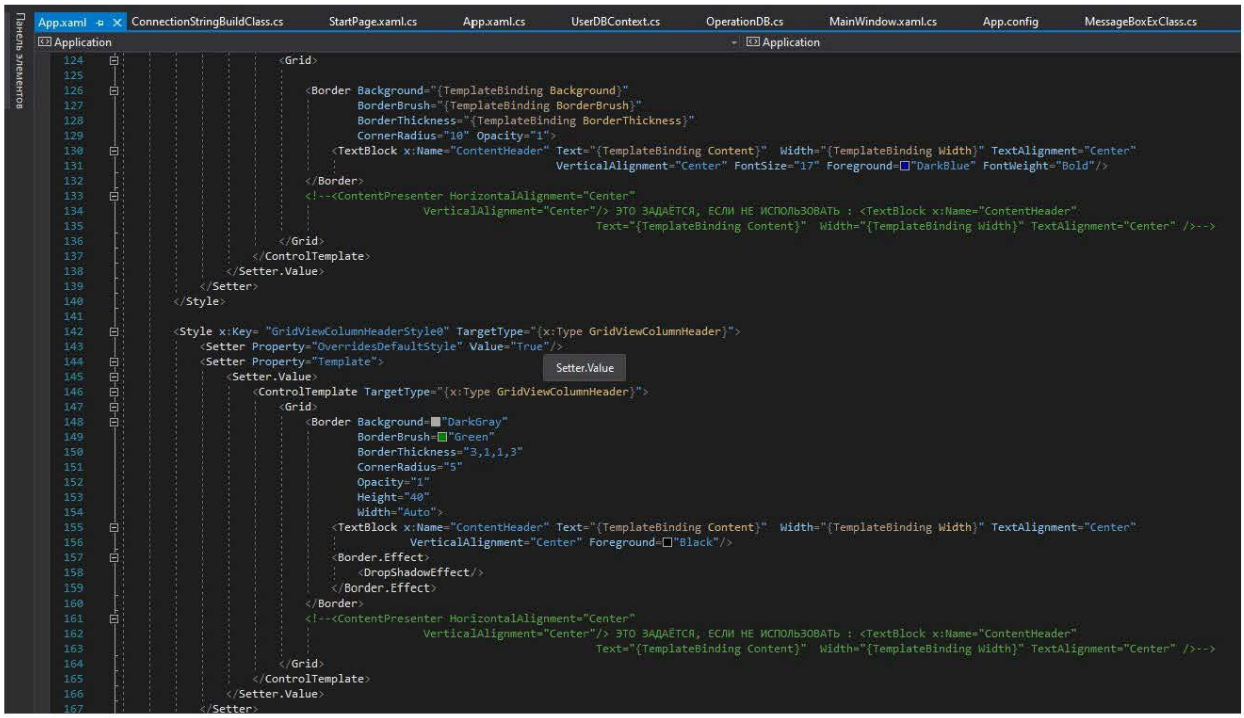
```
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.xaml.cs  App.co
WPF_Thermometry_2                                  WPF_Thermometry.StartPage
877             this.NavigationService.Navigate(new Uri("ShowAllProtocolsPage.xaml", UriKind.Relative));
878         }
879     }
880     //#################  температурные фильтры для выявления критических значений температуры ###################################
        ссылка: 1
881     private void FilterStartPage_Click(object sender, RoutedEventArgs e)
882     {
883         MenuItem menuItem = (MenuItem)sender;
884
885         if (menuItem.Name == "FilterTemperatureStartPage")
886         {
887             this.NavigationService.Navigate(new Uri("FilterTemperaturePage.xaml", UriKind.Relative));
888         }
889     }
890     //###############  авторское право  ##########################
        ссылка: 1
891     private void Copyright_Click(object sender, RoutedEventArgs e)
892     {
893         Button button  = (Button)sender;
894
895         if (button.Name == "Copyright")
896         {
897             PasswordWindow passwordWindow = new PasswordWindow();
898
899             if (passwordWindow.ShowDialog() == true)
900             {
901                 if (passwordWindow.Password == "0908boris") // пароль задаётся только один раз в этом месте
902                 {
903                     PasswordWindow.FlagParoleAccess = true;
904
905                     Access_SystemMeas_Window access = new Access_SystemMeas_Window(); //
906                     access.Show();
907                 }
908                 else
909                 {
910                     PasswordWindow.FlagParoleAccess = false;
911                     MessageBox.Show("Неверный пароль", "Доступ к системе измерения", MessageBoxButton.OK, MessageBoxImage.Warning);
912                 }
913             }
914         }
915     }
916
917     //#################  переключение вкладок TabControl ###################################
        ссылка: 1
918     private void NameSiloTabControl_SelectionChanged(object sender, SelectionChangedEventArgs e)
919     {
```

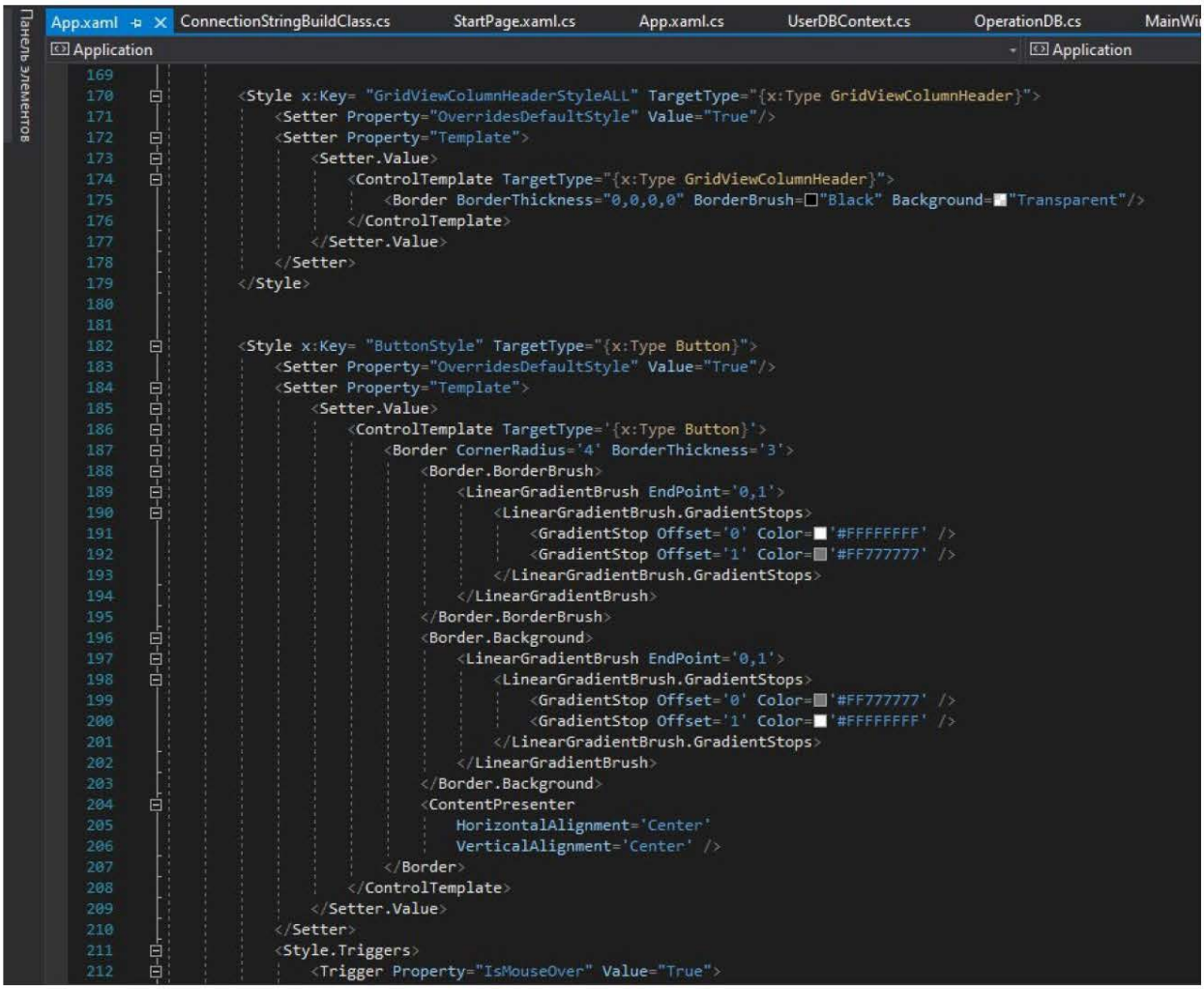


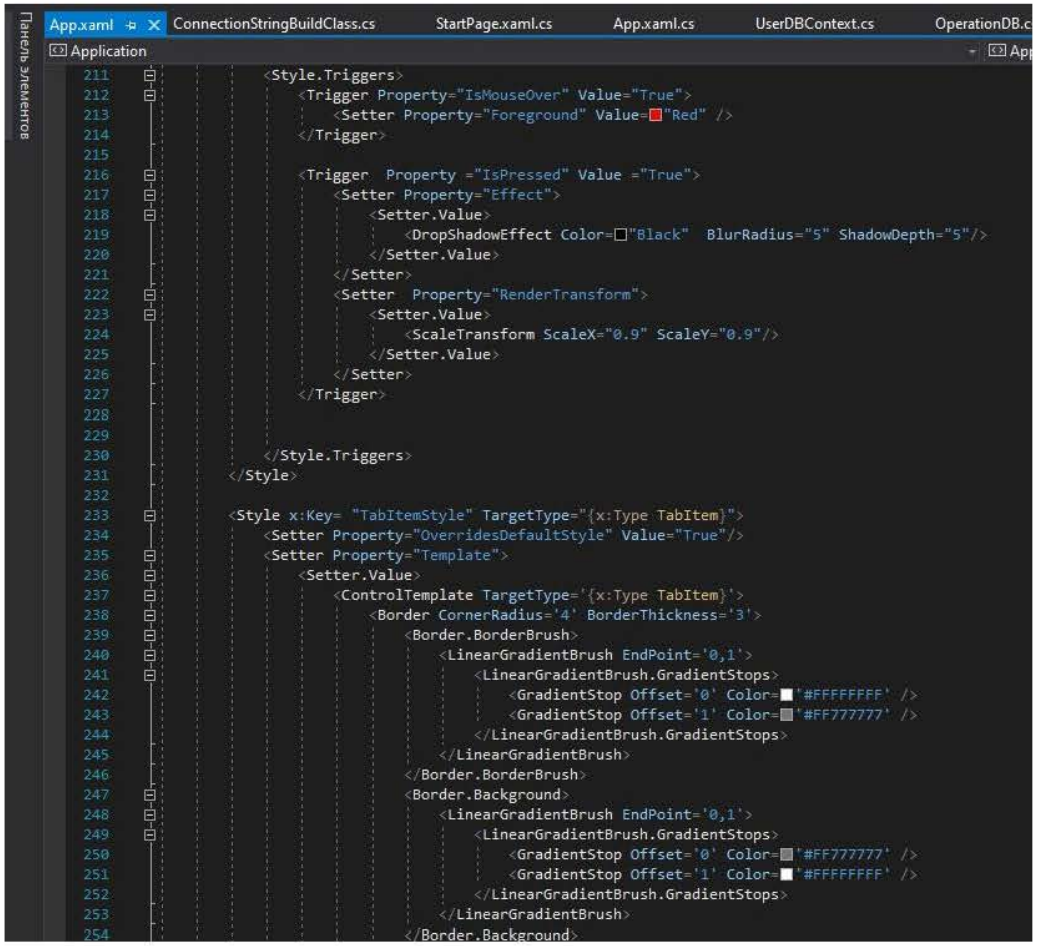
```
Панель элементов
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.xar
Application                                                  Application
1   <Application x:Class="WPF_Thermometry.App"
2                xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3                xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4                xmlns:local="clr-namespace:WPF_Thermometry"
5                xmlns:core ="clr-namespace:WPF_Thermometry.VisualEditor"
6                StartupUri="MainWindow.xaml" ShutdownMode="OnLastWindowClose"  Startup="App_Startup"  >
7
8       <!--DispatcherUnhandledException="Application_DispatcherUnhandledException"  -->
9       <Application.Resources>
10
11          <core:TemperatureToBackgroundConverterClass x:Key="TemperatureBrushConverter" />
12
13
14
15          <!-- LowerLimit="0"
16               NormalLimit="20"
17              UpperLimit="30"
18              LowerBrush="MediumBlue"
19              NormalBrush="Black"
20          AlarmBrush="Yellow"
21              UpperBrush="Crimson"
22          DefaultBrush="Black"-->
23          <!--Firebrick  "DeepPink Gold  DefaultBrush="{x:Null}" -->
24
25          <core:ColorCellFill_In_DataGridColorLegend_ConverterClass x:Key="ColorCellFill_In_DataGridColorLegend_Converter"
26          <core:AlarmMeasuring_ConverterClass x:Key="AlarmMeasuringConverter" />
27
28
29          <LinearGradientBrush x:Key="BBB" EndPoint="0.5,1" StartPoint="0.5,0">
30              <GradientStop Color="Black" Offset="0"/>
31              <GradientStop Color="#FF06FF00" Offset="1"/>
32          </LinearGradientBrush>
33
34          <!--кисть для заливки силкорпуса круглого  LightGray DarkGray DimGray-->
35          <LinearGradientBrush x:Key="CircleGradientStyle" StartPoint="0.5,0.5" EndPoint="0.5,0">
36              <GradientStop Color="DimGray" Offset="0" />
37              <GradientStop Color="WhiteSmoke" Offset="1" />
38          </LinearGradientBrush>
39
40          <LinearGradientBrush x:Key="CircleGradientEllipseTS" StartPoint="0.5,1" EndPoint="0.5,0">
41              <GradientStop Color="LawnGreen" Offset="0.2" />
42              <GradientStop Color="WhiteSmoke" Offset="0.8" />
43          </LinearGradientBrush>
44
```


App.xaml
ConnectionStringBuildClass.cs
StartPage.xaml.cs
App.xaml.cs
UserDBContext.cs
OperationDB.cs
Application
<!--кисть для заливки силкорпуса круглого LightGray DarkGray DimGray-->
<LinearGradientBrush x:Key="CircleGradientStyle" StartPoint="0.5,0.5" EndPoint="0.5,0">
<GradientStop Color="DimGray" Offset="0" />
<GradientStop Color="WhiteSmoke" Offset="1" />
</LinearGradientBrush>
<LinearGradientBrush x:Key="CircleGradientEllipseTS" StartPoint="0.5,1" EndPoint="0.5,0">
<GradientStop Color="LawnGreen" Offset="0.2" />
<GradientStop Color="WhiteSmoke" Offset="0.8" />
</LinearGradientBrush>
<LinearGradientBrush x:Key="CircleGradientNameSilo" StartPoint="0.5,1" EndPoint="0.5,0">
<GradientStop Color="Chocolate" Offset="0" />
<GradientStop Color="Azure" Offset="0.9" />
<GradientStop Color="White" Offset="1" />
</LinearGradientBrush>
<ImageBrush x:Key="EllipseSilo" ImageSource="Images\j10_1.jpg" Opacity="10.5"/>
<ImageBrush x:Key="EllipseTS" ImageSource="Images\2krug_1.jpg" Opacity="10.5"/>
<!--<ImageBrush x:Key="RectBunker" ImageSource="Images\silver4.jpg" Opacity="1"/>-->
<!--<ImageBrush x:Key="RectBunker" ImageSource="Images\svetlo-serii-fon3.jpg" Opacity="1"/>-->
<ImageBrush x:Key="RectBunker" ImageSource="Images\ser-gradient.jpg" Opacity="1"/>
<!--<ImageBrush x:Key="RectBunker" ImageSource="Images\serii-gradient1.jpg" Opacity="1"/>-->
<ImageBrush x:Key="RoundBunker" ImageSource="Images\sero-goluboi.jpg" Opacity="1"/>
<!--<ImageBrush x:Key="SiloRectBunker" ImageSource="Images\grad1920x1080.jpg" Opacity="1"/>-->
<ImageBrush x:Key="SiloRectBunker" ImageSource="Images\black-fon4.jpeg" Opacity="1"/>
<ImageBrush x:Key="BorderStarBunker" ImageSource="Images\1313.jpg" Opacity="1"/>
<Image x:Key="CanvasStart" Source = "odessa_elevator.jpg" />
Панель элементов

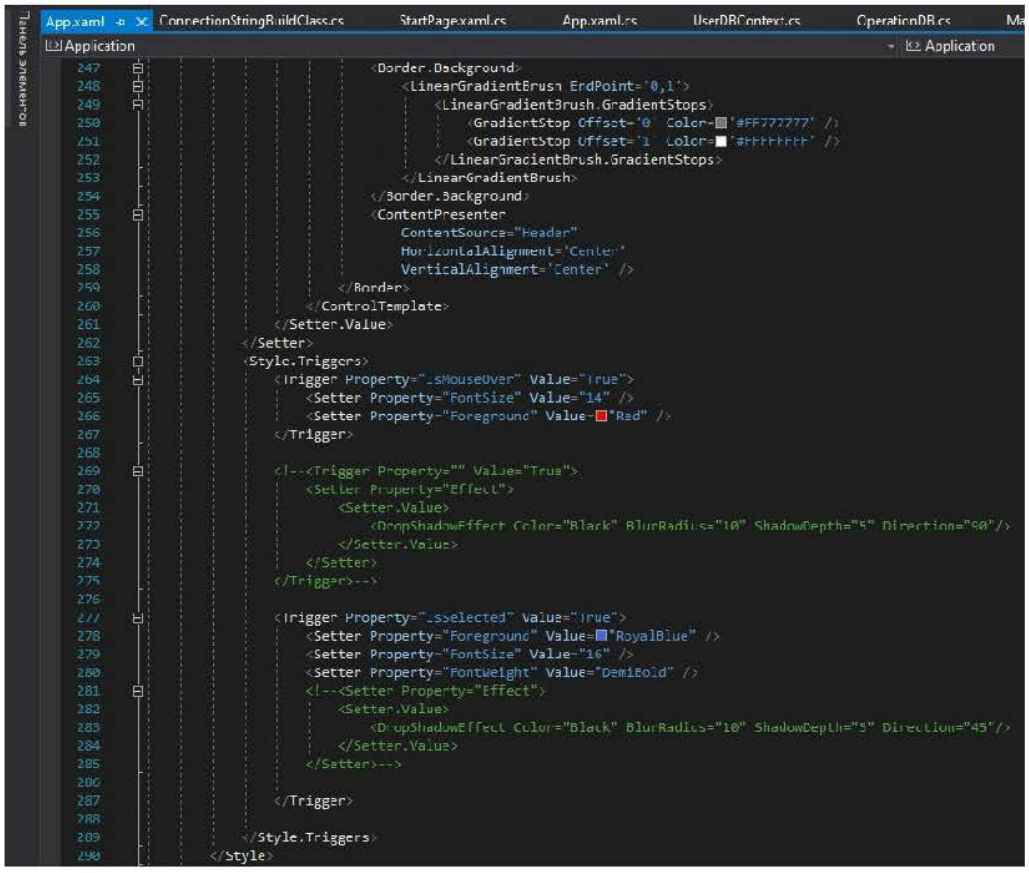

App.xaml
ConnectionStringBuildClass.cs
StartPage.xaml.cs
App.xaml.cs
UserDBContext.cs
OperationDB.cs
MainWindow.xaml.cs
Application
<Style TargetType="{x:Type ListViewItem}">
<Style.Triggers>
<Trigger Property="IsSelected" Value="true" >
<Setter Property="Background" Value="SlateGray" />
</Trigger>
<Trigger Property="IsMouseOver" Value="true">
<Setter Property="Background" Value="White"/>
</Trigger>
</Style.Triggers>
</Style>
<Style x:Key="GridViewColumnHeaderStyle" TargetType="{x:Type GridViewColumnHeader}">
<Setter Property="OverridesDefaultStyle" Value="True"/>
<Setter Property="Background">
<Setter.Value>
<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
<GradientStop Color="Black" Offset="0.017"/>
<GradientStop Color="White" Offset="0.017"/>
<GradientStop Color="Grey" Offset="0.414"/>
<GradientStop Color="{DynamicResource {x:Static SystemColors.GradientInactiveCaptionColorKey}}" Offset="0.638"/>
</LinearGradientBrush>
</Setter.Value>
</Setter>
<Setter Property="Foreground" Value="Black"/>
<Setter Property="BorderThickness" Setter.Value="3"/>
<Setter Property="BorderBrush">
<Setter.Value>
<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
<GradientStop Color="Black" Offset="0"/>
<GradientStop Color="{DynamicResource {x:Static SystemColors.ActiveBorderColorKey}}" Offset="1"/>
</LinearGradientBrush>
</Setter.Value>
</Setter>
<Setter Property="Width" Value="Auto"/>
<Setter Property="Height" Value="50"/>
<Setter Property="Template">
<Setter.Value>
<ControlTemplate TargetType="{x:Type GridViewColumnHeader}">
<Grid>


App.xaml
ConnectionStringBuildClass.cs
StartPage.xaml.cs
App.xaml.cs
UserDBContext.cs
OperationDB.cs
MainWindow.xaml.cs
App.config
MessageBoxExClass.cs
Application
Application
Setter.Value

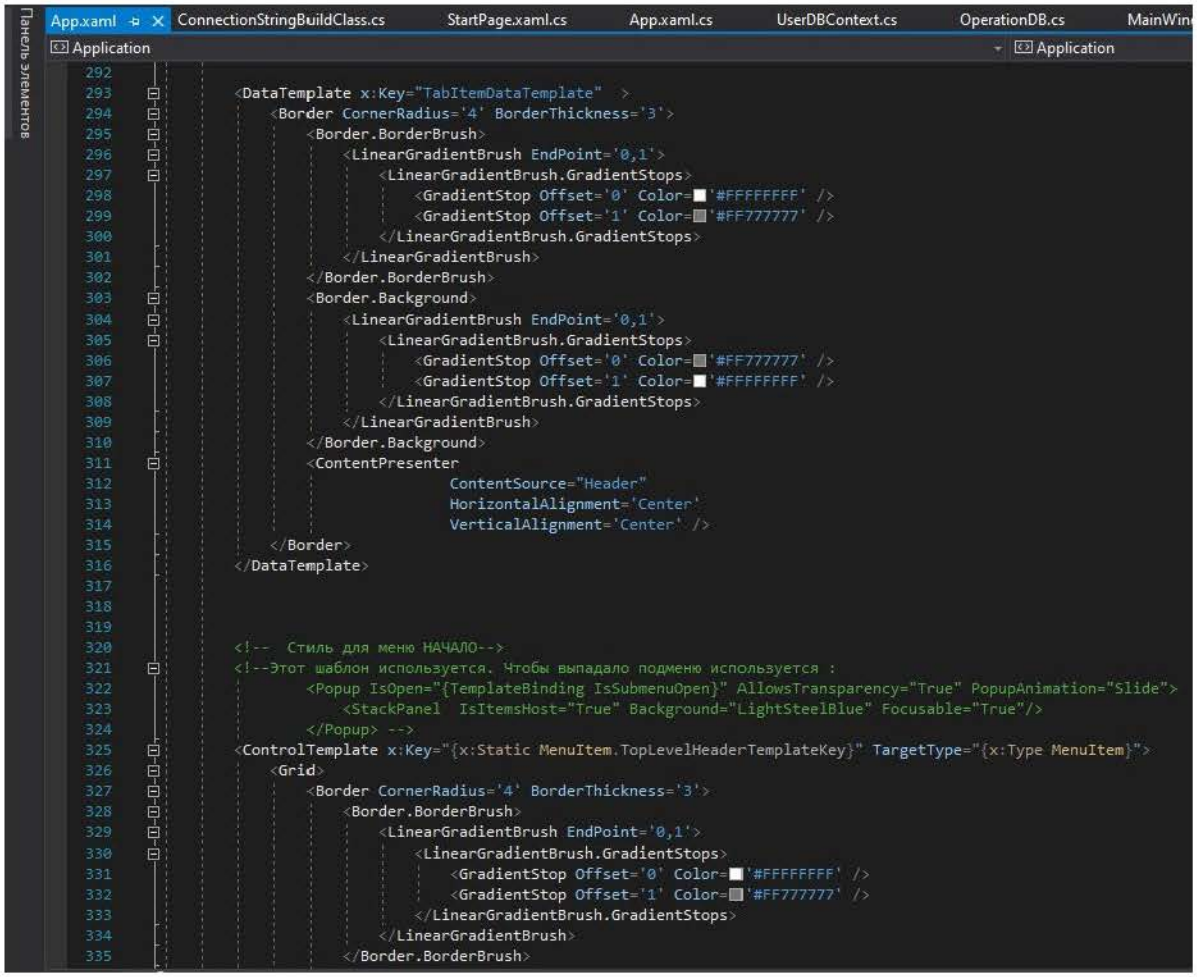

App.xaml
ConnectionStringBuildClass.cs
StartPage.xaml.cs
App.xaml.cs
UserDBContext.cs
OperationDB.cs
Application
Application

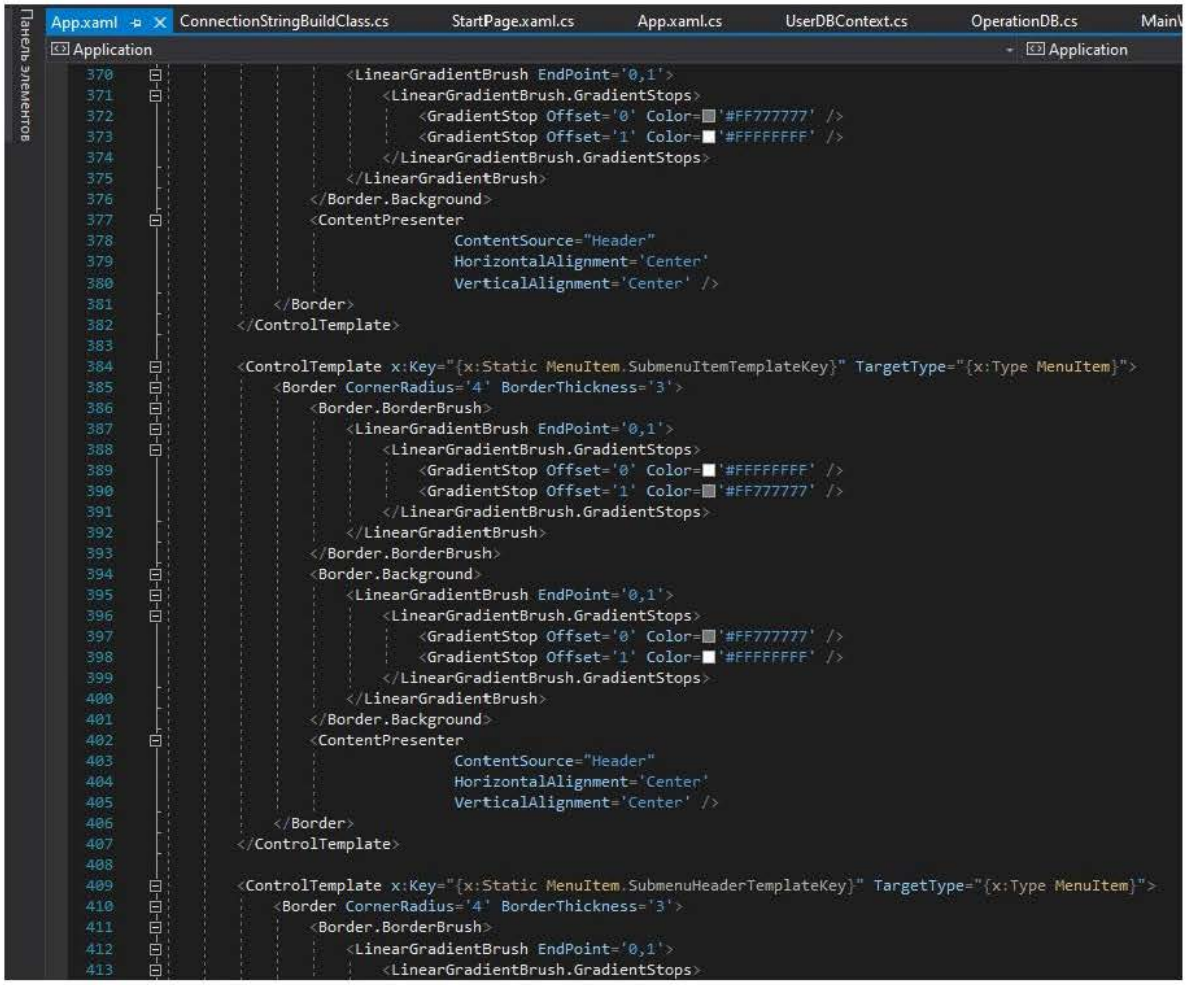


```
<Style.Triggers>
    <Trigger Property="IsMouseOver" Value="True">
        <Setter Property="Foreground" Value="Red" />
    </Trigger>

    <Trigger  Property ="IsPressed" Value ="True">
        <Setter Property="Effect">
            <Setter.Value>
                <DropShadowEffect Color="Black"  BlurRadius="5" ShadowDepth="5"/>
            </Setter.Value>
        </Setter>
        <Setter  Property="RenderTransform">
            <Setter.Value>
                <ScaleTransform ScaleX="0.9" ScaleY="0.9"/>
            </Setter.Value>
        </Setter>
    </Trigger>


</Style.Triggers>
</Style>

<Style x:Key= "TabItemStyle" TargetType="{x:Type TabItem}">
    <Setter Property="OverridesDefaultStyle" Value="True"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType='{x:Type TabItem}'>
                <Border CornerRadius='4' BorderThickness='3'>
                    <Border.BorderBrush>
                        <LinearGradientBrush EndPoint='0,1'>
                            <LinearGradientBrush.GradientStops>
                                <GradientStop Offset='0' Color='#FFFFFFFF' />
                                <GradientStop Offset='1' Color='#FF777777' />
                            </LinearGradientBrush.GradientStops>
                        </LinearGradientBrush>
                    </Border.BorderBrush>
                    <Border.Background>
                        <LinearGradientBrush EndPoint='0,1'>
                            <LinearGradientBrush.GradientStops>
                                <GradientStop Offset='0' Color='#FF777777' />
                                <GradientStop Offset='1' Color='#FFFFFFFF' />
                            </LinearGradientBrush.GradientStops>
                        </LinearGradientBrush>
                    </Border.Background>
```

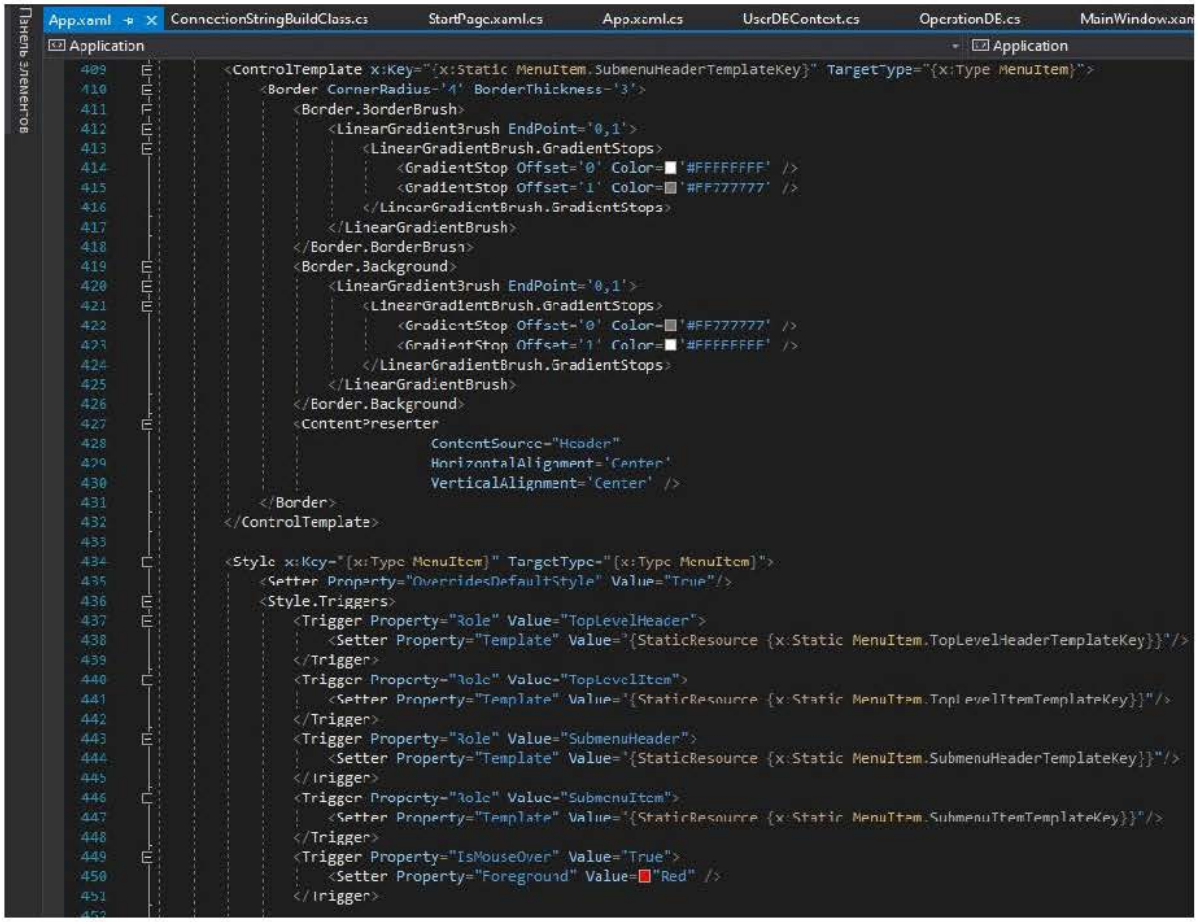


```
                    <Border.Background>
                        <LinearGradientBrush EndPoint='0,1'>
                            <LinearGradientBrush.GradientStops>
                                <GradientStop Offset='0' Color='#FF777777' />
                                <GradientStop Offset='1' Color='#FFFFFFFF' />
                            </LinearGradientBrush.GradientStops>
                        </LinearGradientBrush>
                    </Border.Background>
                    <ContentPresenter
                        ContentSource="Header"
                        HorizontalAlignment='Center'
                        VerticalAlignment='Center' />
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
    <Style.Triggers>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="FontSize" Value="14" />
            <Setter Property="Foreground" Value="Red" />
        </Trigger>

        <!--<Trigger Property="" Value="True">
            <Setter Property="Effect">
                <Setter.Value>
                    <DropShadowEffect Color="Black" BlurRadius="10" ShadowDepth="5" Direction="90"/>
                </Setter.Value>
            </Setter>
        </Trigger>-->

        <Trigger Property="IsSelected" Value="True">
            <Setter Property="Foreground" Value="RoyalBlue" />
            <Setter Property="FontSize" Value="16" />
            <Setter Property="FontWeight" Value="DemiBold" />
            <!--<Setter Property="Effect">
                <Setter.Value>
                    <DropShadowEffect Color="Black" BlurRadius="10" ShadowDepth="5" Direction="45"/>
                </Setter.Value>
            </Setter>-->

        </Trigger>

    </Style.Triggers>
</Style>
```

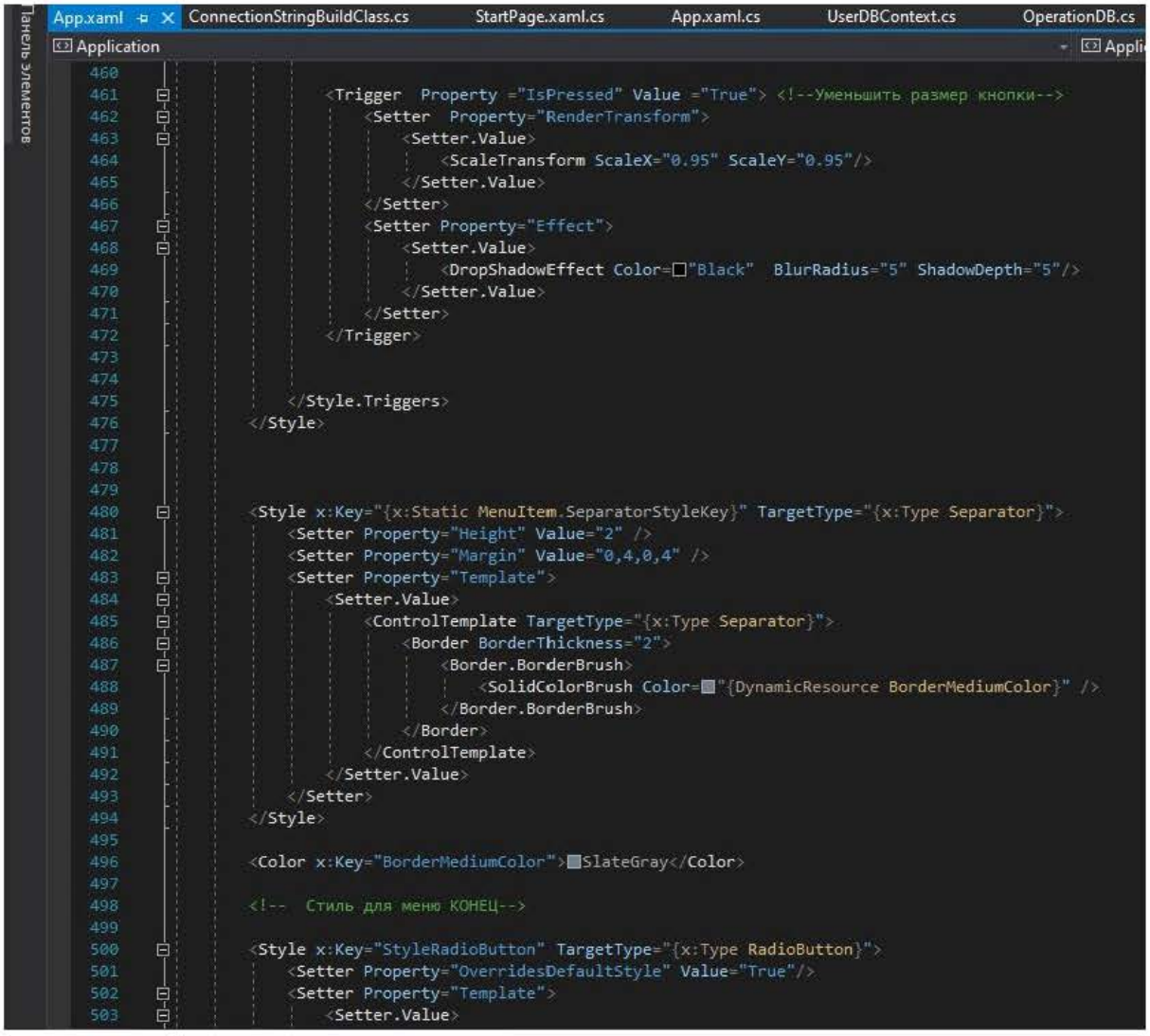

Панель элементов
App.xaml
ConnectionStringBuildClass.cs
StartPage.xaml.cs
App.xaml.cs
UserDBContext.cs
OperationDB.cs
MainWin
Application
Application
292
293 <DataTemplate x:Key="TabItemDataTemplate" >
294 <Border CornerRadius='4' BorderThickness='3'>
295 <Border.BorderBrush>
296 <LinearGradientBrush EndPoint='0,1'>
297 <LinearGradientBrush.GradientStops>
298 <GradientStop Offset='0' Color='#FFFFFFFF' />
299 <GradientStop Offset='1' Color='#FF777777' />
300 </LinearGradientBrush.GradientStops>
301 </LinearGradientBrush>
302 </Border.BorderBrush>
303 <Border.Background>
304 <LinearGradientBrush EndPoint='0,1'>
305 <LinearGradientBrush.GradientStops>
306 <GradientStop Offset='0' Color='#FF777777' />
307 <GradientStop Offset='1' Color='#FFFFFFFF' />
308 </LinearGradientBrush.GradientStops>
309 </LinearGradientBrush>
310 </Border.Background>
311 <ContentPresenter
312 ContentSource="Header"
313 HorizontalAlignment='Center'
314 VerticalAlignment='Center' />
315 </Border>
316 </DataTemplate>
317
318
319
320 <!-- Стиль для меню НАЧАЛО-->
321 <!--Этот шаблон используется. Чтобы выпадало подменю используется :
322 <Popup IsOpen="{TemplateBinding IsSubmenuOpen}" AllowsTransparency="True" PopupAnimation="Slide">
323 <StackPanel IsItemsHost="True" Background="LightSteelBlue" Focusable="True"/>
324 </Popup> -->
325 <ControlTemplate x:Key="{x:Static MenuItem.TopLevelHeaderTemplateKey}" TargetType="{x:Type MenuItem}">
326 <Grid>
327 <Border CornerRadius='4' BorderThickness='3'>
328 <Border.BorderBrush>
329 <LinearGradientBrush EndPoint='0,1'>
330 <LinearGradientBrush.GradientStops>
331 <GradientStop Offset='0' Color='#FFFFFFFF' />
332 <GradientStop Offset='1' Color='#FF777777' />
333 </LinearGradientBrush.GradientStops>
334 </LinearGradientBrush>
335 </Border.BorderBrush>



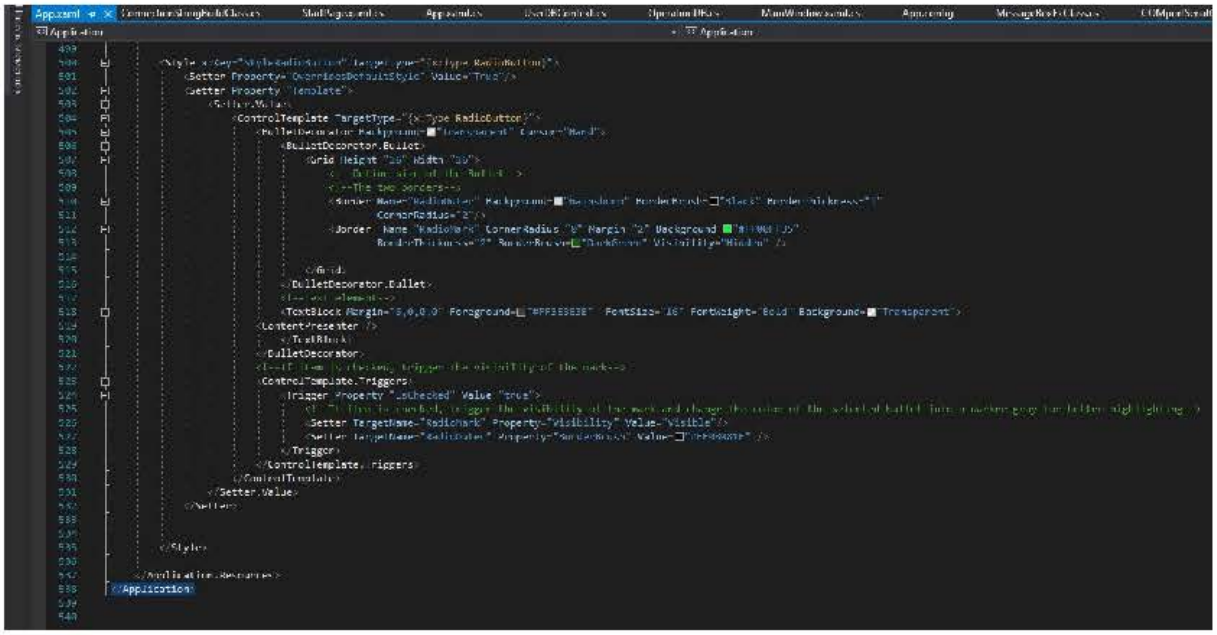
```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainW
Application                                                                                    Application
                    <LinearGradientBrush EndPoint='0,1'>
                        <LinearGradientBrush.GradientStops>
                            <GradientStop Offset='0' Color='#FF777777' />
                            <GradientStop Offset='1' Color='#FFFFFFFF' />
                        </LinearGradientBrush.GradientStops>
                    </LinearGradientBrush>
                </Border.Background>
                <ContentPresenter
                            ContentSource="Header"
                            HorizontalAlignment='Center'
                            VerticalAlignment='Center' />
            </Border>
        </ControlTemplate>

        <ControlTemplate x:Key="{x:Static MenuItem.SubmenuItemTemplateKey}" TargetType="{x:Type MenuItem}">
            <Border CornerRadius='4' BorderThickness='3'>
                <Border.BorderBrush>
                    <LinearGradientBrush EndPoint='0,1'>
                        <LinearGradientBrush.GradientStops>
                            <GradientStop Offset='0' Color='#FFFFFFFF' />
                            <GradientStop Offset='1' Color='#FF777777' />
                        </LinearGradientBrush.GradientStops>
                    </LinearGradientBrush>
                </Border.BorderBrush>
                <Border.Background>
                    <LinearGradientBrush EndPoint='0,1'>
                        <LinearGradientBrush.GradientStops>
                            <GradientStop Offset='0' Color='#FF777777' />
                            <GradientStop Offset='1' Color='#FFFFFFFF' />
                        </LinearGradientBrush.GradientStops>
                    </LinearGradientBrush>
                </Border.Background>
                <ContentPresenter
                            ContentSource="Header"
                            HorizontalAlignment='Center'
                            VerticalAlignment='Center' />
            </Border>
        </ControlTemplate>

        <ControlTemplate x:Key="{x:Static MenuItem.SubmenuHeaderTemplateKey}" TargetType="{x:Type MenuItem}">
            <Border CornerRadius='4' BorderThickness='3'>
                <Border.BorderBrush>
                    <LinearGradientBrush EndPoint='0,1'>
                        <LinearGradientBrush.GradientStops>
```

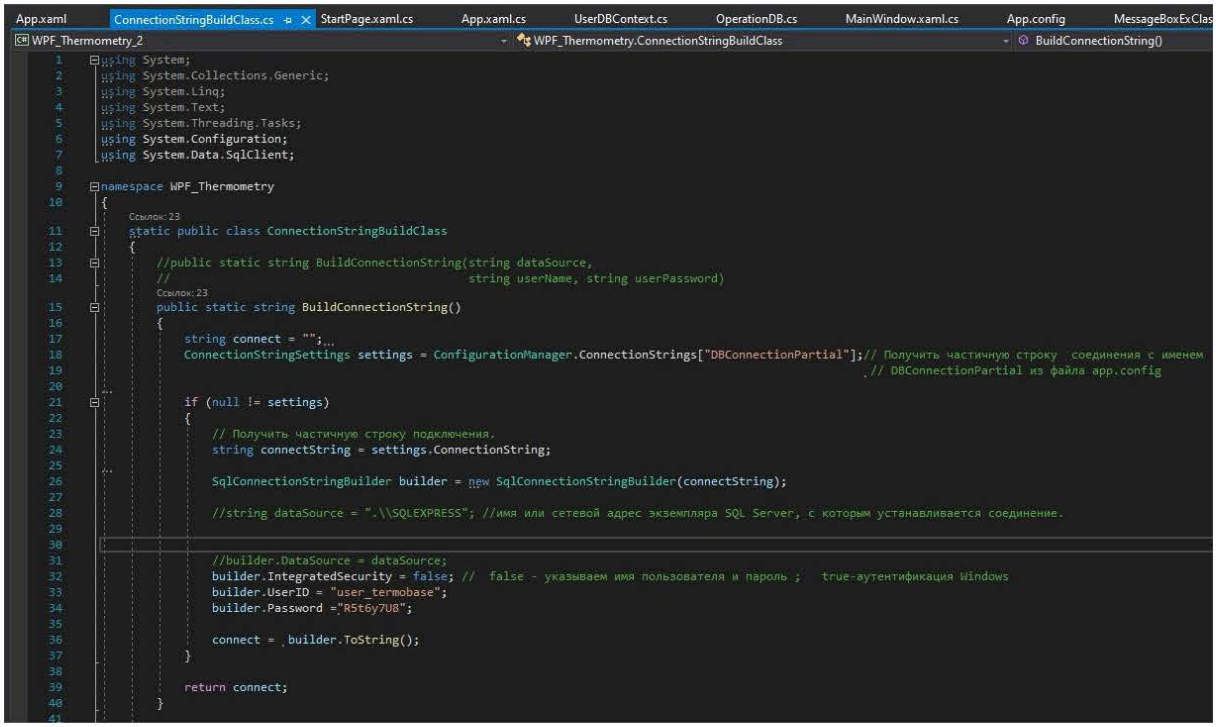


```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xam
Application                                                                                    Application
        <ControlTemplate x:Key="{x:Static MenuItem.SubmenuHeaderTemplateKey}" TargetType="{x:Type MenuItem}">
            <Border CornerRadius='4' BorderThickness='3'>
                <Border.BorderBrush>
                    <LinearGradientBrush EndPoint='0,1'>
                        <LinearGradientBrush.GradientStops>
                            <GradientStop Offset='0' Color='#FFFFFFFF' />
                            <GradientStop Offset='1' Color='#FF777777' />
                        </LinearGradientBrush.GradientStops>
                    </LinearGradientBrush>
                </Border.BorderBrush>
                <Border.Background>
                    <LinearGradientBrush EndPoint='0,1'>
                        <LinearGradientBrush.GradientStops>
                            <GradientStop Offset='0' Color='#FF777777' />
                            <GradientStop Offset='1' Color='#FFFFFFFF' />
                        </LinearGradientBrush.GradientStops>
                    </LinearGradientBrush>
                </Border.Background>
                <ContentPresenter
                            ContentSource="Header"
                            HorizontalAlignment='Center'
                            VerticalAlignment='Center' />
            </Border>
        </ControlTemplate>

        <Style x:Key="{x:Type MenuItem}" TargetType="{x:Type MenuItem}">
            <Setter Property="OverridesDefaultStyle" Value="True"/>
            <Style.Triggers>
                <Trigger Property="Role" Value="TopLevelHeader">
                    <Setter Property="Template" Value="{StaticResource {x:Static MenuItem.TopLevelHeaderTemplateKey}}"/>
                </Trigger>
                <Trigger Property="Role" Value="TopLevelItem">
                    <Setter Property="Template" Value="{StaticResource {x:Static MenuItem.TopLevelItemTemplateKey}}"/>
                </Trigger>
                <Trigger Property="Role" Value="SubmenuHeader">
                    <Setter Property="Template" Value="{StaticResource {x:Static MenuItem.SubmenuHeaderTemplateKey}}"/>
                </Trigger>
                <Trigger Property="Role" Value="SubmenuItem">
                    <Setter Property="Template" Value="{StaticResource {x:Static MenuItem.SubmenuItemTemplateKey}}"/>
                </Trigger>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Foreground" Value="Red" />
                </Trigger>
```

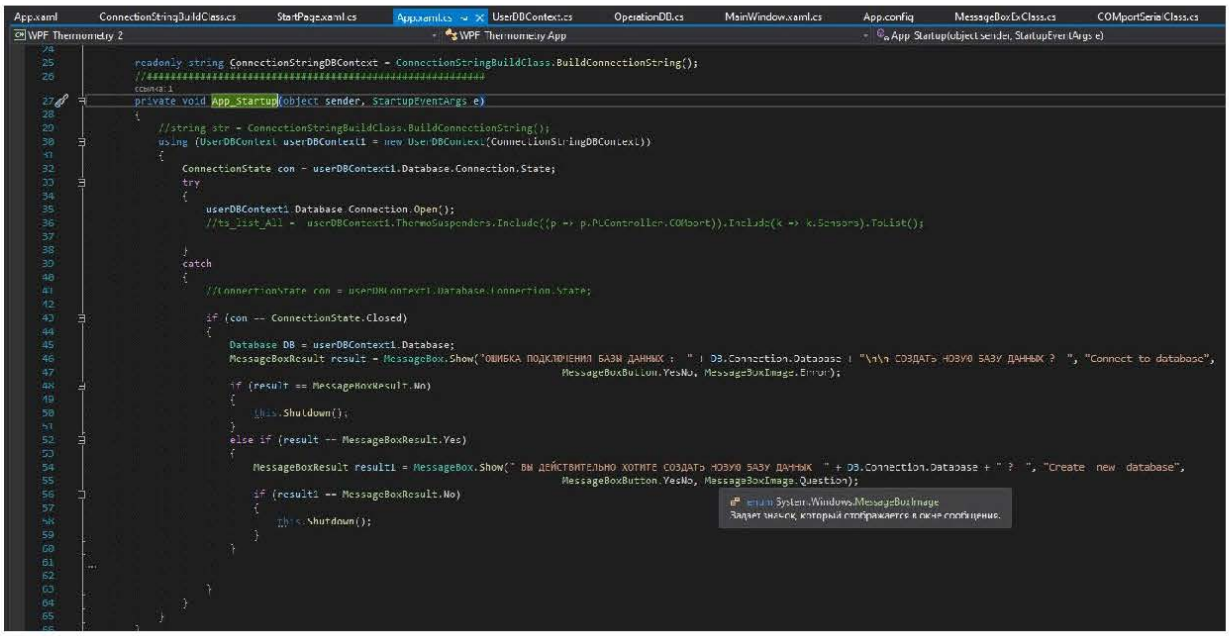


```
460
461                 <Trigger  Property ="IsPressed" Value ="True"> <!--Уменьшить размер кнопки-->
462                     <Setter  Property="RenderTransform">
463                         <Setter.Value>
464                             <ScaleTransform ScaleX="0.95" ScaleY="0.95"/>
465                         </Setter.Value>
466                     </Setter>
467                     <Setter Property="Effect">
468                         <Setter.Value>
469                             <DropShadowEffect Color="Black"  BlurRadius="5" ShadowDepth="5"/>
470                         </Setter.Value>
471                     </Setter>
472                 </Trigger>
473
474
475             </Style.Triggers>
476         </Style>
477
478
479
480         <Style x:Key="{x:Static MenuItem.SeparatorStyleKey}" TargetType="{x:Type Separator}">
481             <Setter Property="Height" Value="2" />
482             <Setter Property="Margin" Value="0,4,0,4" />
483             <Setter Property="Template">
484                 <Setter.Value>
485                     <ControlTemplate TargetType="{x:Type Separator}">
486                         <Border BorderThickness="2">
487                             <Border.BorderBrush>
488                                 <SolidColorBrush Color="{DynamicResource BorderMediumColor}" />
489                             </Border.BorderBrush>
490                         </Border>
491                     </ControlTemplate>
492                 </Setter.Value>
493             </Setter>
494         </Style>
495
496         <Color x:Key="BorderMediumColor">SlateGray</Color>
497
498         <!--  Стиль для меню КОНЕЦ-->
499
500         <Style x:Key="StyleRadioButton" TargetType="{x:Type RadioButton}">
501             <Setter Property="OverridesDefaultStyle" Value="True"/>
502             <Setter Property="Template">
503                 <Setter.Value>
```





```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs | App.config | MessageBoxExClas
WPF_Thermometry_2 | WPF_Thermometry.ConnectionStringBuildClass | BuildConnectionString()

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Configuration;
using System.Data.SqlClient;

namespace WPF_Thermometry
{
    Ссылок: 23
    static public class ConnectionStringBuildClass
    {
        //public static string BuildConnectionString(string dataSource,
        //                                           string userName, string userPassword)
        Ссылок: 23
        public static string BuildConnectionString()
        {
            string connect = "";...
            ConnectionStringSettings settings = ConfigurationManager.ConnectionStrings["DBConnectionPartial"];// Получить частичную строку  соединения с именем
                                                                                                               ,// DBConnectionPartial из файла app.config
...
            if (null != settings)
            {
                // Получить частичную строку подключения.
                string connectString = settings.ConnectionString;

...
                SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder(connectString);

                //string dataSource = ".\\SQLEXPRESS"; //имя или сетевой адрес экземпляра SQL Server, с которым устанавливается соединение.


                //builder.DataSource = dataSource;
                builder.IntegratedSecurity = false; //  false - указываем имя пользователя и пароль ;   true-аутентификация Windows
                builder.UserID = "user_termobase";
                builder.Password ="R5t6y7U8";

                connect = builder.ToString();
            }

            return connect;
        }
```



```
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs | App.config | MessageBoxExClass.cs | COMportSerialClass.cs
WPF_Thermometry_2 | WPF_Thermometry.App | App_Startup(object sender, StartupEventArgs e)

        readonly string ConnectionStringDBContext = ConnectionStringBuildClass.BuildConnectionString();
        //############################################################
        ссылка: 1
        private void App_Startup(object sender, StartupEventArgs e)
        {
            //string str = ConnectionStringBuildClass.BuildConnectionString();
            using (UserDBContext userDBContext1 = new UserDBContext(ConnectionStringDBContext))
            {
                ConnectionState con = userDBContext1.Database.Connection.State;
                try
                {
                    userDBContext1.Database.Connection.Open();
                    //ts_list_All =  userDBContext1.ThermoSuspenders.Include(p => p.PLController.COMport)).Include(k => k.Sensors).ToList();

                }
                catch
                {
                    //ConnectionState con = userDBContext1.Database.Connection.State;

                    if (con == ConnectionState.Closed)
                    {
                        Database DB = userDBContext1.Database;
                        MessageBoxResult result = MessageBox.Show("ОШИБКА ПОДКЛЮЧЕНИЯ БАЗЫ ДАННЫХ :  " + DB.Connection.Database + "\n\n СОЗДАТЬ НОВУЮ БАЗУ ДАННЫХ ?  ", "Connect to database",
                                                                  MessageBoxButton.YesNo, MessageBoxImage.Error);
                        if (result == MessageBoxResult.No)
                        {
                            this.Shutdown();
                        }
                        else if (result == MessageBoxResult.Yes)
                        {
                            MessageBoxResult result1 = MessageBox.Show(" ВЫ ДЕЙСТВИТЕЛЬНО ХОТИТЕ СОЗДАТЬ НОВУЮ БАЗУ ДАННЫХ  " + DB.Connection.Database + " ?  ", "Create  new  database",
                                                                       MessageBoxButton.YesNo, MessageBoxImage.Question);
                            if (result1 == MessageBoxResult.No)
                            {
                                this.Shutdown();
                            }
                        }
...

                    }
                }
            }
        }

System.Windows.MessageBoxImage
Задает значок, который отображается в окне сообщения.
```

App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs

WPF_Thermometry_2 — WPF_Thermometry.UserDBContext

```
#define DB_IF_MODEL_CHANGES   // если меняется контекст(модель) БД
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity;
//using System.Configuration;

namespace WPF_Thermometry
{
    // класс Контекста EntityFramework
    Ссылок: 99+
    public class UserDBContext : DbContext
    {
        //#############################################################################
        Ссылок: 0
        public UserDBContext() : base("name=DBConnectionPartial")
        {
            // Установить новый инициализатор

#if DB_IF_MODEL_CHANGES                                         //????? если установлена конфигурация решения  DEBUG

            // инициализатор, если меняется контекст(модель) БД
            Database.SetInitializer<UserDBContext>(new MyInitializerDBIfModelChanges());

#else                                                           // ????если установлена конфигурация решения  Release
            // инициализатор, если БД не существует
            Database.SetInitializer<UserDBContext>(new MyInitializerDBIfNotExist());
#endif
        }

        //#############################################################################
        Ссылок: 23
        public UserDBContext(string ConnectString) : base(ConnectString)
        {
            // Установить новый инициализатор

#if DB_IF_MODEL_CHANGES                                     // ????? если установлена конфигурация решения  DEBUG

            // инициализатор, если меняется контекст(модель) БД
            Database.SetInitializer<UserDBContext>(new MyInitializerDBIfModelChanges());

#else                                                       // ????? если установлена конфигурация решения  Release
```

App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs

WPF_Thermometry_2 — WPF_Thermometry.UserDBContext

```
#if DB_IF_MODEL_CHANGES                                     // ????? если установлена конфигурация решения  DEBUG

            // инициализатор, если меняется контекст(модель) БД
            Database.SetInitializer<UserDBContext>(new MyInitializerDBIfModelChanges());

#else                                                       // ????? если установлена конфигурация решения  Release

            // инициализатор, если БД не существует
            Database.SetInitializer<UserDBContext>(new MyInitializerDBIfNotExist()); // инициализатор, если БД не существует
#endif

        }
        //#############################################################################
        Ссылок: 21
        public DbSet<TypeSilo> TypeSilos { get; set; }
        Ссылок: 36
        public DbSet<TypeBunker> TypeBunkers { get; set; }
        Ссылок: 11
        public DbSet<TypeMeasureSystem> TypeMeasureSystems { get; set; }

        Ссылок: 13
        public DbSet<Company> Companies { get; set; }
        Ссылок: 52
        public DbSet<Silo> Silos { get; set; }
        Ссылок: 51
        public DbSet<Bunker> Bunkers { get; set; }

        Ссылок: 10
        public DbSet<COMport> COMports { get; set; }
        Ссылок: 13
        public DbSet<PLController> PLControllers { get; set; }
        Ссылок: 66
        public DbSet<ThermoSuspender> ThermoSuspenders { get; set; }

        public DbSet<Sensor> Sensors { get; set; }

        Ссылок: 25
        public DbSet<Culture> Cultures { get; set; }
        Ссылок: 8
        public DbSet<OutsideTemperature> OutsideTemperatures { get; set; }
        Ссылок: 8
        public DbSet<ColorLegend> ColorLegends { get; set; }
        Ссылок: 23
        public DbSet<Result> Results { get; set; }
        Ссылок: 0
        public DbSet<RealNumber> RealNumbers { get; set; }
```

App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs

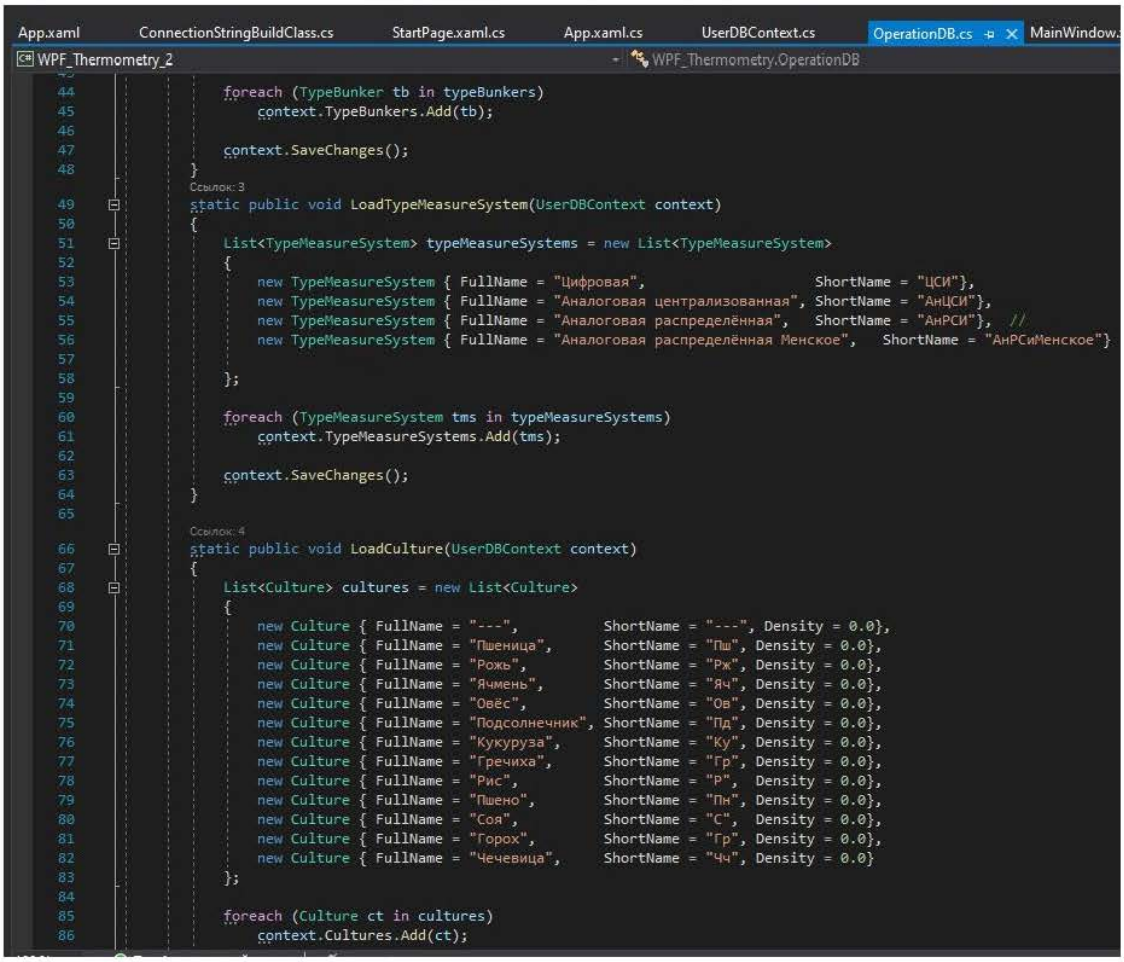
WPF_Thermometry_2 — WPF_Thermometry.UserDBContext

```
72  class MyInitializerDBIfNotExist : CreateDatabaseIfNotExists<UserDBContext> // создать БД , если она не существует
73  {
74      // В этом методе можно заполнить таблицу по умолчанию
        Ссылок: 0
75      protected override void Seed(UserDBContext context)
76      {
77
78          OperationDB.LoadTypeSilo(context);
79          OperationDB.LoadTypeBunker(context);
80          OperationDB.LoadTypeMeasureSystem(context);
81          OperationDB.LoadCulture(context);
82          OperationDB.LoadColorLegend(context);
83          OperationDB.LoadCOMport(context);
84
85      }
86  }
87  //###############################################
    Ссылок: 2
88  class MyInitializerDBIfModelChanges : DropCreateDatabaseIfModelChanges<UserDBContext> //  если меняется модель контекста
89  {
90      // В этом методе можно заполнить таблицу по умолчанию
        Ссылок: 0
91      protected override void Seed(UserDBContext context)
92      {
93          OperationDB.LoadTypeSilo(context);
94          OperationDB.LoadTypeBunker(context);
95          OperationDB.LoadTypeMeasureSystem(context);
96          OperationDB.LoadCulture(context);
97          OperationDB.LoadColorLegend(context);
98          OperationDB.LoadCOMport(context);
99      }
100 }
101
102 }
103
```

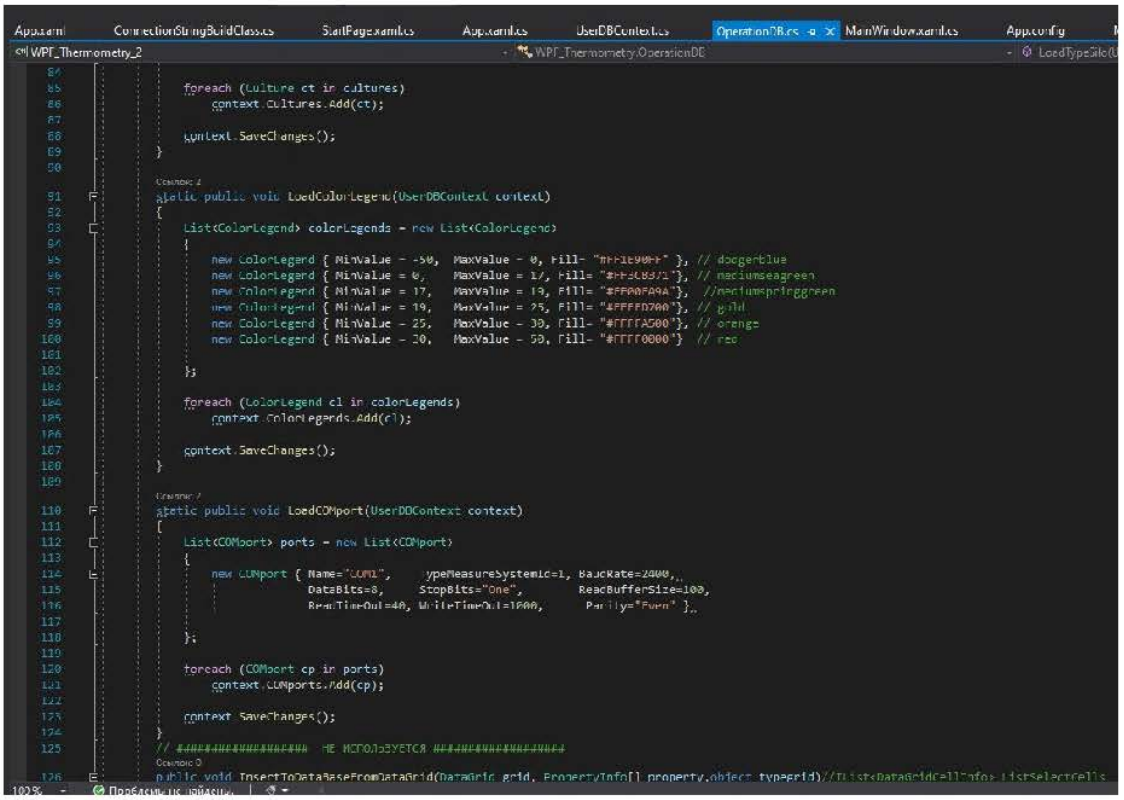
App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs

WPF_Thermometry_2 — WPF_Thermometry.OperationDB

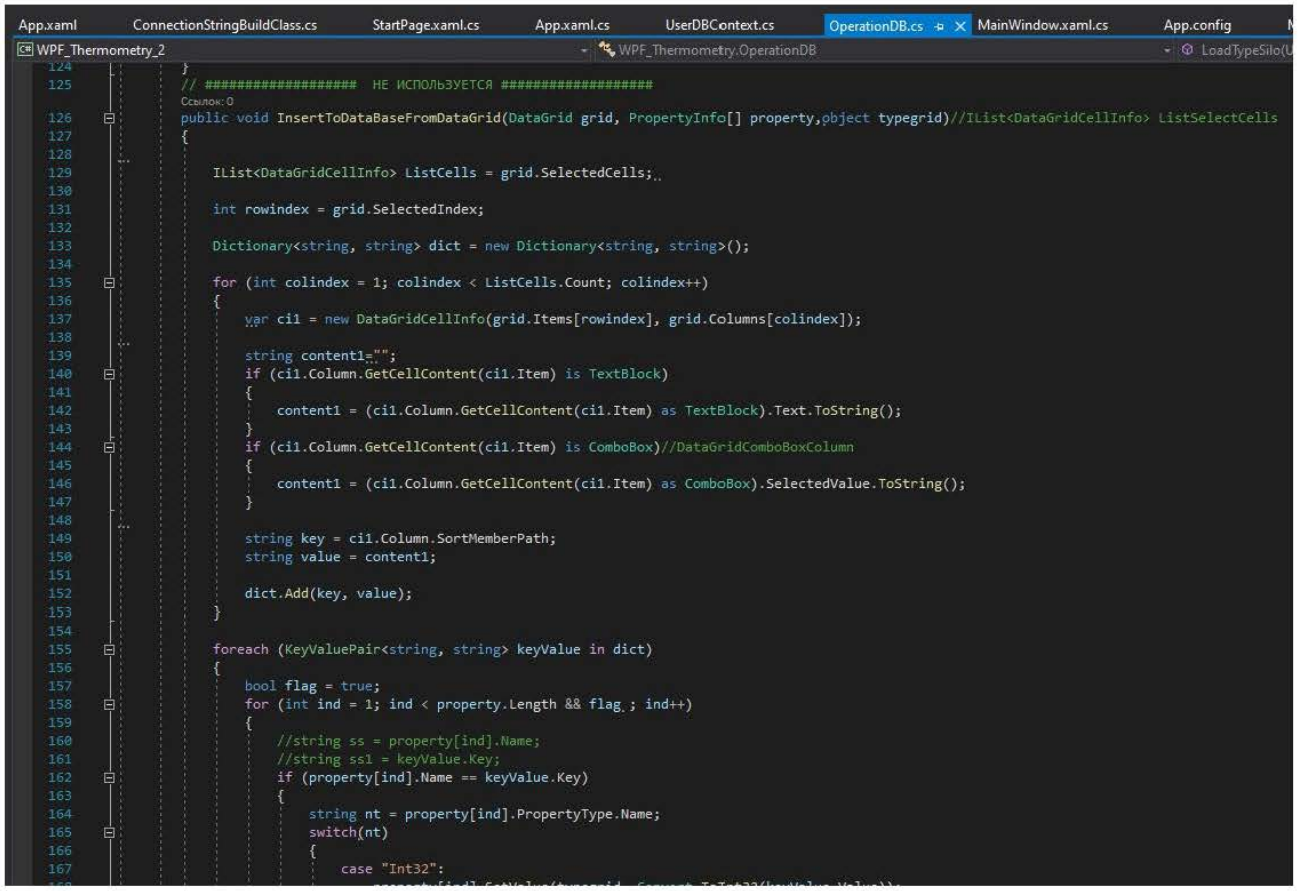
```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6   using System.Windows.Controls;
7   using System.Reflection;
8   using System.Windows;
9   using System.Windows.Data;
10  using System.Windows.Media;
11
12  namespace WPF_Thermometry
13  {
14      class OperationDB
15      {
            Ссылок: 3
16          static public void LoadTypeSilo(UserDBContext context)
17          {
18              List<TypeSilo> typeSilos = new List<TypeSilo>
19              {
20                  // new TypeSilo { FullName = "Круглый",       ShortName = "Крг"},
21                  new TypeSilo { FullName = "Группа Круглых",  ShortName = "Грп Крг"},
22                  new TypeSilo { FullName = "Прямоугольный", ShortName = "Прм"}
23              };
24
25              foreach (TypeSilo ts in typeSilos)
26                  context.TypeSilos.Add(ts);
27
28              context.SaveChanges();
29          }
30
            Ссылок: 3
31          static public void LoadTypeBunker(UserDBContext context)
32          {
33              List<TypeBunker> typeBunkers = new List<TypeBunker>
34              {
35                  new TypeBunker { FullName = "Круглый",          ShortName = "Крг"},
36                  new TypeBunker { FullName = "Прямоугольный",    ShortName = "Прм"},
37                  new TypeBunker { FullName = "Силкорпус",        ShortName =  "Слк" },
38                  new TypeBunker { FullName = "Звезда",           ShortName =  "*"},
39                  new TypeBunker { FullName = "ПолуЗвезда правая",ShortName =  "Полу* П"},
40                  new TypeBunker { FullName = "ПолуЗвезда левая", ShortName =  "Полу* Л"}
41
42              };
```

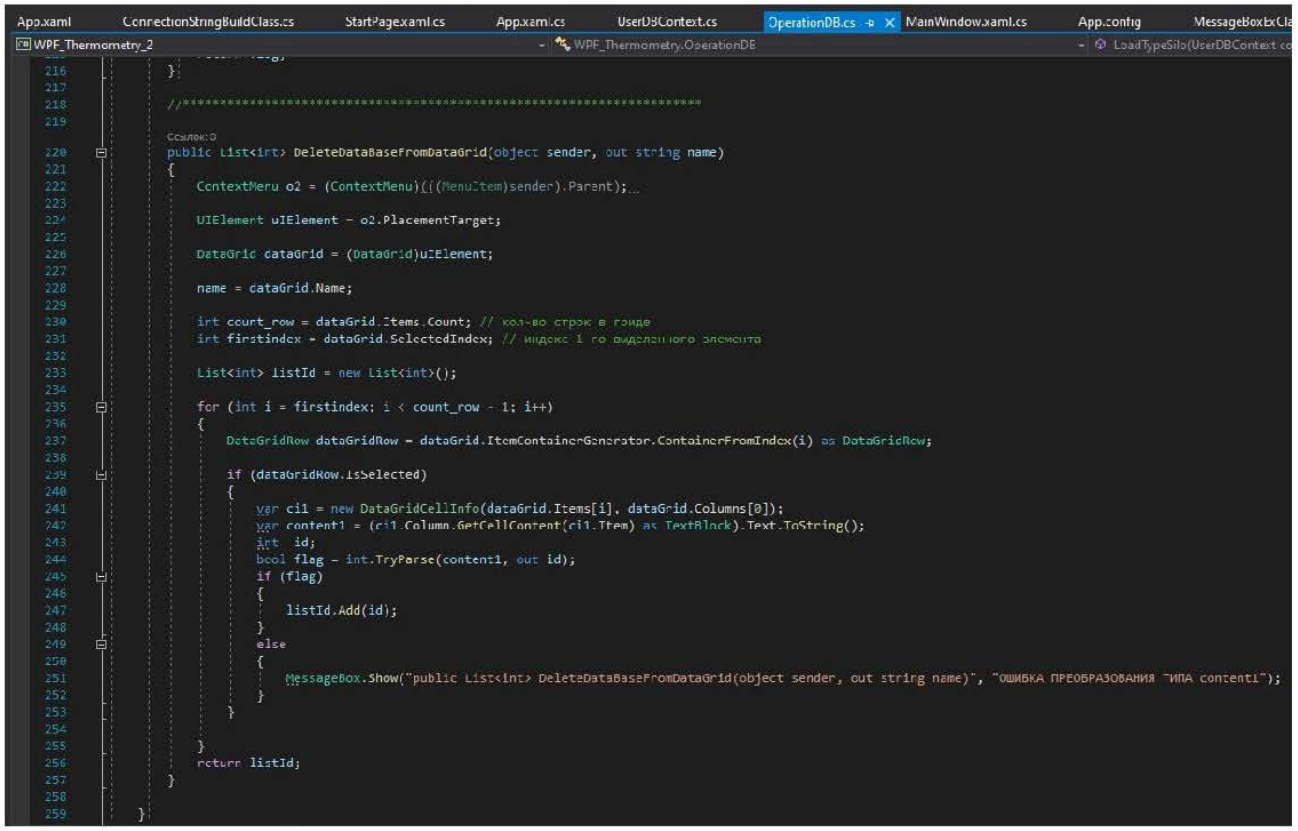


```
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.
WPF_Thermometry_2                                  WPF_Thermometry.OperationDB

            foreach (TypeBunker tb in typeBunkers)
                context.TypeBunkers.Add(tb);

            context.SaveChanges();
        }
        Ссылок: 3
        static public void LoadTypeMeasureSystem(UserDBContext context)
        {
            List<TypeMeasureSystem> typeMeasureSystems = new List<TypeMeasureSystem>
            {
                new TypeMeasureSystem { FullName = "Цифровая",                    ShortName = "ЦСИ"},
                new TypeMeasureSystem { FullName = "Аналоговая централизованная", ShortName = "АнЦСИ"},
                new TypeMeasureSystem { FullName = "Аналоговая распределённая",   ShortName = "АнРСИ"},  //
                new TypeMeasureSystem { FullName = "Аналоговая распределённая Менское",   ShortName = "АнРСиМенское"}

            };

            foreach (TypeMeasureSystem tms in typeMeasureSystems)
                context.TypeMeasureSystems.Add(tms);

            context.SaveChanges();
        }

        Ссылок: 4
        static public void LoadCulture(UserDBContext context)
        {
            List<Culture> cultures = new List<Culture>
            {
                new Culture { FullName = "---",          ShortName = "---", Density = 0.0},
                new Culture { FullName = "Пшеница",      ShortName = "Пш", Density = 0.0},
                new Culture { FullName = "Рожь",         ShortName = "Рж", Density = 0.0},
                new Culture { FullName = "Ячмень",       ShortName = "Яч", Density = 0.0},
                new Culture { FullName = "Овёс",         ShortName = "Ов", Density = 0.0},
                new Culture { FullName = "Подсолнечник", ShortName = "Пд", Density = 0.0},
                new Culture { FullName = "Кукуруза",     ShortName = "Ку", Density = 0.0},
                new Culture { FullName = "Гречиха",      ShortName = "Гр", Density = 0.0},
                new Culture { FullName = "Рис",          ShortName = "Р",  Density = 0.0},
                new Culture { FullName = "Пшено",        ShortName = "Пн", Density = 0.0},
                new Culture { FullName = "Соя",          ShortName = "С",  Density = 0.0},
                new Culture { FullName = "Горох",        ShortName = "Гр", Density = 0.0},
                new Culture { FullName = "Чечевица",     ShortName = "Чч", Density = 0.0}
            };

            foreach (Culture ct in cultures)
                context.Cultures.Add(ct);
```



```
App.xaml  ConnectionStringBuildClass.cs  StartPage.xaml.cs  App.xaml.cs  UserDBContext.cs  OperationDB.cs  MainWindow.xaml.cs  App.config
WPF_Thermometry_2                                  WPF_Thermometry.OperationDB                    LoadTypeSilo(

            foreach (Culture ct in cultures)
                context.Cultures.Add(ct);

            context.SaveChanges();
        }

        Ссылок: 2
        static public void LoadColorLegend(UserDBContext context)
        {
            List<ColorLegend> colorLegends = new List<ColorLegend>
            {
                new ColorLegend { MinValue = -50,  MaxValue = 0,  Fill= "#FF1E90FF" }, // dodgerblue
                new ColorLegend { MinValue = 0,    MaxValue = 17, Fill= "#FF3CB371"}, // mediumseagreen
                new ColorLegend { MinValue = 17,   MaxValue = 19, Fill= "#FF00FA9A"},  //mediumspringgreen
                new ColorLegend { MinValue = 19,   MaxValue = 25, Fill= "#FFFFD700"}, // gold
                new ColorLegend { MinValue = 25,   MaxValue = 30, Fill= "#FFFFA500"}, // orenge
                new ColorLegend { MinValue = 30,   MaxValue = 50, Fill= "#FFFF0000"}  // red

            };

            foreach (ColorLegend cl in colorLegends)
                context.ColorLegends.Add(cl);

            context.SaveChanges();
        }

        Ссылок: 2
        static public void LoadCOMport(UserDBContext context)
        {
            List<COMport> ports = new List<COMport>
            {
                new COMport { Name="COM1",     TypeMeasureSystemId=1, BaudRate=2400,
                              DataBits=8,      StopBits="One",        ReadBufferSize=100,
                              ReadTimeOut=40, WriteTimeOut=1000,      Parity="Even" },

            };

            foreach (COMport cp in ports)
                context.COMports.Add(cp);

            context.SaveChanges();
        }
        // ################  НЕ ИСПОЛЬЗУЕТСЯ ################
        Ссылок: 0
        public void InsertToDataBaseFromDataGrid(DataGrid grid, PropertyInfo[] property,object typegrid)//List<DataGridCellInfo> listSelectCells
```


App.xaml
ConnectionStringBuildClass.cs
StartPage.xaml.cs
App.xaml.cs
UserDBContext.cs
OperationDB.cs
MainWindow.xaml.cs
App.config
WPF_Thermometry_2
WPF_Thermometry.OperationDB
LoadTypeSilo(U
}
// ################## НЕ ИСПОЛЬЗУЕТСЯ ##################
Ссылок: 0
public void InsertToDataBaseFromDataGrid(DataGrid grid, PropertyInfo[] property,object typegrid)//IList<DataGridCellInfo> ListSelectCells
{
IList<DataGridCellInfo> ListCells = grid.SelectedCells;
int rowindex = grid.SelectedIndex;
Dictionary<string, string> dict = new Dictionary<string, string>();
for (int colindex = 1; colindex < ListCells.Count; colindex++)
{
var ci1 = new DataGridCellInfo(grid.Items[rowindex], grid.Columns[colindex]);
string content1="";
if (ci1.Column.GetCellContent(ci1.Item) is TextBlock)
{
content1 = (ci1.Column.GetCellContent(ci1.Item) as TextBlock).Text.ToString();
}
if (ci1.Column.GetCellContent(ci1.Item) is ComboBox)//DataGridComboBoxColumn
{
content1 = (ci1.Column.GetCellContent(ci1.Item) as ComboBox).SelectedValue.ToString();
}
string key = ci1.Column.SortMemberPath;
string value = content1;
dict.Add(key, value);
}
foreach (KeyValuePair<string, string> keyValue in dict)
{
bool flag = true;
for (int ind = 1; ind < property.Length && flag ; ind++)
{
//string ss = property[ind].Name;
//string ss1 = keyValue.Key;
if (property[ind].Name == keyValue.Key)
{
string nt = property[ind].PropertyType.Name;
switch(nt)
{
case "Int32":


App.xaml
ConnectionStringBuildClass.cs
StartPage.xaml.cs
App.xaml.cs
UserDBContext.cs
OperationDB.cs
MainWindow.xaml.cs
App.config
MessageBoxEx
WPF_Thermometry_2
WPF_Thermometry.OperationDB
LoadTypeSilo(UserDBContext
}
//***
Ссылок: 0
public List<int> DeleteDataBaseFromDataGrid(object sender, out string name)
{
ContextMenu o2 = (ContextMenu)((MenuItem)sender).Parent;
UIElement uIElement = o2.PlacementTarget;
DataGrid dataGrid = (DataGrid)uIElement;
name = dataGrid.Name;
int count_row = dataGrid.Items.Count; // кол-во строк в гриде
int firstindex = dataGrid.SelectedIndex; // индекс 1 го выделенного элемента
List<int> listId = new List<int>();
for (int i = firstindex; i < count_row - 1; i++)
{
DataGridRow dataGridRow = dataGrid.ItemContainerGenerator.ContainerFromIndex(i) as DataGridRow;
if (dataGridRow.IsSelected)
{
var ci1 = new DataGridCellInfo(dataGrid.Items[i], dataGrid.Columns[0]);
var content1 = (ci1.Column.GetCellContent(ci1.Item) as TextBlock).Text.ToString();
int id;
bool flag = int.TryParse(content1, out id);
if (flag)
{
listId.Add(id);
}
else
{
MessageBox.Show("public List<int> DeleteDataBaseFromDataGrid(object sender, out string name)", "ОШИБКА ПРЕОБРАЗОВАНИЯ ТИПА content1");
}
}
}
return listId;
}
}

App.xaml | ConnectionStringBuildClass.cs | StartPage.xaml.cs | App.xaml.cs | UserDBContext.cs | OperationDB.cs | MainWindow.xaml.cs

WPF_Thermometry_2 — WPF_Thermometry.MainWindow

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Data.Entity;
using System.ComponentModel;


namespace WPF_Thermometry
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    /// 
    Ссылок: 2
    public partial class MainWindow : NavigationWindow
    {
        Ссылок: 0
        public MainWindow()
        {
            SplashScreen splashScreen = new SplashScreen("Images\\screenSaverText.png"); // заставка
            splashScreen.Show(true);
            System.Threading.Thread.Sleep(900);
            splashScreen.Close(new TimeSpan(0,0,5));

            InitializeComponent();

        }
        //#############################################################################

    }
}

```