Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр
за освітньо-професійною програмою
«Комп'ютерні науки»
зі спеціальності
122 – Комп'ютерні науки

тема роботи:

«Розробка клієнтського застосунку системи неперервного контролю якості мінеральної сировини на конвеєрній стрічці»

Виконав студент гр. КН-21 _____ Єрьоменко Д.І.

Керівник _____ Гриценко А.М.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2025

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Бакалавр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *29* » *січня* 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

студентові групи КН-21 *Єрьоменко Даніілу Івановичу*

1. Тема кваліфікаційної роботи: *«Розробка клієнтського застосунку системи неперервного контролю якості мінеральної сировини на конвеєрній стрічці»*

затверджено наказом по університету № *254с* від *13.05.2025 р.*

2. Термін здачі кваліфікаційної роботи: *05.06.2025 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 54 с., додатки, презентація у Microsoft PowerPoint (13 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-2 к.т.н. Гриценко А. М.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*01.04.25*
2	*Розділ 1*	*05.04.25*
3	*Розділ 2*	*01.05.25*
4	*Висновки*	*25.05.25*
5	*Оформлення кваліфікаційної роботи*	*28.05.25*
6	*Підготовка презентації та графічного матеріалу*	*20.05.25*
7	*Підготовка доповіді до захисту*	*05.06.25*

6. Дата видачі завдання: *29.01.2025р.*

Керівник _____ / Гриценко А.М./

7. Запевнення: *Я, Єрьоменко Данііл Іванович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ / Єрьоменко Д.І./

АНОТАЦІЯ

Єрьоменко Д.І. Розробка клієнтського застосунку системи неперервного контролю якості на конвеєрній стрічці: кваліфікаційна робота бакалавра : 122 – Комп’ютерні науки. Кривий Ріг. Криворізький національний університет, 2025. 54 с.

У роботі розглянуто питання проектування та реалізації клієнтського програмного забезпечення для автоматизованої системи безперервного контролю якості мінеральної сировини «НАКС-ПК». Такий контроль є критично важливим у гірничовидобувній та збагачувальній промисловості, оскільки дозволяє оперативно відстежувати вміст заліза в потоці руди безпосередньо на конвеєрі.

Метою роботи є створення програмного продукту, що забезпечує підключення до віддаленої СКБД Firebird, обробку та візуалізацію вимірювальних даних, збереження налаштувань у локальній базі SQLite, а також перегляд історії вимірів.

Практичне значення одержаних результатів полягає у створенні стабільного та функціонального клієнтського інтерфейсу, що готовий до впровадження в умовах реального виробництва.

У першому розділі розглянуто загальну архітектуру та функціональність системи НАКС-ПК, досліджено структуру бази даних Firebird, а також виконано критичний огляд існуючих клієнтських програм, які дозволяють працювати з віддаленими базами даних за SQL-запитами.

Другий розділ присвячено процесу розробки застосунку: спроектовано архітектуру програмного забезпечення, розроблено алгоритми взаємодії з віддаленими БД, реалізовано засоби інтерфейсу користувача, а також протестована працездатність системи в реальних умовах. Застосунок побудовано з використанням платформи .NET (C#) у середовищі Visual
КЛЮЧОВІ СЛОВА: НАКС-ПК, FIREBIRD, КОНТРОЛЬ ЯКОСТІ, КЛІЄНТСЬКИЙ ЗАСТОСУНОК, SQL, C#, VISUAL STUDIO.

ЗМІСТ

ВСТУП

У сучасних умовах розвитку гірничодобувної та гірничопереробної галузей особливу актуальність набувають питання ефективного управління якістю сировини на всіх етапах виробничого циклу. Постійне підвищення вимог до чистоти та однорідності рудної маси зумовлює необхідність впровадження систем оперативного контролю, які здатні не лише здійснювати вимірювання фізико-хімічних параметрів матеріалу, а й оперативно реагувати на відхилення від заданих норм. Особливо важливо забезпечити такий контроль у потоці — безперервно, без зупинки виробничого процесу та з можливістю аналітичної оцінки тенденцій зміни якості.

Одним із найбільш критичних етапів контролю є момент, коли рудна маса транспортується конвеєрною стрічкою — саме тут можливо оперативно отримати повну картину про якість вхідної сировини, вміст корисних компонентів та інші параметри, що впливають на економічну ефективність виробництва. Використання ручного відбору проб або періодичних лабораторних аналізів у таких умовах не є достатнім — це не дозволяє отримати своєчасну інформацію та реагувати на зміни у реальному часі.

У відповідь на ці виклики створюються автоматизовані програмно-апаратні комплекси, які об'єднують вимірювальні прилади, алгоритми обробки даних та зручні інтерфейси для операторів. Одним із прикладів такої системи є комплекс НАКС-ПК, що призначений для неперервного контролю вмісту загального заліза в руді під час її транспортування по конвеєру. Система поєднує апаратну частину — набір датчиків, зокрема гамма-детектор та тензометричну платформу, з програмним забезпеченням, яке забезпечує зчитування, обробку, збереження та візуалізацію результатів.

Комплекс НАКС-ПК дозволяє здійснювати моніторинг якісного складу сировини в реальному часі, формувати базу вимірювань для подальшого аналізу, здійснювати калібрування за хімічними лабораторними даними та виводити інформацію у зручній для користувача формі. Це дає змогу

інженерно-технологічному персоналу своєчасно приймати рішення щодо зміни параметрів збагачення або сортування, знижуючи втрати та підвищуючи загальну ефективність виробництва.

Однак ефективне функціонування такого комплексу значною мірою залежить від якості реалізації його програмного забезпечення. Зокрема, клієнтський застосунок, який відповідає за інтерфейс користувача, має забезпечувати стабільну роботу з великою кількістю даних, підтримувати інтерактивну візуалізацію, дозволяти швидке перемикання між точками контролю, а також бути зручним у налаштуванні та обслуговуванні. Важливу роль відіграє також архітектура системи зберігання даних — реляційна база, що забезпечує структурований доступ до історії вимірювань, параметрів калібрування та конфігураційних налаштувань [1].

У рамках кваліфікаційної роботи поставлено завдання розробити клієнтський застосунок системи НАКС-ПК, що забезпечує підключення до віддалених серверних баз даних Firebird, відображення поточних вимірювань у зручному форматі, а також надає функції перегляду історії, управління калібруванням і налаштування точок контролю. Особливу увагу приділено архітектурі програмного продукту, вибору засобів реалізації, формату зберігання локальних налаштувань та забезпеченню надійної роботи в умовах промислового середовища.

Таким чином, дана кваліфікаційна робота є прикладом реального прикладного ІТ-проекту, спрямованого на покращення технологічного контролю у гірничодобувній галузі шляхом удосконалення та розширення компонентів програмного забезпечення існуючої системи.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО СТАНУ НЕПЕРЕРВНОГО КОНТРОЛЮ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Дослідження функціональності системи НАКС-ПК

Програмно-апаратний комплекс «НАКС-ПК» розроблений для здійснення автоматизованого, безперервного контролю якості мінеральної сировини безпосередньо в умовах технологічного процесу — на конвеєрній стрічці гірничодобувного або збагачувального підприємства. Завдяки впровадженню такого комплексу забезпечується високоточний моніторинг ключових параметрів корисної копалини ще в процесі транспортування на подальші етапи переробки.

Основна функціональна роль комплексу полягає в оперативному аналізі вмісту загального заліза (Fe заг.) в рудній масі, що переміщується по конвеєру. Вимірювання здійснюються у реальному часі із використанням ядерно-фізичних методів та тензометричних вагових систем, дані з яких синхронізуються для одержання достовірного результату.

Апаратна частина комплексу відповідає за збір первинної інформації (інтенсивність гамма-випромінювання, масова подача тощо), та їх передачу до обчислювальної системи, після чого програмна частина виконує низку обчислювальних операцій:

- запит, прийом і попередню обробку сигналів;
- визначення фізико-хімічних показників відповідно до калібрувальних характеристик;
- збереження результатів у централізовану базу даних, що є джерелом для їх подальшого аналізу;
- візуалізацію результатів у вигляді таблиць, графіків та звітів;
- виконання діагностичних процедур, включаючи контроль стабільності роботи системи та метрологічну перевірку.

Таким чином, НАКС-ПК не лише виконує функції вимірювального пристрою, але і є повноцінним елементом інформаційно-аналітичної системи контролю якості, яка дозволяє приймати рішення на основі об’єктивних і достовірних даних у режимі, наближеному до реального часу.

Програмно-апаратний комплекс системи НАКС-ПК реалізує такі функції:

1) Безперервне вимірювання параметрів потоку руди. НАКС-ПК здійснює реєстрацію поточних значень масової подачі матеріалу на конвеєрі (т/год) та інтенсивності відбитого гамма-випромінювання. В основі лежить використання гамма-датчиків і тензометричних платформ.

2) Розрахунок вмісту заліза загального (Fe заг.). Програма на основі калібрувальних залежностей перетворює отримані фізичні значення (інтенсивність, масу) у вміст загального заліза в рудному потоці, забезпечуючи точність у промислових умовах.

3) Автоматичне збереження даних у базі даних. Усі результати вимірювань зберігаються у централізованій базі даних (Firebird). Дані структуруються відповідно до точок контролю, часу вимірювання, калібрувань та параметрів середовища.

4) Візуалізація даних у реальному часі. Відображає оновлену інформацію з заданою періодичністю у вигляді таблиць, числових індикаторів і графіків, забезпечуючи зручний моніторинг для оператора (рисунок 1.1).

5) Перегляд та аналіз історії вимірів. Система дає змогу переглядати архів вимірювань за будь-який період, здійснювати фільтрацію за точками контролю, обчислювати середні значення, порівнювати інтервали та формувати тренди (рисунок. 1.2).

6) Побудова графіків та формування звітів. Користувач може генерувати графіки зміни вмісту Fe заг. за вибраний проміжок часу, а також формувати табличні звіти, придатні для друку чи подальшого аналізу (рисунок. 1.3).

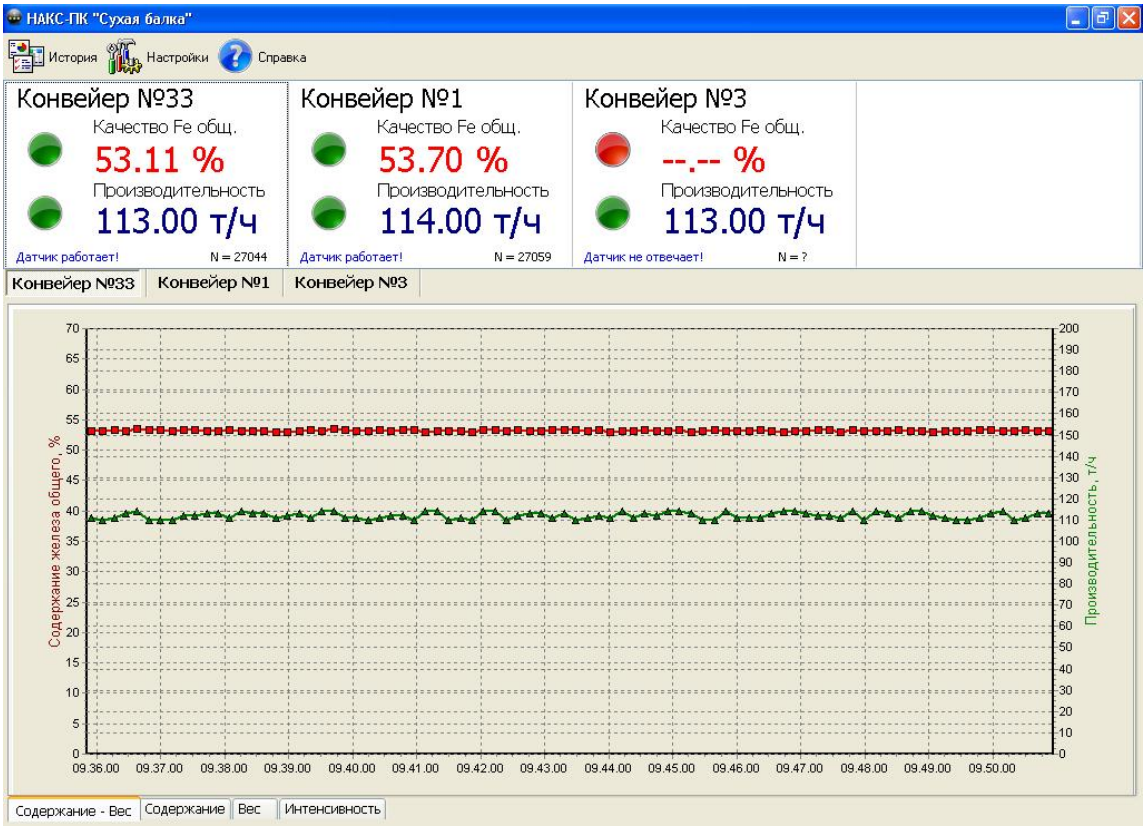

Рисунок 1.1 – Інтерфейс Головного вікна програмного забезпечення.

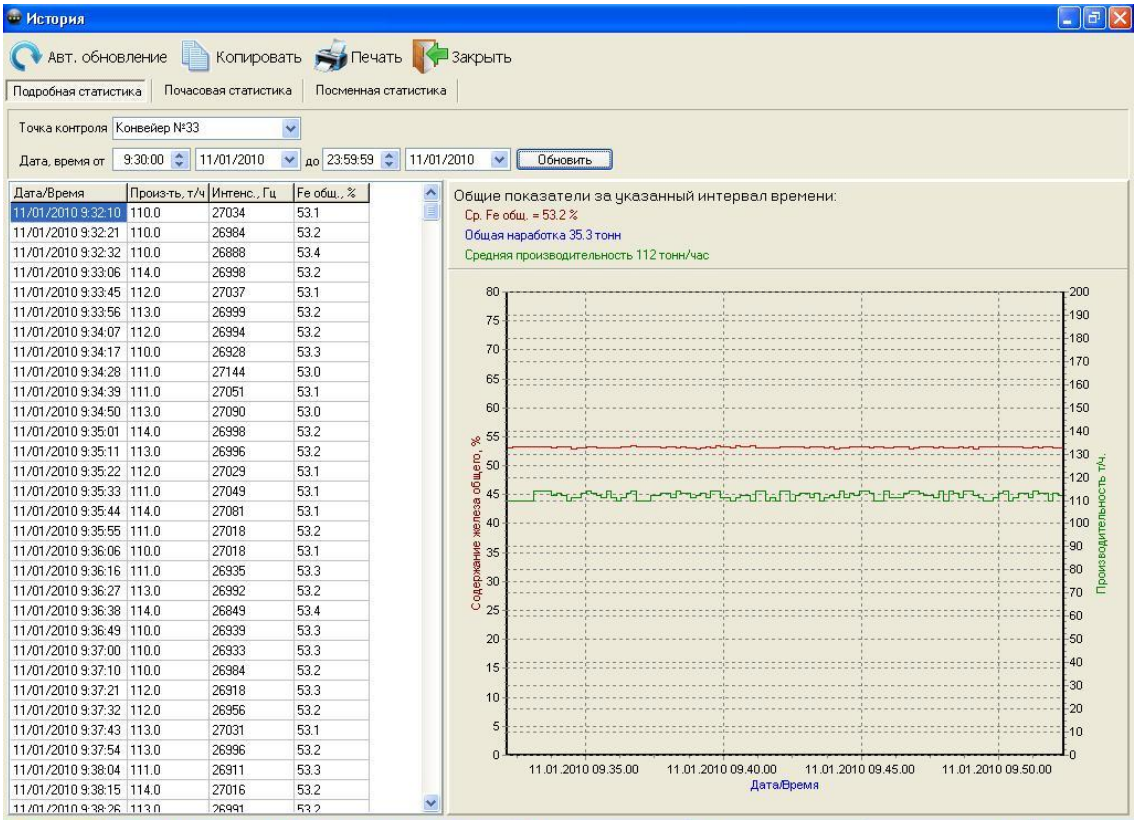

Рисунок 1.2 – Інтерфейс вікна перегляду історії вимірювань по заданим параметрам.

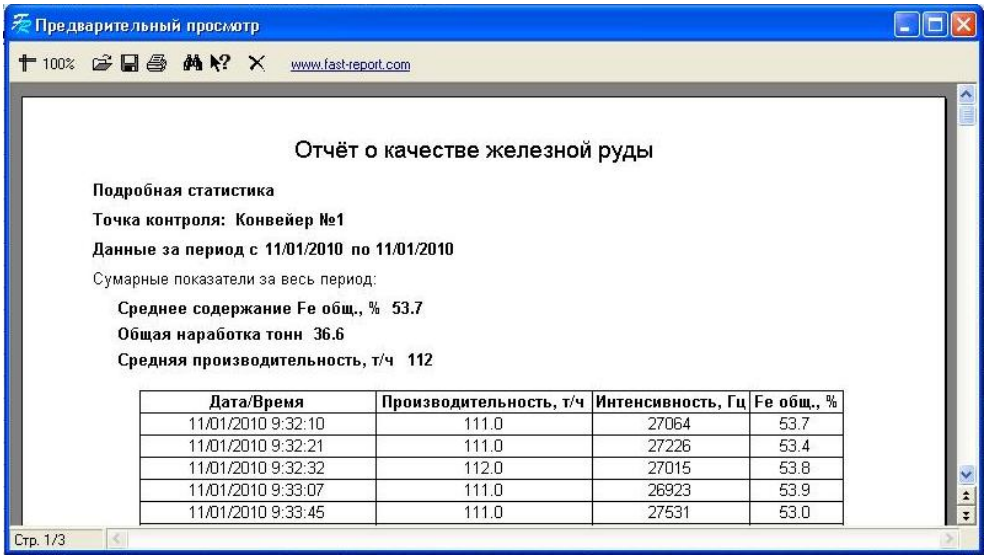

Рисунок 1.3 – Інтерфейс вікна з переглядом сформованого звіту та підготовкою до переносу на паперові носії.

7) Калібрування та коригування параметрів. Через модуль налаштувань можна задавати або редагувати параметри калібрувальних функцій, зберігати нові конфігурації, а також оновлювати значення реперів для компенсації зміни зовнішніх умов (рисунок 1.4).

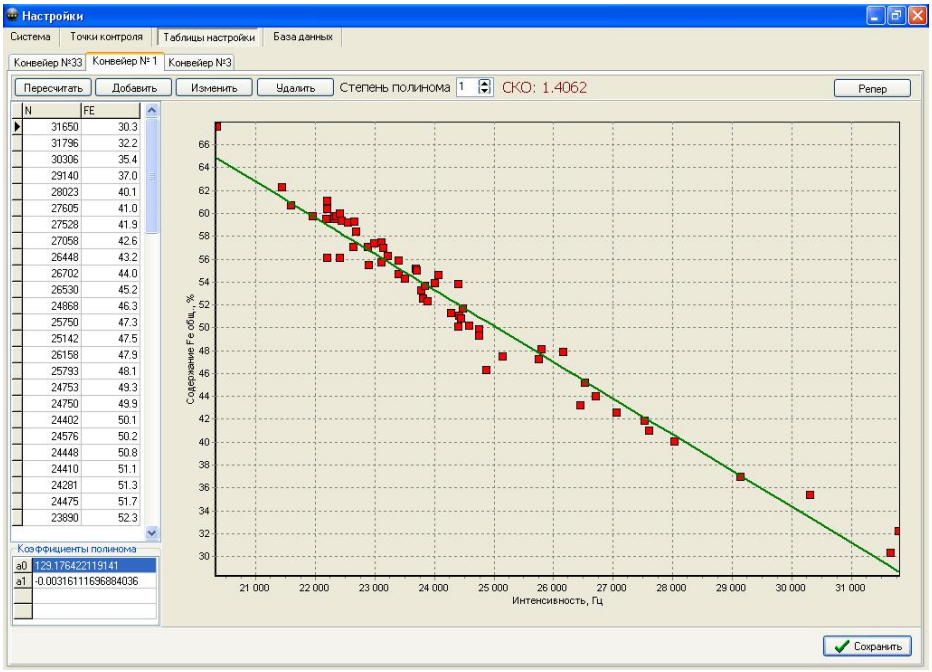

Рисунок 1.4 – Інтерфейс вікна калібрування точок контролю.

8) Діагностика роботи пристрою та контроль технічного стану. Інтерфейс передбачає відображення повідомлень про помилки, перевищення контрольних меж, нестабільність живлення або втрату зв'язку. Це забезпечує вчасне виявлення збоїв та підвищує надійність системи.

9) Метрологічна перевірка та верифікація результатів. Передбачена функція перевірки точності пристрою за допомогою еталонних зразків (реперів). Результати верифікації фіксуються у базі для подальшої оцінки точності роботи.

10) Гнучке налаштування точок контролю та параметрів системи. Конфігураційні дані зберігаються у локальній у відповідній таблиці БД, що дозволяє зручно керувати підключеннями до точок контролю, та налаштовувати їх представлення оператору системи (рисунок. 1.5).

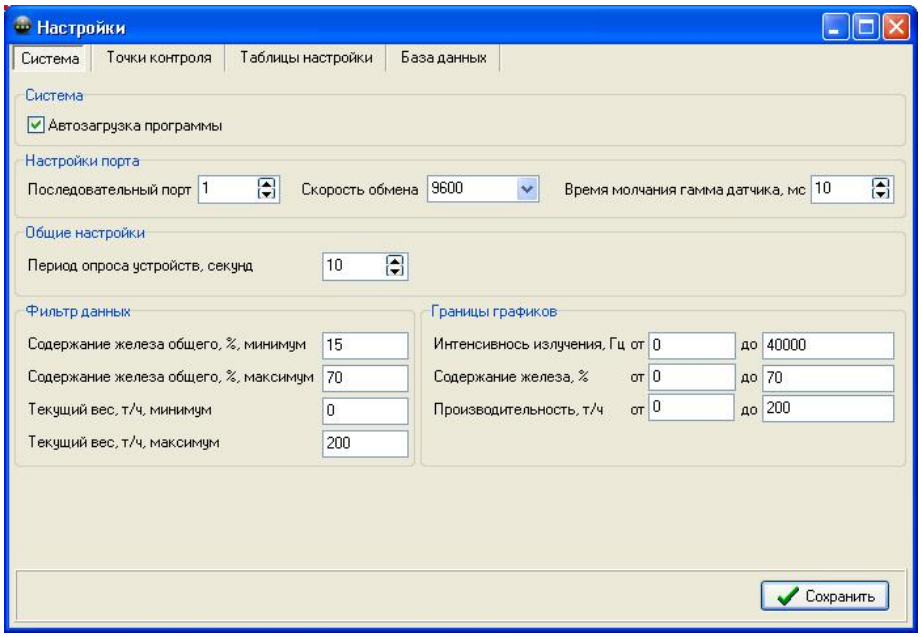

Рисунок 1.5 – Інтерфейс вікна налаштувань системи (вкладка «Система»).

1.2 Дослідження архітектури системи НАКС-ПК

Програмне забезпечення комплексу НАКС-ПК побудовано за принципами модульної, багаторівневої архітектури з розподілом

відповідальностей між окремими компонентами. Застосунок умовно поділяється на три основні рівні [2]:

1. Рівень презентації (Presentation Layer). Цей рівень реалізовано на основі WinForms. Він відповідає за відображення даних, взаємодію з користувачем, відправлення команд та отримання результатів у зручному візуальному форматі. Основні форми інтерфейсу:

- MainForm – головне вікно, де відображаються поточні значення параметрів (Fe заг., масова подача, інтенсивність), а також графіки в реальному часі.
- HistoryForm – вікно перегляду історичних даних: користувач вибирає часовий діапазон, а система будує графіки та таблиці.
- SettingsForm – інтерфейс для керування параметрами підключення до Firebird, редагування локальних налаштувань, встановлення інтервалу оновлення тощо.
- CalibrEditForm – форма редагування калібрувальних параметрів (A, B, C, D, R).

Усі форми мають чітке функціональне розмежування і взаємодіють з іншими модулями винятково через API внутрішніх класів.

2. Рівень бізнес-логіки (Business Logic Layer). Цей рівень реалізує основні функціональні сценарії застосунку, включаючи обробку даних, реалізацію алгоритмів, взаємодію з протоколами, формування запитів та обробку виключень. Він відокремлений від інтерфейсу та виконує роль посередника між UI та базами даних. Основні компоненти:

- DataController – керує підключенням до віддаленого сервера Firebird. Формує SQL-запити, обробляє результати, повертає їх у вигляді об'єктів.
- HistoryManager – виконує агрегацію історичних записів, будує вибірки за датами, розраховує середні значення, готує вхідні дані для побудови графіків.

- CalibrProcessor – реалізує логіку розрахунку вмісту Fe заг. на основі поліномів (A, B, C, D) та інтенсивності гамма-випромінювання.
- SettingsManager – відповідає за читання/запис локальних параметрів із бази SQLite, що дозволяє динамічно змінювати конфігурацію програми без перекомпіляції.
- TimerService – реалізує періодичне опитування Firebird-сервера для отримання актуальних даних згідно з встановленим інтервалом.
- ErrorHandler – централізовано обробляє винятки, веде лог помилок, забезпечує повідомлення в інтерфейсі користувача.

Уся логіка обробки винесена в окремі класи з мінімальною кількістю залежностей для полегшення тестування.

3. Рівень доступу до даних (Data Access Layer). Основне сховище даних, яке містить результати вимірювань (MEASURE), опис точок контролю (DEVICE), калібрування (CALIBR) та дані калібрувальних таблиць (CALIBR_DATA).

Для взаємодії з Firebird використовується. SQL-запити формуються через параметризовані шаблони, що запобігає SQL-ін’єкціям [1].

Інтерфейс WinForms взаємодіє з бізнес-логікою через публічні методи класів-контролерів. Логіка отримує дані з DAL, обробляє їх, а результат повертає до інтерфейсу для відображення. Залежності між модулями реалізовано через інверсію керування (IoC) для гнучкості, що у майбутньому дозволяє адаптувати код до інших баз даних або платформ.

Серед особливості реалізації варто зазначити, що:

- асинхронні методи використовуються для уникнення блокування UI під час виконання SQL-запитів;
- логіка оновлення графіків реалізована через System.Timers.Timer, що гарантує точне виконання навіть при зміні навантаження;
- динамічне оновлення даних дозволяє контролювати ситуацію на виробництві в майже реальному часі (затримка лише у межах інтервалу опитування);

підтримка декількох точок контролю одночасно — через організацію черги запитів, та процес усереднення показників підлеглими пристроями між інтервалами опитування.

1.3 Структура бази даних

Функціонування системи НАКС-ПК базується на обробці, збереженні та подальшому аналізі значних обсягів технологічної інформації, що надходить у реальному часі від апаратних сенсорів. Для забезпечення цілісності, достовірності та зручності доступу до цих даних у системі використовується централізована реляційна база даних, побудована на основі СКБД Firebird [3].

На рисунку 1.6 наведено структуру бази даних, що використовується у комплексі НАКС-ПК, з переліком основних таблиць, їх полів, типів даних та функціонального призначення (таблиці 1.1 – 1.4).

Таблиця 1.1 - Структура таблиці DEVICE

Назва поля	Тип даних	Опис
ID_DEVICE	INTEGER	Первинний ключ, унікальний ідентифікатор точки
DEVICE_TITLE	VARCHAR(100)	Назва або опис точки контролю
ADR_GAMMA	INTEGER	Адреса гамма-датчика в Modbus
ADR_WEIGHT	INTEGER	Адреса вагового тензодатчика (вагів)

Таблиця 1.4 - Структура таблиці CALIBR_DATA

Назва поля	Тип даних	Опис
ID_CALIBR_DATA	INTEGER	Первинний ключ
N	DECIMAL(10,2)	Інтенсивність гамма-випромінювання
FE	DECIMAL(10,2)	Показник хімічного аналізу — вміст Fe заг. у пробі
ID_DEVICE	INTEGER	Зовнішній ключ до таблиці DEVICE

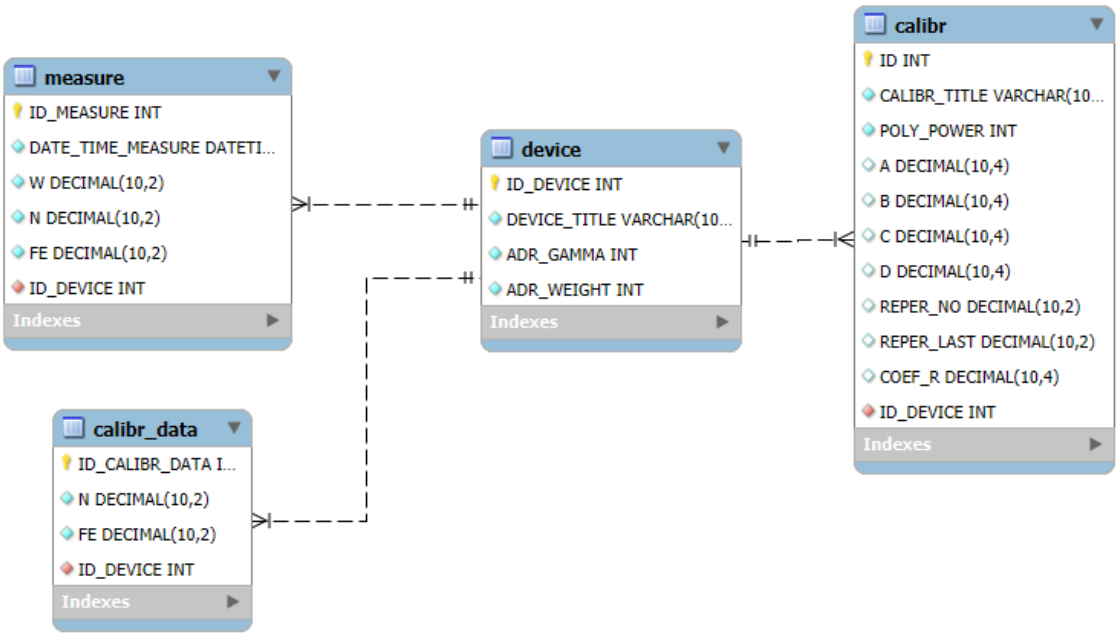

Рисунок 1.6 – ER-діаграма бази даних.

Таблиця 1.2 - Структура таблиці MEASURE

Назва поля	Тип даних	Опис
ID_MEASURE	INTEGER	Первинний ключ, унікальний ID вимірювання
DATE_TIME_MEASURE	TIMESTAMP	Дата та час проведеного вимірювання
W	DECIMAL(10,2)	Миттєвий потік (маса) в тонн/год під час вимірювання
N	DECIMAL(10,2)	Інтенсивність гамма-випромінювання
FE	DECIMAL(10,2)	Вміст заліза загального (Fe заг.)
ID_DEVICE	INTEGER	Зовнішній ключ до таблиці DEVICE

База даних виступає ядром серверної частини програмного забезпечення. Вона структурує інформацію про контрольні точки, результати вимірювань, калібрувальні параметри та еталонні значення для математичного моделювання. Така організація дозволяє ефективно здійснювати фільтрацію,

агрегацію, візуалізацію та аналітичну обробку інформації за критеріями часу, місце та стани виробничих процесів.

Таблиця 1.3 - Структура таблиці CALIBR

Назва поля	Тип даних	Опис
ID	INTEGER	Первинний ключ
CALIBR_TITLE	VARCHAR(100)	Назва калібрування
POLY_POWER	INTEGER	Ступінь аппроксимуючого полінома
A	DECIMAL(10,4)	Коефіцієнт A у поліномі
B	DECIMAL(10,4)	Коефіцієнт B у поліномі
C	DECIMAL(10,4)	Коефіцієнт C у поліномі
D	DECIMAL(10,4)	Коефіцієнт D у поліномі
REPER_NO	DECIMAL(10,2)	Значення від реперного зразка (еталону) при початковій калібровці
REPER_LAST	DECIMAL(10,2)	Останнє зареєстроване значення репера
COEF_R	DECIMAL(10,4)	Поправковий коефіцієнт, пов'язаний з температурною компенсацією
ID_DEVICE	INTEGER	Зовнішній ключ до таблиці DEVICE

1.4 Критичний огляд існуючих програмних рішень для підключення до віддалених СКБД та візуалізації даних за SQL-запитами

У процесі розробки клієнтського програмного забезпечення, що взаємодіє з віддаленими системами керування базами даних (СКБД), важливою задачею є вибір ефективного засобу перегляду, аналізу та візуалізації інформації, збереженої у цих БД. Для оцінки доцільності створення власного програмного забезпечення розглянемо переваги та недоліки найбільш поширених універсальних засобів, що дозволяють підключатися до віддалених

СКБД, виконувати попередньо сформульовані SQL-запити та представляти дані у табличному або графічному вигляді.

Існуючі програмні рішення для підключення та візуалізації даних з віддалених систем керування базами даних (СКБД), зокрема Firebird, мають ряд як переваг, так і обмежень. Розглянемо деякі з них.

DBeaver — це багатоплатформовий інструмент з відкритим кодом для адміністрування баз даних. Він підтримує підключення до Firebird через JDBC-драйвер. Інтерфейс користувача дозволяє виконувати SQL-запити, переглядати структуру бази даних, а також візуалізувати дані у вигляді графіків та діаграм наведено на рисунку 1.7.

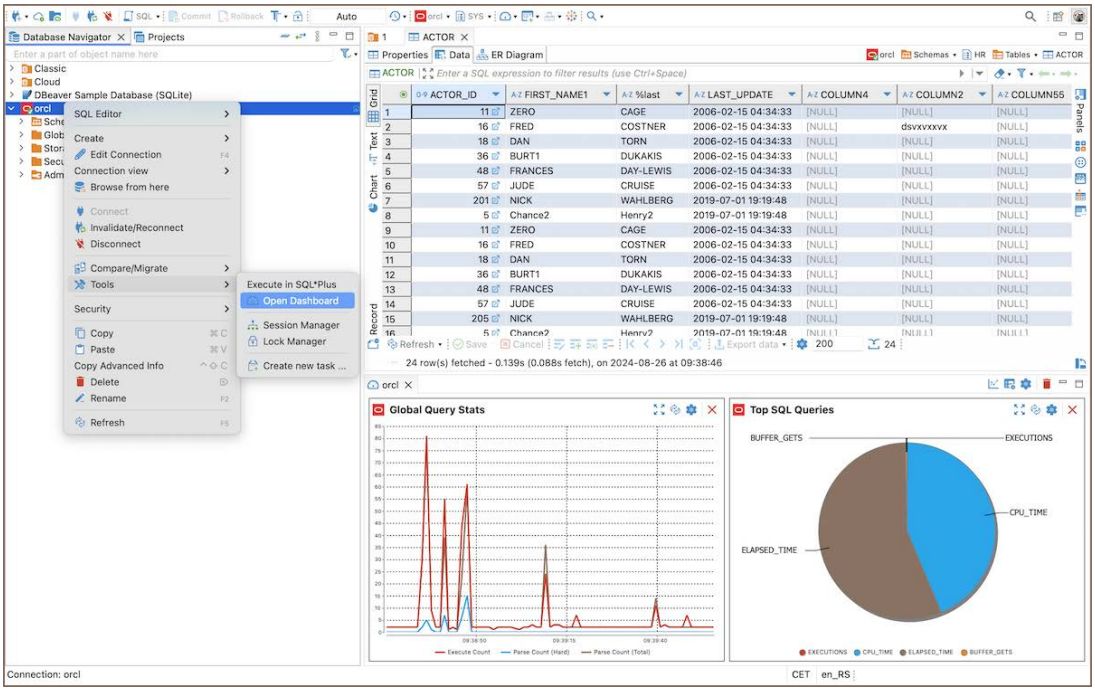

Рисунок 1.7 – Інтерфейс DBeaver з відображенням даних у табличному та графічному вигляді.

Переваги:

- підтримка великої кількості СКБД, включаючи Firebird;
- можливість візуалізації даних та побудови ER-діаграм;
- інтуїтивно зрозумілий інтерфейс.

Недоліки:

- відсутність спеціалізованих функцій для моніторингу в реальному часі;
- обмежені можливості кастомізації під специфічні виробничі потреби;

Beekeeper Studio — сучасний GUI-клієнт для роботи з базами даних, включаючи Firebird. Він надає можливість виконання SQL-запитів, перегляду та редагування даних, а також підтримує вкладки для одночасної роботи з кількома запитами, його графічний інтерфейс наведено на рисунку 1.8.

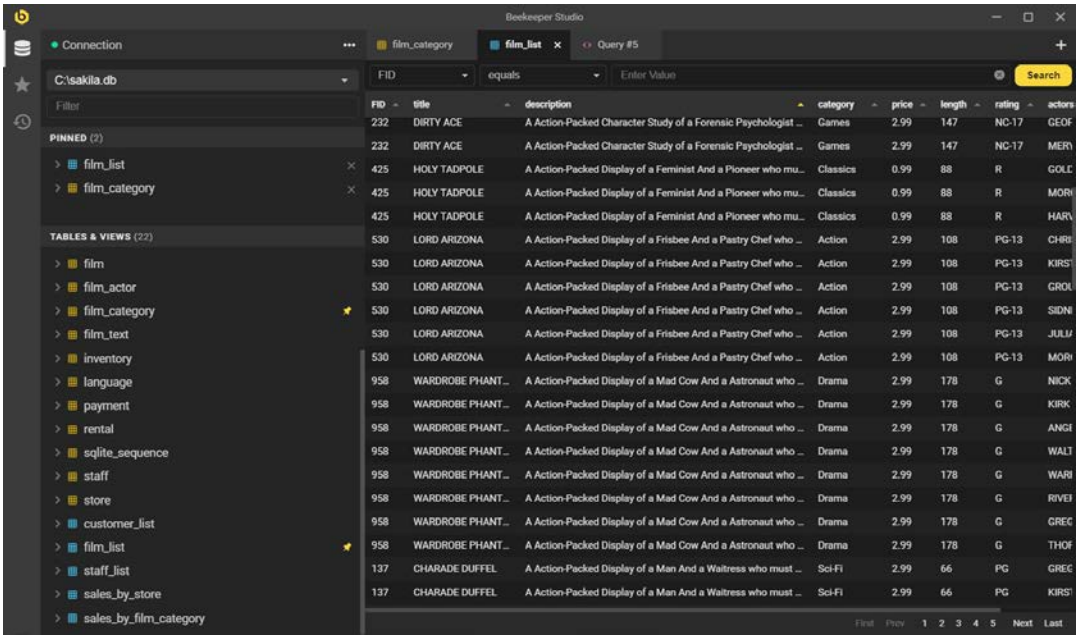

Рисунок 1.8 – Інтерфейс Beekeeper Studio з відображенням даних у табличному вигляді.

Переваги:

- кросплатформеність (Windows, macOS, Linux);
- простий та сучасний інтерфейс.
- підтримка збереження історії запитів.

Недоліки:

- відсутність розширених функцій для візуалізації даних.
- обмежені можливості для автоматизації процесів моніторингу.

RazorSQL — комерційний інструмент для роботи з різними СКБД, включаючи Firebird. Він пропонує розширені можливості для редагування, виконання запитів та візуалізації даних візуальний інтерфейс якого наведено на рисунку . 1.9.

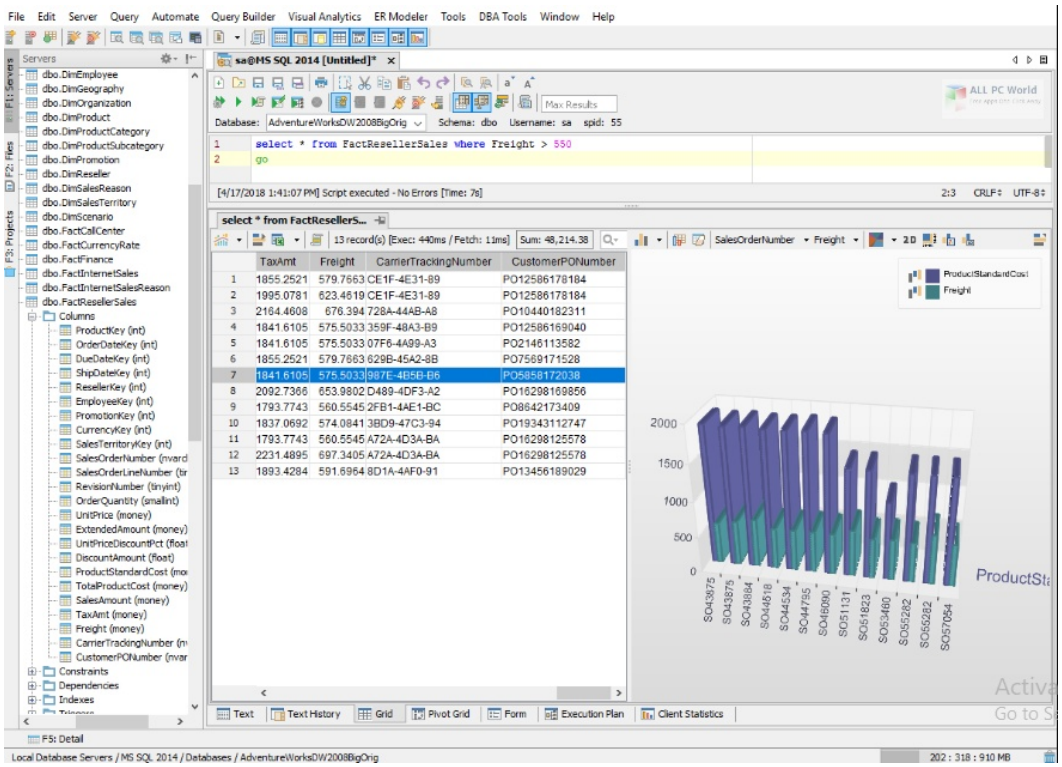

Рисунок 1.9 – Інтерфейс RazorSQL з відображенням даних у табличному та графічному вигляді.

Переваги:

- потужний SQL-редактор з автодоповненням.
- можливість імпорту та експорту даних у різних форматах.
- підтримка побудови графіків на основі результатів запитів.

Недоліки:

- платна ліцензія.
- відсутність спеціалізованих функцій для безперервного моніторингу виробничих процесів.

Metabase – це інструмент бізнес-аналітики з відкритим кодом, орієнтований на побудову звітів та візуалізацію даних (рисунок 1.10). Він дозволяє користувачам без знань SQL створювати прості запити через інтерфейс конструктора, а також підтримує написання SQL-запитів вручну для складніших запитів. Основною перевагою є легкість створення інтерактивних дашбордів.

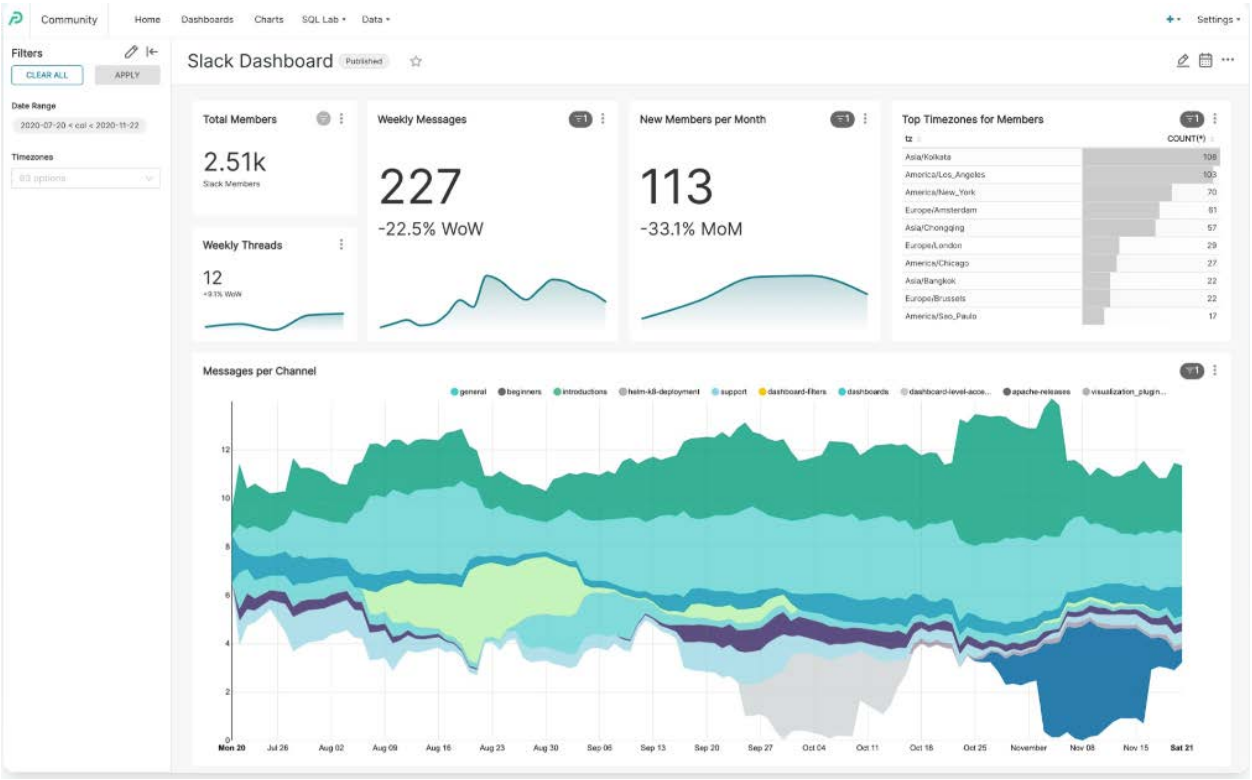

Рисунок 1.10 – Dashboard Metabase з відображенням даних в різних формах.

Переваги:

- простий веб-інтерфейс;
- підтримка запитів без програмування;
- інтерактивні графіки та таблиці.

Недоліки:

- обмежена підтримка Firebird (необхідні сторонні конектори);

- відсутність гнучкого управління відображенням специфічних виробничих параметрів;
- не підтримує глибоку інтеграцію із сторонніми системами.

Grafana — потужна платформа для моніторингу, яка в основному використовується з time-series даними, але підтримує підключення до SQL-джерел за допомогою плагінів. Вона дозволяє створювати інформаційні панелі з автоматичним оновленням даних у режимі реального часу, приклад реалізації наведено на рисунку 1.11.

Рисунок 1.11 – Dashboard Grafana з графічним представленням даних.

Переваги:

- чудова підтримка графіків і візуалізацій;
- автоматичне оновлення даних;
- можливість інтеграції з іншими системами сповіщень (email, Telegram тощо).

Недоліки:

- низька інтеграція із Firebird;
- необхідна складна конфігурація;
- потребує окремого веб-сервера.

DataGrip — комерційний продукт компанії JetBrains. Він створений для розробників і аналітиків, які працюють з різними базами даних. DataGrip пропонує інтелектуальне автодоповнення запитів, детальний перегляд структури БД, потужний редактор SQL, можливість перевірки запитів на помилки до виконання, наглядний візуальний інтерфейс користувача, вигляд якого наведено на рисунку 1.12.

Переваги:

- професійний багатофункціональний інтерфейс;
- підтримка декількох підключень одночасно;
- гнучкі інструменти для роботи зі складними SQL-запитами.

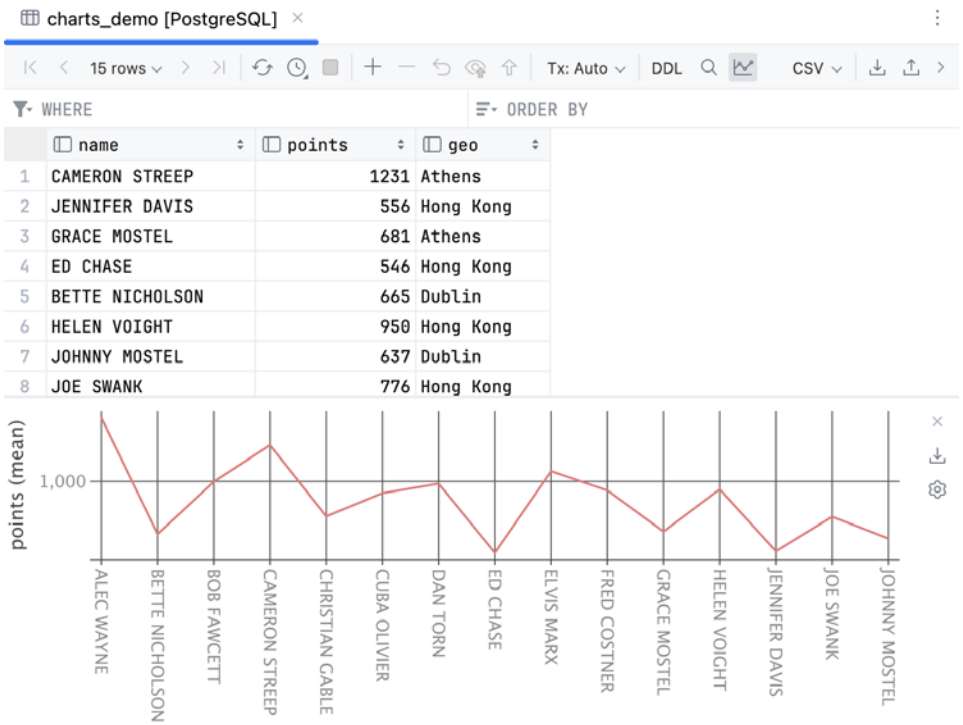

Рисунок 1.12 – Інтерфейс DataGrip з відображенням даних у табличному та графічному вигляді.

Недоліки:

- платна ліцензія;
- висока складність інтерфейсу для звичайного оператора;
- відсутність вбудованих засобів постійного оновлення даних у реальному часі.

Висновки по розділу

Зазначені інструменти надають базові можливості для роботи з СКБД Firebird, вони не повністю відповідають специфічним вимогам операторів гірничовидобувної та гірничопереробної галузей, що використовують систему НАКС-ПК, зокрема:

- відсутність реального часу: Більшість інструментів не підтримують безперервний моніторинг даних у реальному часі, що є критичним для контролю якості сировини на конвеєрі.
- обмежена кастомізація: Інструменти не дозволяють легко адаптувати інтерфейс та функціональність під специфічні виробничі процеси та вимоги.
- відсутність інтеграції з іншими системами: Існуючі рішення не забезпечують легку інтеграцію з іншими виробничими системами та обладнанням.

Зважаючи на вищезазначені обмеження, виникає потреба у розробці спеціалізованого програмного забезпечення, яке б враховувало специфіку гірничовидобувної та гірничопереробної галузей. Таке ПЗ повинно забезпечувати:

- безперервний моніторинг: Збір та обробку даних у реальному часі з можливістю оперативного реагування на зміни.
- гнучку візуалізацію: Побудову графіків та діаграм, що відображають ключові параметри якості сировини.
- інтеграцію з обладнанням: Можливість взаємодії з виробничим обладнанням для автоматичного збору даних.
- кастомізацію: Адаптацію інтерфейсу та функціоналу під конкретні виробничі потреби.

Розглянуті інструменти добре підходять для внутрішнього використання спеціалістами ІТ або аналітики, проте вони не завжди забезпечують потреби операторів у промислових умовах, де важливе значення мають: простота

інтерфейсу, специфічна логіка представлення даних, автоматичне оновлення і повна інтеграція з процесом. У такому контексті розробка спеціалізованого застосунку на мові C# дозволяє врахувати вимоги конкретного виробництва, підтримати роботу з Firebird, реалізувати цикл оновлення, гнучку конфігурацію точок контролю та простий інтерфейс для користувача без глибоких знань SQL.

РОЗДІЛ 2
ПРОЕКТУВАННЯ, РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Проектування архітектури та вибір засобів реалізації

Структура взаємодії програмного забезпечення абонентських пунктів та віддалених серверів (c1, c2, …, cn) з програмним забезпеченням серверного типу NaksPC.exe, яке здійснює збір, перетворення , локальну візуалізацію та зберігання інформації у БД з підключених до них точок контролю (ТК) наведено на рисунку 2.1 [2].

Робота програмного забезпечення сервера організовано таким чином, що запис даних від програми NaksPC.exe в БД здійснюється за допомогою мережевих компонентів і СКБД Firebird. Завдяки тому, що СКБД Firebird є багатоклієнтською, можливе одночасне використання кількох абонентських програм [3].

Функціональна схема системи з одним сервером та одним абонентським пунктом наведено на рисунку. 2.2 і складається з:

- Графічних компонентів, що здійснюють перетворення інформації для візуалізації сигналів для пристроїв, що здійснюють відображення.

- Бібліотека SQLite, реалізує всю взаємодію з файлом БД шляхом організації абстрактного представлення БД та програмних функцій[2].

- Локальна БД, файл бази даних, в якому зберігається список точок контролю та інформація про них

- Порти В/В, за допомогою яких оператор абонентського пункту здійснює роботу з застосунком (перехід між віконними інтерфейсами, введення діапазону відображення, вибір поточного каналу для відображення тощо);

- Мережеві компоненти, за допомогою яких здійснюється підключення та робота у локальної мережі підприємства, та через які виконується підключення до віддалених серверів.

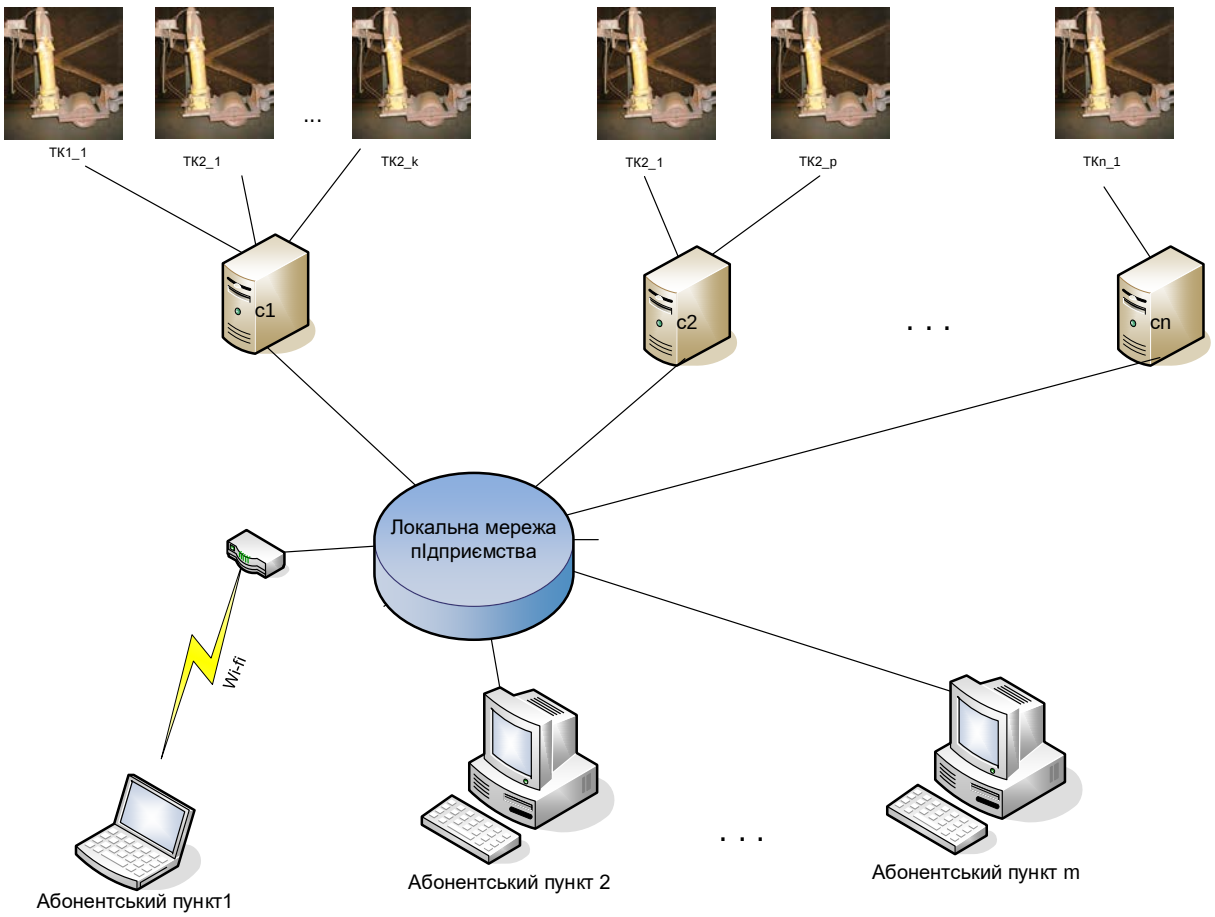

Рисунок 2.1 - Структна схема взаємодії пристроїв в системі контролю НАКС.

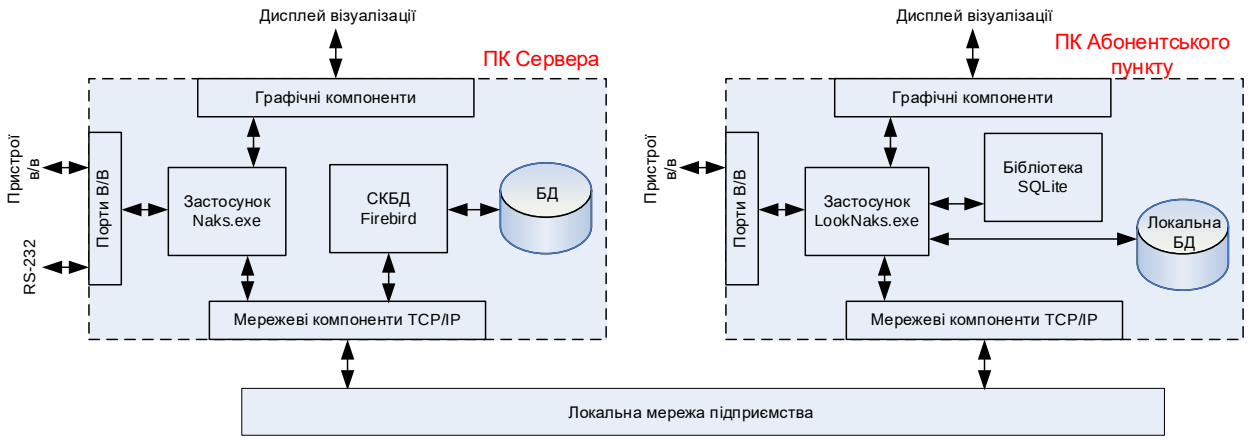

Рисунок 2.2 - Функціональна схема системи.

Застосунок LookNAKS - програмне забезпечення, здійснює при запуску завантаження інформації про віддалені сервери з Локальної БД. Через встановлений інтервал часу, за допомогою локальної мережі та мережевих компонентів ПК абонентського пункту циклічно відправляє запити даних до

віддалених серверів та відображення їх (даних) за допомогою графічних компонентів. У режимі перегляду історії вимірів, за допомогою портів В/В оператор вводить параметри, з яких застосунком формується запит до віддаленого сервера, та графічне представлення отриманих даних користувачу з розрахунком статистичних показників.

Розробка клієнтського застосунку NaksPC Look для моніторингу якості мінеральної сировини на конвеєрній стрічці передбачає побудову гнучкої, масштабованої та надійної архітектури. Враховуючи специфіку промислового середовища, зокрема потребу у стійкому підключенні до віддалених серверів, візуалізації у реальному часі та збереженні локальних конфігурацій, було прийнято рішення реалізувати програмне забезпечення на основі трирівневої архітектури (three-tier architecture). Такий підхід дозволяє досягти високого рівня відокремлення логіки програми, даних і користувацького інтерфейсу, що вкрай важливо для промислових систем з потенційним масштабуванням.

Перший рівень архітектури відповідає за взаємодію з оператором або інженером-користувачем системи. Для його реалізації використано технологію WinForms, яка забезпечує створення класичного десктопного GUI з можливістю динамічного оновлення даних та інтеграції графічних елементів. Вибір WinForms обґрунтовано його простотою у розробці, стабільністю роботи та сумісністю з системами Windows, які традиційно використовуються в промисловості.

Головне вікно програми (MainForm) дозволяє переглядати список точок контролю, які відображаються у вигляді таблиць та графічними елементами. Для кожної точки у реальному часі оновлюються показники: загальний вміст заліза (Fe), маса потоку (W), інтенсивність (N). Дані не тільки відображаються чисельно, але й виводяться на графіки з часовою шкалою, що значно підвищує зручність контролю та візуального аналізу.

Додаткові форми реалізують функції перегляду історії та налаштування параметрів. Форма історії замірів (HystoryForm) дозволяє обрати інтервал часу та побачити відповідну динаміку параметрів. Вікно налаштувань

(SettingsForm) дозволяє змінювати IP-адреси, шляхи до файлів бази даних, діапазони графіків, а також налаштовувати періодичність оновлення.

Другий рівень архітектури відповідає за логіку обробки даних, що надходять із зовнішніх джерел. Він пов’язує вхідні дані з елементами інтерфейсу, керує подіями оновлення та формує запити до бази даних.

Логіку реалізовано через набір C#-класів:

- DataController — координує підключення до Firebird, формує та виконує SQL-запити, обробляє результати. Він працює асинхронно, щоб не блокувати головний потік додатку.
- HistoryManager — забезпечує формування вибірок історичних даних, обчислення середніх значень, підготовку результатів для графічного відображення.
- SettingsManager — працює з локальною базою SQLite, зчитуючи та зберігаючи дані про точки контролю.
- TimerService — відповідає за реалізацію періодичних звернень до Firebird. Користувач може вказати інтервал у налаштуваннях, після чого система автоматично оновлює графіки.
- ErrorHandler — централізовано обробляє помилки, зокрема проблеми з мережею, втрату підключення або некоректні дані.

Такий поділ забезпечує розділення функцій, зручність у тестуванні окремих компонентів та можливість подальшого масштабування функціоналу (наприклад, додавання push-нотифікацій або автоматичних звітів).

На рівні даних архітектура включає два типи джерел:

1. Віддалена СКБД Firebird

Основним джерелом даних є сервер з базою даних Firebird. Він містить таблиці які безпосередньо необхідні для організації віддаленого перегляду:

- DEVICE — точки контролю,
- MEASURE — заміри (час, Fe, W, N),

До сервера встановлюється TCP-з'єднання, після чого виконується запит типу:

Для з’єднання використовується офіційна бібліотека FirebirdClient. Програма також отримує поточну дату/час із запитом до системної таблиці RDB$DATABASE, що дозволяє синхронізувати відображення часових величин, що відображаються на графіках та таблицях в усій системі за часом серверу точки контрою.

2. Локальна база SQLite, що застосовується для зберігання конфігурацій точок контролю. Це легка, вбудована реляційна СКБД, яка не потребує окремого серверу. База даних містить лише одну таблицю ItemsDevice, яка має структуру, наведену у таблиці 2.1 [3].

SQLite дає змогу зберігати локальні параметри навіть при відсутності підключення до мережі. Цей підхід підвищує надійність системи та дозволяє швидко змінювати конфігурацію без ручного редагування коду.

Таблиця 2.1 - Структура таблиці ItemsDevice локальної бази даних

Назва поля	Тип	Призначення
ID_DEVICE	INT, первинний ключ, автоінкремент	Ідентифікатор канала в таблиці
IP_ADRESS	TEXT(15)	IP-адресса сервера БД точки контролю
PATH	TEXT(255)	шлях до файлу БД на сервері
ID_DEVICE_FROM_DB	INT	Унікальний ідентифікатор в таблиці каналів серверу
DEV_TITLE	TEXT(255)	Назва точни контролю для ідентифікації у клієнтському застосунку
NUM_FROM_LIST	INT	Номер, визначаючий порядок відображення каналів

Реалізація збереження локальних конфігурацій за допомогою SQLite — обгрунтовується її перевагами, зокрема [4]:

- висока швидкість доступу до даних;
- не потребує серверної частини (файлова структура);

- легке резервне копіювання;
- простота реалізації через ADO.NET [5].

Серед засобів програмної реалізації обрано мову C# як основну, з врахуванням кількох важливих факторів [5]:

- за допомогою .NET Core та .NET 5/6/7 розробники можуть створювати застосунки, що працюють не лише під Windows, але й на Linux та macOS. Це забезпечує гнучкість розгортання систем на різноманітних платформах.
- C# має потужну систему типів, вбудовані механізми обробки винятків та захисту пам’яті. Це зменшує кількість помилок під час виконання програми й дозволяє створювати стабільні додатки, що критично важливо для виробничих систем.
- платформа .NET забезпечує ефективну реалізацію багатопоточних обчислень, що дозволяє оптимізувати оновлення даних у режимі реального часу.
- для C# доступна велика кількість відкритих та комерційних бібліотек для роботи з графікою, базами даних, мережами, звітами тощо.
- C# активно використовується у створенні вебсервісів, API, IoT-рішень та Azure-додатків, що відкриває перспективи розширення функціоналу майбутньої системи
- зручність розробк (середовище Visual Studio забезпечує інструменти для швидкого створення інтерфейсів, налагодження, тестування, профілювання продуктивності).

Таким чином, вибір C# як основної мови розробки обумовлений його стабільністю, підтримкою сучасних стандартів, гнучкістю і придатністю для довготривалого супроводу корпоративних застосунків.

Запропонована архітектура дозволяє досягти таких результатів:

- Гнучкість: локальна база SQLite дозволяє зберігати налаштування незалежно від серверної частини.

- Швидкодія: підключення напряму до Firebird без посередників забезпечує низьку затримку.
- Масштабованість: можливість підключення до кількох серверів одночасно, завдяки конфігураціям у SQLite.
- Надійність: логіка помилок відокремлена, інтерфейс захищено від зависань під час обробки запитів.
- Зрозумілість інтерфейсу: простий вигляд, орієнтований на персонал без глибоких технічних знань.

Розроблена архітектура програмного забезпечення NaksPC Look демонструє ефективне поєднання сучасних технологій і промислової надійності. Вона повністю відповідає специфічним вимогам гірничовидобувного виробництва, де ключовими є стабільність, точність, простота обслуговування та можливість адаптації до змін конфігурації. Подальший розвиток може передбачати реалізацію веб-клієнта або модулів аналітики, не змінюючи при цьому фундаментальної логіки взаємодії між компонентами системи.

2.2 Розробка алгоритмічного забезпечення клієнтського застосунку

У межах реалізації клієнтської частини програмного забезпечення NaksPC Look було спроєктовано та розроблено низку алгоритмів, що забезпечують взаємодію користувача із системою, зчитування та оновлення даних із локальної бази SQLite та серверної Firebird, а також побудову вибірки історій вимірювань. Алгоритмічне забезпечення застосунку реалізоване у вигляді структурованих процедур, які відповідають за основні життєві цикли роботи програми. Для підвищення наочності функціонування кожної з процедур було побудовано узагальнені блок-схеми, а також розроблено низку SQL-запитів, що забезпечують функціональність системи, однозначність даних та ефективне навантаження СУБД [6].

Загалом логіку функціонування клієнтського застосунку можна умовно

поділити на три ключові алгоритмічні процедури, які відображають реакцію програми на основні події — запуск, надходження нових даних та ініціативні дії користувача.

Алгоритм початкового запуску застосунку, що починається після запуску виконуваного файлу відбувається ініціалізація роботи з локальною конфігураційною базою даних (SQLite), в якій зберігається перелік доступних точок контролю (каналів) та параметри підключення до відповідних серверів. На основі отриманих даних формується структура активних підключень і ініціалізуються компоненти інтерфейсу користувача. Далі відбувається зчитування поточних значень параметрів із віддалених серверів, побудова відповідних графічних елементів (таблиць, індикаторів, графіків), а також налаштування циклічного таймера, що відповідає за періодичне оновлення даних. Узагальнена блок-схема алгоритму наведена на рисунку 2.3.

Ця процедура є критично важливою для коректного запуску та подальшого функціонування всієї програми, оскільки саме на цьому етапі визначається конфігурація системи, активуються канали зв'язку та встановлюються початкові параметри.

Алгоритм періодичного оновлення даних - основна процедура, яка активується автоматично після спрацювання таймера, період якого задається у конфігураційних параметрах. Після завершення циклу очікування «таймауту», програма виконує опитування відповідних точок контролю на сервері Firebird, отримуючи лише ті записи, які з'явилися з моменту останнього зчитування. Завдяки цьому реалізується механізм дельта-оновлення, що дозволяє суттєво зменшити навантаження на канал зв’язку та забезпечити високу швидкодію інтерфейсу.

Дані, які завантажуються з віддаленої БД, одразу ж оновлюють графіки, цифрові показники та інші візуальні компоненти інтерфейсу користувача. У разі втрати зв’язку або недоступності одного з серверів, програма реєструє помилку у внутрішньому журналі та продовжує роботу з іншими точками. Такий підхід забезпечує безвыдказныcть системи та дозволяє уникнути повної

зупинки в разі локальних збоїв. Блок-схема алгоритму періодичного оновлення даних представлена на рисунку 2.4.

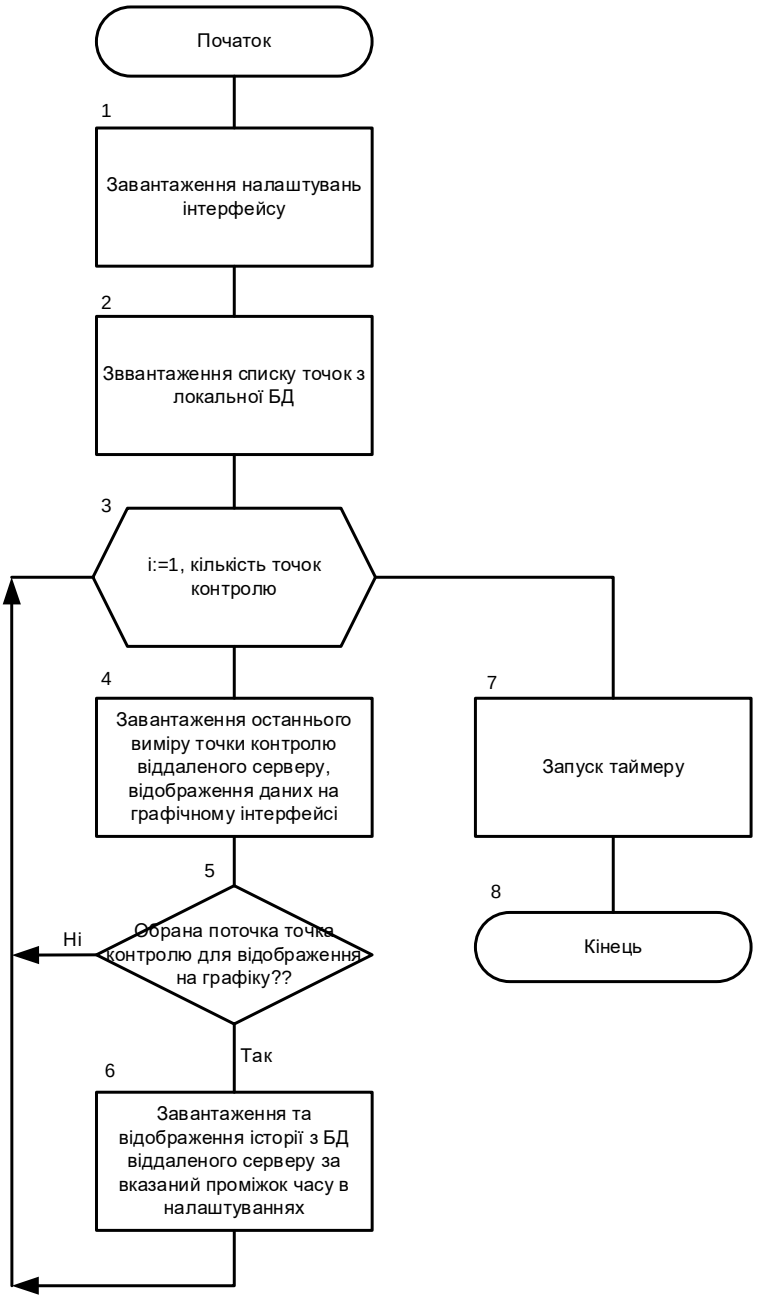

Рисунок 2.3 – Блок-схема узагальненого алгоритму запуску застосунку.

4. Алгоритм ручного отримання історичних даних. Третя важлива процедура реалізується у відповідь на дію користувача — натискання кнопки «Оновити» в інтерфейсі перегляду історії. У цьому випадку система формує SQL-запит до таблиці вимірювань MEASURE на сервері Firebird, де вказується заданий користувачем часовий діапазон. З

отриманого результату формується набір записів, які відображаються у вигляді таблиці або графіка, дозволяючи користувачу аналізувати історичні зміни параметрів потоку руди.

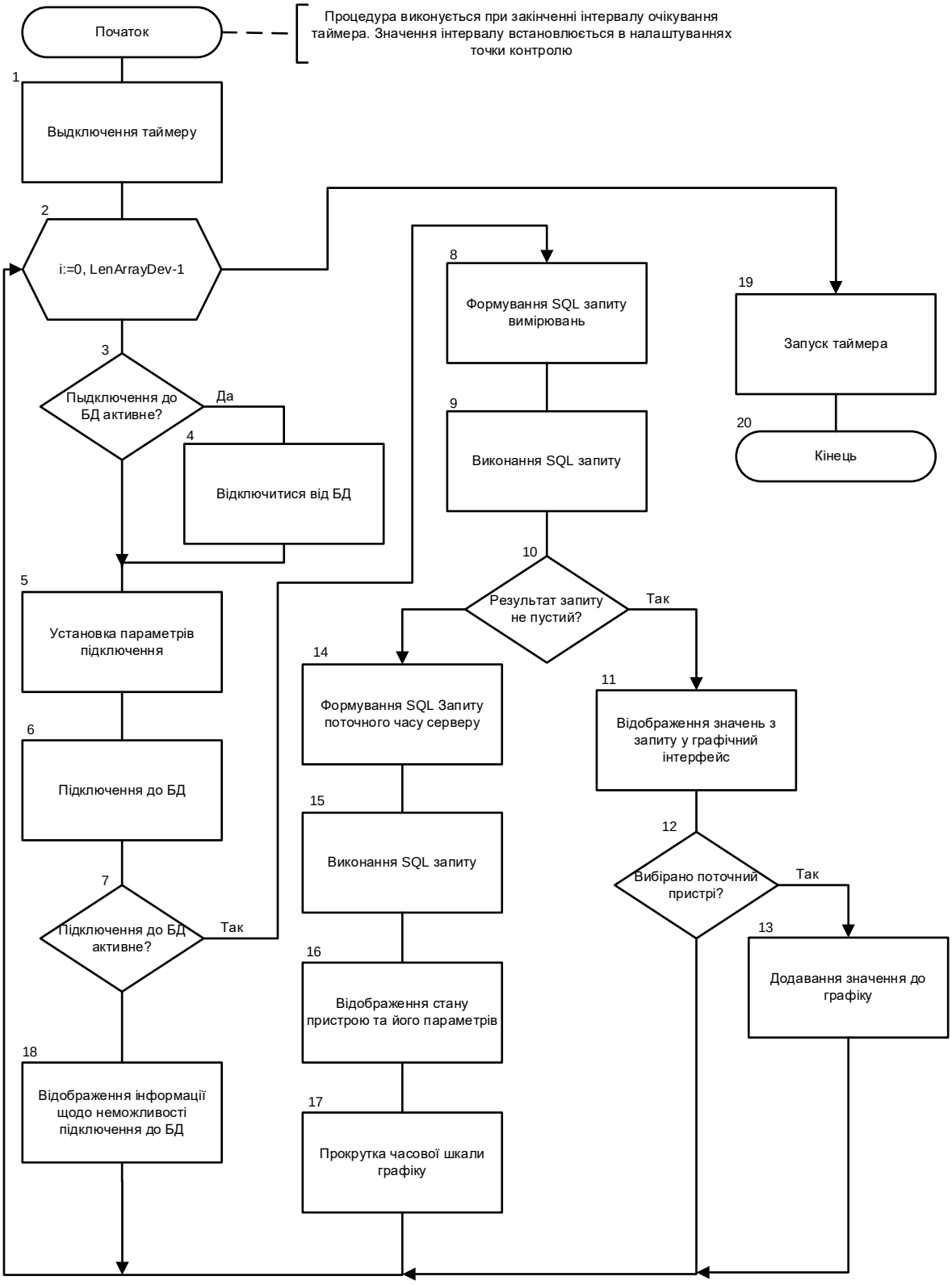

Рисунок 2.4 – Блок-схема алгоритму періодичного опитування серверу.

Алгоритм перегляду історії вимірювань - третя важлива процедура, яка ініціюється в результаті дій користувача — натискання кнопки «Оновити» в інтерфейсі перегляду історії. У цьому випадку система формує SQL-запит, до таблиці вимірювань MEASURE на сервері Firebird, де вказуються параметри, введені користувачем в відповідні поля. З отриманого запиту формується набір записів, які відображаються у вигляді таблиці або графіка, дозволяючи користувачу аналізувати історичні зміни параметрів потоку руди.

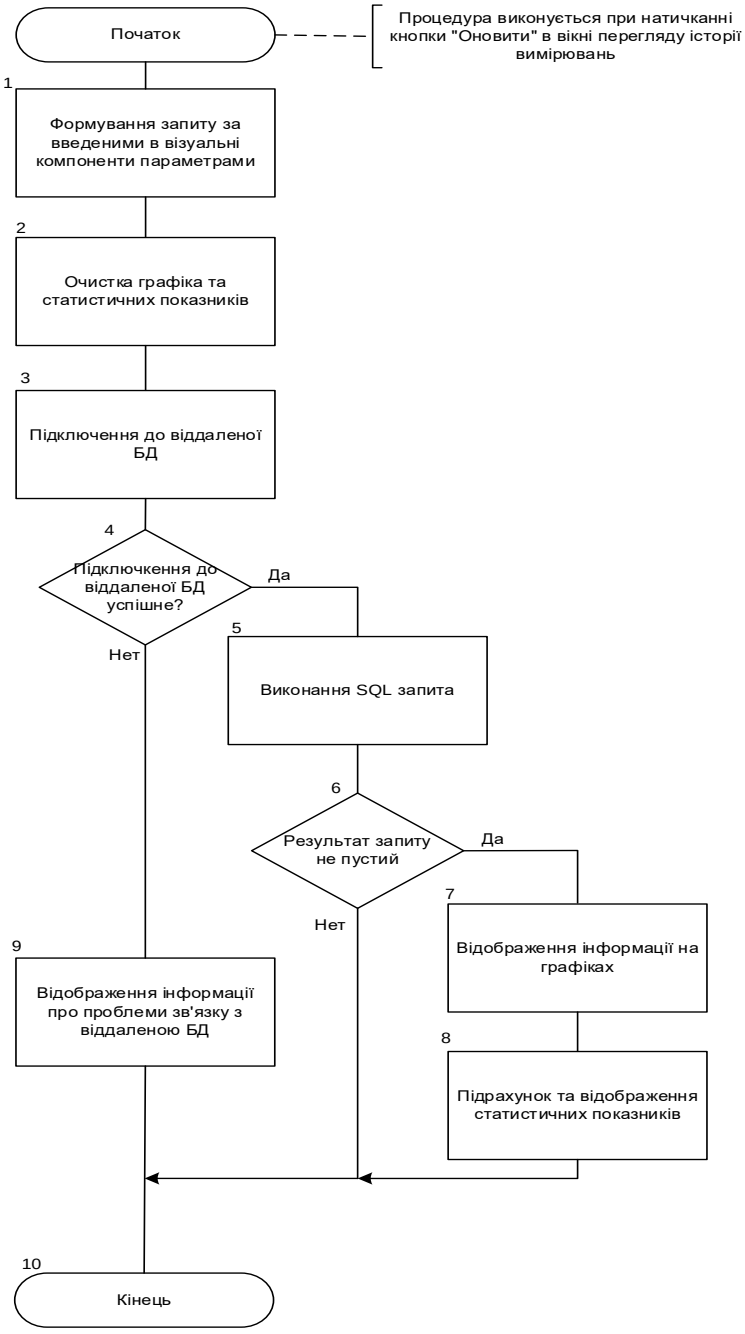

Рисунок 2.5 – Блок-схема алгоритму перегляду історії вимірювань.

Процедура підтримує фільтрацію за точками контролю, а також виконує перевірку обмежень на обсяг даних, щоб запобігти перевантаженню системи при виборі занадто широкого інтервалу. У разі виникнення помилок доступу або відсутності даних у вибраному проміжку часу, система формує відповідне повідомлення для користувача. Блок-схема алгоритму перегляду історії вимірювань наведена на рисунку 2.5.

Таким чином, алгоритмічне забезпечення клієнтського застосунку системи побудоване з урахуванням принципів надійності, розширюваності та відмовостійкості. Поділ функціоналу на окремі логічні процедури дозволяє підтримувати модульність коду, полегшує супровід та забезпечує високу читабельність і масштабованість у майбутньому розширення системи. Кожен з описаних алгоритмів взаємодіє з базою даних Firebird та локальною конфігураційною базою SQLite, що створює гнучку та адаптивну архітектуру, придатну для застосування в промислових умовах.

2.3 Розробка засобів взаємодії з віддаленими базами даних

У процесі реалізації клієнтської частини системи НАКС-ПК одним із ключових етапів стало створення ефективного механізму взаємодії з базою даних Firebird. Зважаючи на специфіку прикладної області — безперервний контроль якості рудної сировини — до запитів висувалися підвищені вимоги щодо продуктивності, стабільності та точності результатів [7].

Ефективна взаємодія клієнтського застосунку з базою даних є одним з ключових компонентыв у реалізації інформаційної системи НАКС-ПК. У контексті побудови клієнтської частини програмного забезпечення критично важливим є не лише забезпечення стабільного каналу зв’язку із серверною СКБД Firebird, а й оптимізація SQL-запитів, які використовуються для зчитування, фільтрації, агрегації та подання даних [8].

Для реалізації механізму підключення було використано офіційний .NET-драйвер FirebirdSql.Data.FirebirdClient, який дозволяє встановлювати

підключення до віддаленої бази даних у середовищі C# з використанням стандартного набору ADO.NET.

Створення з’єднання реалізовано через побудову рядка підключення із зазначенням усіх необхідних параметрів (рисунок 2.6):

```
string connectionString = new FbConnectionStringBuilder
{
    DataSource = "192.168.1.100",
    Database = @"D:\DB\NaksDB.fdb",
    UserID = "SYSDBA",
    Password = "masterkey",
    Charset = "UTF8",
    Port = 3050,
    Dialect = 3
}.ToString();

using (FbConnection connection = new FbConnection(connectionString))
{
    connection.Open();
    // подальша робота з базою
}
```

Рисунок 2.6 – Контекст процедури підключення до віддаленої БД.

Реалізація з’єднання інкапсульована в окремому класі FirebirdService, що дозволяє централізовано керувати підключенням, логуванням помилок та повторними спробами у разі втрати доступу. Додатково реалізовано механізм автоматичної перевірки доступності БД перед виконанням критичних запитів.

Для забезпечення однозначності та коректності зіставлення вимірювань у часовому просторі клієнтський застосунок синхронізовується з сервером бази даних. Один із перших запитів, який виконується після встановлення з’єднання, — це отримання поточного часу від серверної машини. Це дозволяє не покладатися на локальний системний годинник, що особливо важливо у розподілених індустріальних середовищах. Синтаксис цього запиту наведено на рисунку 2.7.

Результат цього запиту використовується у логіці оновлення даних, формуванні діапазонів для історичних запитів, а також на графіках журналі користувацької активності (якщо реалізований).

```
1  SELECT CURRENT_TIMESTAMP FROM RDB$DATABASE;
```

Рисунок 2.7 – SQL-запит часу серверу.

Оновлення графіків та цифрових показників у режимі реального часу базується на регулярному зверненні застосунку-клієнта до бази даних із метою отримання лише тих записів, які з’явилися з моменту останнього опитування. Це дозволяє зменшити обсяг переданих даних та покращити загальну продуктивність системи. Синтаксис такого запиту наведено на рисунку 2.8.

```
1  SELECT *
2  FROM MEASURE
3  WHERE ID_DEVICE = @device_id
4    AND DATE_TIME_MEASURE > @last_read_time
5  ORDER BY DATE_TIME_MEASURE;
```

Рисунок 2.8 – Запит оновлених записів в БД.

В запиті @device_id — це ідентифікатор точки контролю, а @last_read_time — останній зафіксований момент оновлення. Результати запиту додаються у візуальний потік графіка або таблиці в інтерфейсі користувача.

Запит виконується асинхронно, за таймером, з інтервалом 3–10 секунд (за налаштуванням), щоб забезпечити безперервність потоку даних.

Користувач має змогу переглядати архів вимірювань за будь-який проміжок часу в межах доступної історії. Для цього реалізується запит (рисунок 2.9) із параметрами @start_time і @end_time, які встановлюються вручну користувачем системи через графічний інтерфейс.

```
SELECT DATE_TIME_MEASURE, FE, W, N
FROM MEASURE
WHERE ID_DEVICE = @device_id
  AND DATE_TIME_MEASURE BETWEEN @start_time AND @end_time
ORDER BY DATE_TIME_MEASURE;
```

Рисунок 2.9 – Запит історії вимірювань за певний період.

Цей запит виконується у відповідь на натискання кнопки «Оновити» у формі перегляду історії. У разі великого обсягу даних (наприклад, при виборі кількох днів) реалізована перевірка кількості рядків і попередження користувача про можливу затримку відображення.

Для візуалізації історичних даних використовується таблична форма та інтерактивний графік з можливістю масштабування. Запит виконується один раз при кожному зверненні користувача, без кешування результатів.

Усі запити реалізовано як параметризовані з метою запобігання SQL-ін’єкціям і забезпечення гнучкості. Їх виконання здійснюється через об’єкти FbCommand бібліотеки FirebirdSql.Data.FirebirdClient, яка забезпечує нативну підтримку протоколу Firebird у середовищі .NET. Параметри передаються явно з використанням типів, одна з таких передач наведена на рисунку 2.10:

```
command.Parameters.Add("@start_time", FbDbType.TimeStamp).Value = startTime;
```

Рисунок 2.10 – Синтаксис передачі параметрів у запит.

Додатково реалізовано логування SQL-запитів у тестовому режимі для спрощення діагностики та оптимізації продуктивності при великих обсягах даних.

Розробка SQL-запитів у системі НАКС-ПК є фундаментальною частиною програмного забезпечення, що забезпечує зв'язок між клієнтським інтерфейсом і серверною базою даних. Запити до бази даних реалізовані таким чином, щоб мінімізувати трафік, забезпечити швидкодію навіть при роботі з великими обсягами інформації та підтримувати стабільну роботу системи в режимі 24/7. Ретельно продумана архітектура обробки запитів дозволяє гнучко

масштабувати систему, забезпечуючи стабільну роботу як у тестовому середовищі, так і у реальних промислових умовах.

2.4 Програмна реалізація системи

Розробка клієнтського застосунку віддаленого моніторингу для системи неперервного контролю якості рудної маси «НАКС-ПК» була здійснена на основі попередньо сформованих вимог, описаних алгоритмів та розробленої структури бази даних. Основною метою реалізації стало створення зручного, надійного та вже звичного для користувачів системи інтерфейсу, який дозволяє у реальному часі відображати показники технологічного контролю, здійснювати історичний аналіз та виконувати базові дії з налаштування точок вимірювання.

Для реалізації клієнтської частини було обрано середовище Microsoft Visual Studio з використанням мови програмування C# та бібліотек WinForms. Це рішення обумовлене кількома факторами. По-перше, платформа .NET надає потужні засоби для розробки десктопних застосунків із доступом до реляційних баз даних. По-друге, C# як мова має багату стандартну бібліотеку, підтримку багатопотоковості, розвинену систему обробки подій, а також широке ком’юніті розробників, що гарантує доступність документації та рішень. Крім того, Visual Studio забезпечує зручну візуальну побудову форм, автоматичне керування залежностями та інструменти налагодження, що суттєво пришвидшило розробку [9].

Структура проекту у Visual Studio представлена у вигляді дерева файлів, у якому кожен компонент має чітко визначене місце. На рисунку 2.11 подано зображення дерева файлів проекту, що демонструє логічну організацію класів, форм, модулів і конфігурацій.

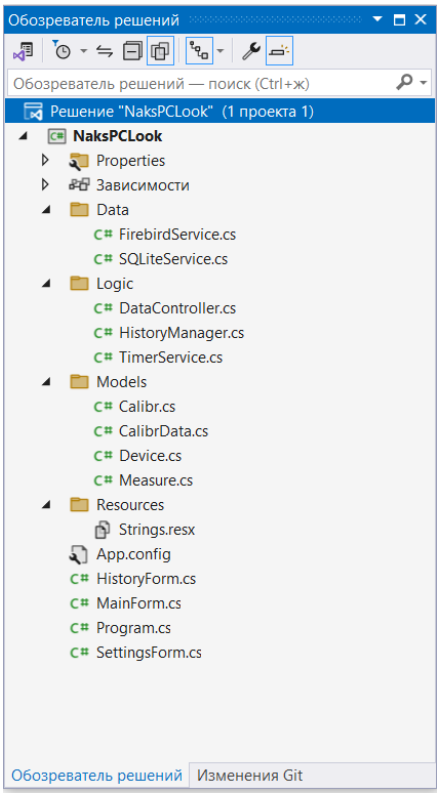

Рис. 2.10 – Дерево проекту клієнтського застосунку у середовищі розробки Visual Studio 2022.

Ключовим компонентом системи є механізм доступу до бази даних. У межах реалізації використано два канали зберігання інформації:

Firebird (основна віддалена база) — для зчитування поточних значень, історичних вимірювань, параметрів калібрування.

SQLite (локальна база) — для збереження конфігурацій, зокрема списку точок доступу, IP-адрес, портів, шаблонів підключення.

З’єднання з Firebird реалізується через бібліотеку FirebirdSql.Data.FirebirdClient, де у класі FirebirdService (рисунок 2.11) централізовано описано всі методи підключення, виконання SQL-запитів, обробки помилок і повернення результатів. Усі запити реалізовані як параметризовані, що захищає від SQL-ін’єкцій та забезпечує коректне кешування на стороні СКБД [10].

```
using System.Data.Common;

Ссылок: 1
public class FirebirdService
{
    private string connectionString;

    Ссылок: 0
    public FirebirdService(string dbPath)
    {
        var builder = new FbConnectionStringBuilder
        {
            DataSource = "192.168.0.10",
            Database = dbPath,
            UserID = "SYSDBA",
            Password = "masterkey",
            Charset = "UTF8"
        };
        connectionString = builder.ToString();
    }

    Ссылок: 0
    public List<Measure> GetLatestMeasures(int deviceId, DateTime afterTime)
    {
        // Повертає останні вимірювання
    }
}
```

Рисунок 2.11 – Класс для взаємодії з віддаленою БД.

Кожна з основних функціональних процедур, описаних в алгоритмічному забезпеченні, реалізована як окрема функція або клас. Так, наприклад:

- запуск застосунку ініціалізує об’єкти DataController та TimerService, які здійснюють зчитування з SQLite і старт опитування серверів;
- обробка подій від таймера виконує SQL-запити до Firebird з фільтрацією DATE_TIME_MEASURE > lastUpdateTime, обробляє нові записи та відображає їх у формі MainForm;
- перегляд історії реалізований у HistoryForm, де вибір користувачем діапазону дат генерує SQL-запит до таблиці MEASURE з обмеженням BETWEEN start AND end.

Результати запитів інкапсулюються у відповідних об’єктах моделі (Measure, Device) і через прив’язку даних виводяться у таблиці та графіки.

Кожна форма інтерфейсу реалізована на базі WinForms-компонентів: DataGridView для табличних даних, Chart для побудови графіків, Label та TextBox для цифрових показників. Реалізовано кольорове маркування за

параметрами, а також сповіщення у разі втрати з’єднання з сервером чи інших станів роботи системи.

Усі події від елементів керування зв’язані з відповідними методами у класах логіки, що забезпечує повну декомпозицію UI та бізнес-логіки.

Враховуючи що застосунок планується використовувати у промислових умовах особливої уваги приділено стійкості програмного забезпечення до збоїв. Застосунок реалізований з урахуванням високих вимог до безперервної роботи, тому всі критичні операції, пов'язані з доступом до бази даних, обробкою файлів, мережевими підключеннями та відображенням інформації, обгорнуті в захищені блоки try-catch [11].

Окрім базової обробки винятків, реалізована централізована система обробки винятків на рівні застосунку, яка відслідковує неперехоплені помилки (Application.ThreadException, AppDomain.CurrentDomain. UnhandledException) і надає користувачу інформативне повідомлення з можливістю збереження лог-файлу для подальшої діагностики фахівцями.

Це особливо важливо в системах, де програма має працювати у режимі 24/7, без необхідності постійного втручання оператора.

З метою відстеження внутрішнього стану програми, ключових подій та помилок, реалізовано модуль логування. Усі записи зберігаються у файл log.txt, що постійно доповнюється у режимі append. До журналу потрапляє:

- дата та час події;
- тип події (інформація, попередження, помилка);
- текст події або повідомлення про виключення;
- SQL-запити (у режимі діагностики).

Приклад запису в лог наведено на рисунку 2.10.:

```
[2025-05-05 14:32:16] [INFO] Підключення до Firebird встановлено.
[2025-05-05 14:33:01] [ERROR] SQL-помилка: таблиця MEASURE не існує.
```

Рисунок 2.10 – Приклад логування подій лог-файлу.

Це дозволяє оперативно проводити аудит поведінки системи та пришвидшує процес діагностики як під час розробки, так і в експлуатації.

2.5 Тестування клієнтського застосунку

Ефективність та стабільність роботи інформаційно-аналітичної системи безпосередньо залежать від якості реалізації та тестування клієнтської частини. Розроблений у рамках кваліфікаційної роботи клієнтський застосунок LookNaks.exe був ретельно перевірений за різними сценаріями взаємодії користувача з програмою. Основною метою тестування стало забезпечення функціональної коректності, стійкості до відмов, відповідності вимогам до візуального представлення даних та зручності користування в умовах виробничої експлуатації [12].

Графічний інтерфейс користувача тестувався з позиції логіки навігації, відповідності інтерфейсу очікуванням користувача та відображення критично важливої інформації у доступній формі. Робочі вікна «Головне» та «Історія» були реалізовані у стилістиці базового застосунк NaksPC для забезпечення візуальної послідовності, що значно спрощує адаптацію користувачів. Головне вікно відображає список контрольних точок, надаючи доступ до налаштувань, індикаторів працездатності пристроїв і основної статистики виглячд вікна наведено на рисунку 2.11.

Періодичне оновлення даних у головному вікні реалізовано за допомогою таймерів з інтервалом, який налаштовується параметром у вікні Налаштувань. Візуальні індикатори змінюють стан відповідно до наявності з’єднання та актуальності даних. Це дозволяє оператору в режимі реального часу контролювати загальну ситуацію без потреби звертатися до історії вимірювань та графіків.

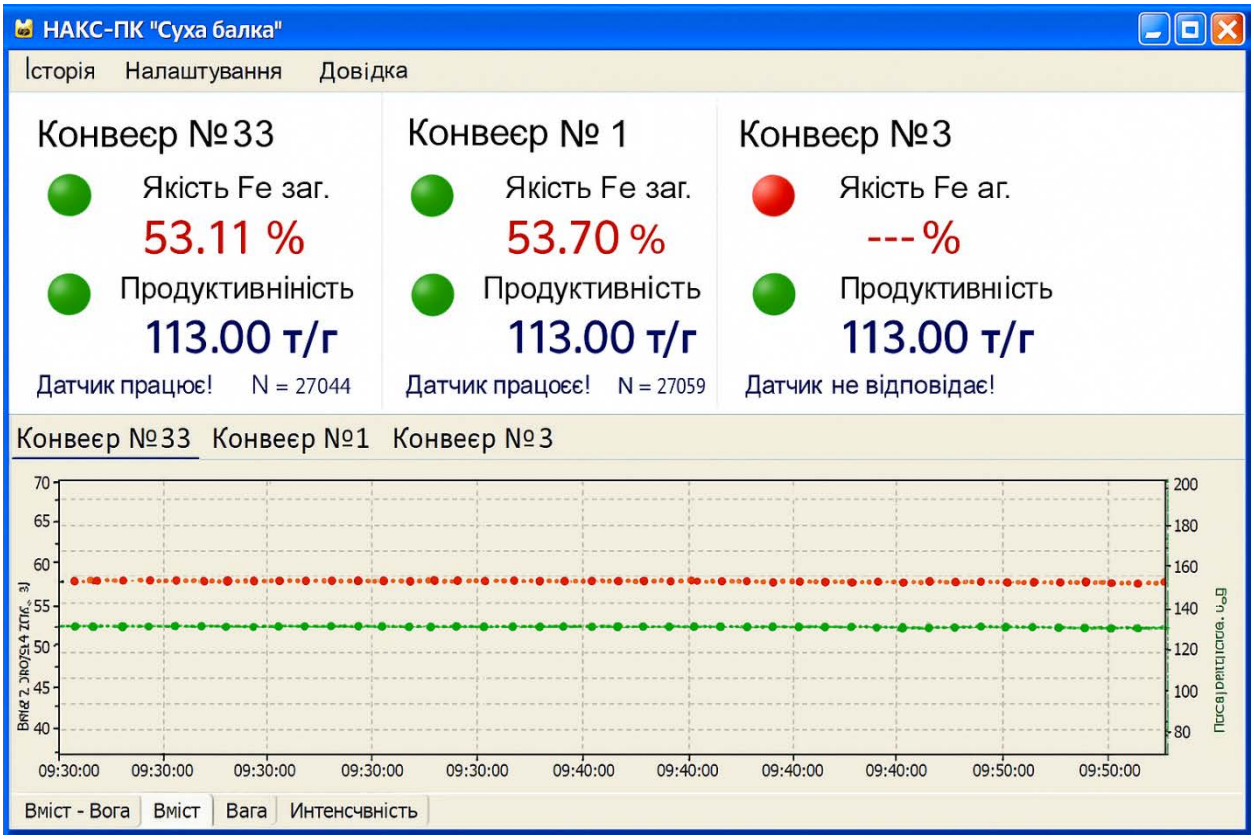

Рисунок 2.11 – Головне вікно графічного інтерфейсу клієнського застосунку.

Вікно перегляду історії дозволяє вибрати конкретну точку контролю, задати період часу та здійснити візуальний і табличний аналіз накопичених даних. Під час тестування перевірялась коректність SQL-запитів до віддаленої БД Firebird, а також обробка граничних ситуацій: порожні вибірки, накладення діапазонів часу, запити з великим обсягом даних (до 50 тис. записів).

Виявлено, що у разі перевищення обсягу результатів система не зазнає зависань, оскільки передбачено посторінкове завантаження. Також реалізована функція автоматичного оновлення історії з інтервалом у хвилину — що дозволяє оператору бачити останні зміни без додаткових дій.

Велика увага приділялася перевірці розділу системи «Налаштування» (рисунки 2.13 – 2.15), яке є критично важливим для коректної роботи системи. У ході тестування виконувались наступні дії:

- додавання нових точок контролю;
- редагування параметрів існуючих точок;
- видалення точок з бази локального збереження;

- перевірка діапазонів графічного відображення;
- тестування реакції на некоректні значення (від'ємні числа, занадто великі діапазони).

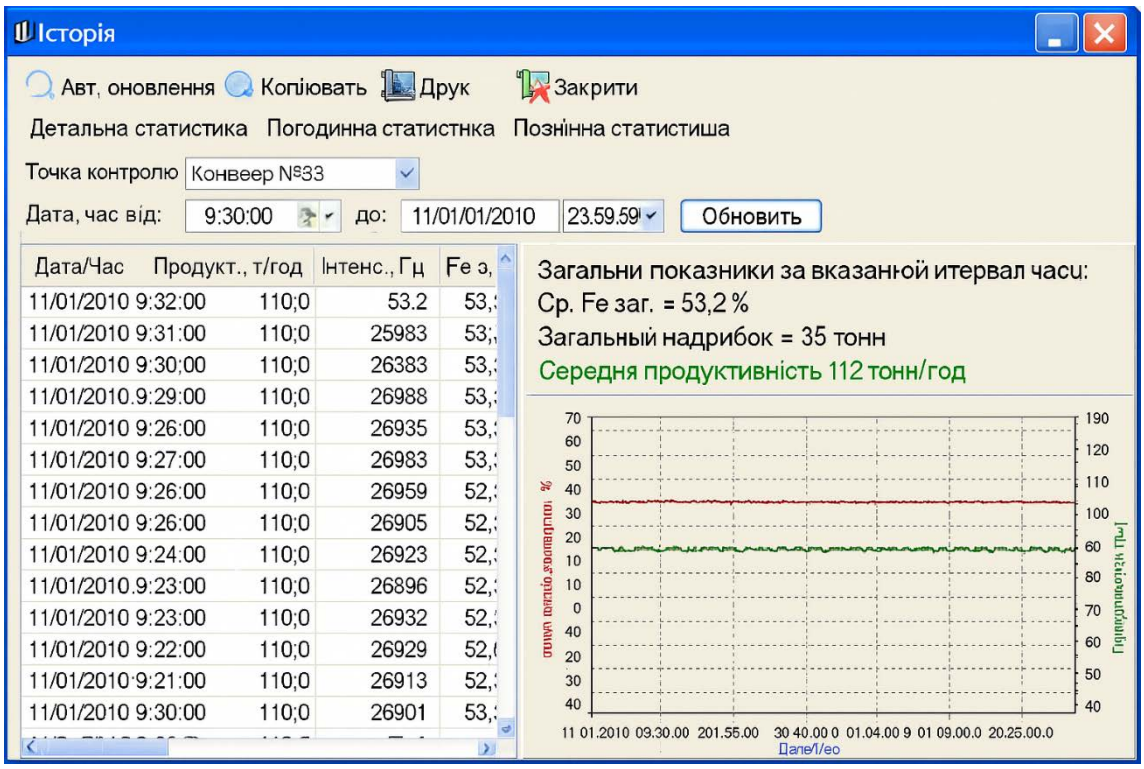

Рисунок 2.12 – Інтерфейс вікна перегляду історії вимірів.

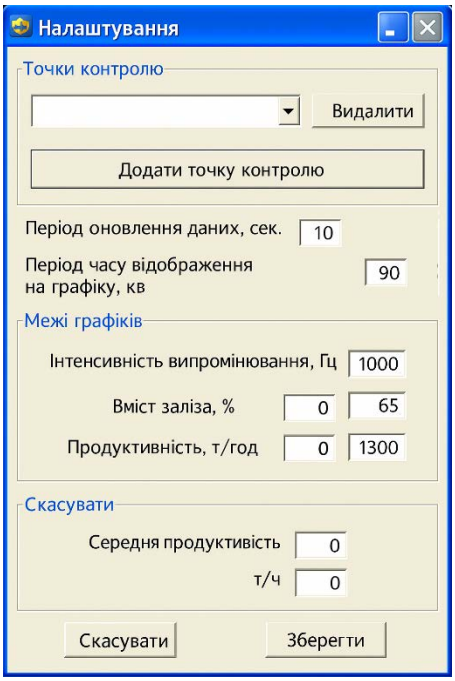

Рисунок 2.13 – Інтерфейс вікна налаштування відображення каналу.

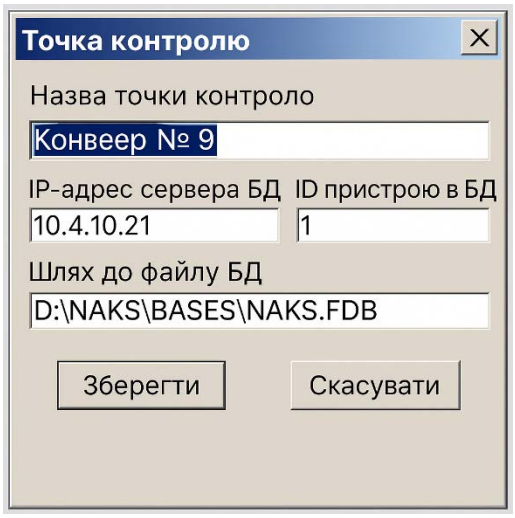

Рисунок 2.14 – Інтерфейс вікна налаштування параметрів каналу.

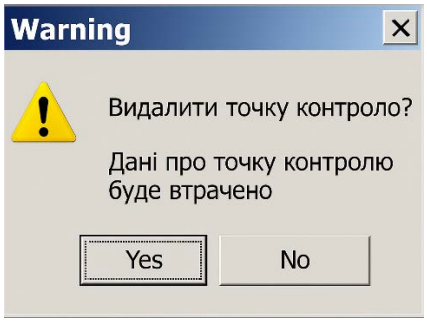

Рисунок 2.15 – Інтерфейс діалогового вікна попередження користувача

Окрема увага приділялася взаємодії із локальною БД SQLite, куди записуються налаштування. Було підтверджено, що усі дії на формі налаштувань адекватно синхронізуються з базою: при натисканні «Зберегти» інформація зберігається, а при натисканні «Скасувати» – відкидається без помилок.

При розробці регулярно проводилась діагностика застосунку в середовищі Visual Studio 2022, використовуючи вбудовані засоби [13]:

- точки зупину (breakpoints) для відстеження логіки виконання;
- перегляду вмісту об’єктів у режимі відлагодження;
- перегляду SQL-запитів у момент виконання;
- обробки виключень (exception watch);
- відстеження втрат з’єднання через модуль логування.

Усі критичні події логуються у log.txt, що зберігається у робочій

директорії програми. Тестувались також помилки з’єднання, втрати доступу до бази, відсутність параметрів конфігурації, комплексні результати наведені у таблиці 2.2.

Таблиця 2.2. – Результати тестування клієнтського застосунку

Тестовий сценарій	Очікуваний результат	Фактичний результат	Статус
Підключення до Firebird при запуску	З'єднання встановлено, дані отримані	Виконано успішно	Успішно
Оновлення даних через таймер	Нові дані з'являються у формі	Виконано успішно	Успішно
Запит історичних вимірювань	Відображення графіків та таблиць без затримки	Виконано успішно	Успішно
Редагування конфігурації точки контролю	Дані збережено у SQLite, форма оновлена	Виконано успішно	Успішно
Відображення даних при втраті зв'язку	Виведено повідомлення, програма працює далі	Виконано з попередженням	Успішно
Робота з великими обсягами (50 тис. записів)	Дані оброблено, не виникло зависань	Виконано успішно	Успішно
Виклик SQL-помилки навмисно	Помилка логована, програма не аварійна	Виконано з повідомленням у log.txt	Успішно
Завантаження даних без результатів	Повідомлення 'даних не знайдено'	Виконано успішно	Успішно
Зміна мови інтерфейсу	Елементи змінюють мову без помилок	Виконано успішно	Успішно
Вихід із програми під час опитування	З'єднання коректно закрите, програма завершилась	Виконано успішно	Успішно

В результаті комплексного тестування встановлено, що застосунок

функціонує стабільно, надійно, адекватно реагує на втрату з’єднання та не допускає аварійного завершення роботи. Всі винятки обробляються коректно, інтерфейс залишає користувача поінформованим, але не перевантаженим.

Застосунок демонструє високу продуктивність при роботі з великими масивами даних та забезпечує цілісність і актуальність інформації. Інтерфейс повністю локалізований, що забезпечує легкість адаптації персоналу.

Висновки до розділу

У цьому розділі здійснено повноцінне проектування, реалізацію та тестування клієнтського програмного забезпечення. Особлива увага приділялася глибокому аналізу архітектури майбутнього продукту, обґрунтованому вибору середовища розробки та технологій, а також забезпеченню стабільності та масштабованості створеної системи.

Першим кроком у розробці стала побудова архітектурної моделі клієнтського застосунку. Визначення модульної структури проєкту дозволило чітко розмежувати відповідальність між компонентами, що значно полегшило процес розробки, а також створило передумови для подальшого розширення функціональності. Ключовими складовими архітектури стали підсистеми доступу до баз даних, логіки взаємодії, інтерфейсів користувача та моделі предметної області. Особливої уваги заслуговує реалізація підтримки двох джерел даних — віддаленої СКБД Firebird, щ, та локальної бази SQLite.

Обґрунтовано обраним середовищем розробки стала платформа Visual Studio, яка в поєднанні з мовою програмування C# надала усі необхідні інструменти для швидкої та надійної реалізації графічного застосунку. Вибір Windows Forms як основного фреймворку для побудови інтерфейсу зумовлений вимогами до простоти розгортання, підтримки стандартного зовнішнього вигляду систем промислового призначення та широкими можливостями кастомізації. Важливим аргументом також стало те, що C# надає багаті засоби роботи з базами даних, доступ до системних ресурсів,

обробки виключень та багатопотоковості.

Подальша розробка алгоритмічного забезпечення клієнтського застосунку дозволила сформувати чітку логіку взаємодії з користувачем та базами даних. У результаті реалізовано основні функціональні елементи кожен з яких описаний у вигляді блок-схем.

Одним із ключових завдань було створення ефективного та стабільного механізму обміну з базами даних. Усі SQL-запити, включаючи вибірку актуальних вимірювань, визначення поточного часу на сервері та отримання історії за вказаними параметрами, реалізовано з урахуванням безпеки та продуктивності.

У результаті програмної реалізації було створено повноцінне прикладне рішення з графічним інтерфейсом, що дозволяє оперативно відображати дані про вміст заліза в рудному потоці, здійснювати перегляд історії вимірювань, редагувати параметри підключення до різних пристроїв та серверів. Усі елементи інтерфейсу українською мовою, що відповідає сучасним вимогам до промислового ПЗ в умовах національного виробництва.

Завершальним етапом розробки стало всебічне тестування клієнтського застосунку. Проведено як функціональну, так і інтеграційну перевірку. Результати тестування підтвердили високу стабільність та відповідність функціоналу поставленим вимогам.

Таким чином, на основі аналізу результатів проектування, реалізації та тестування можна зробити висновок, що клієнтський застосунок системи «НАКС-ПК» є завершеним, стабільним та придатним для розгортання в умовах промислової експлуатації.

ВИСНОВКИ

У межах кваліфікаційної роботи було повністю досліджено, спроєктовано та реалізовано клієнтський застосунок системи «НАКС-ПК» — спеціалізованого інструменту для безперервного контролю якості мінеральної сировини на конвеєрі. Проведені етапи охоплювали як теоретичну, так і практичну частини, що дозволило забезпечити як глибоке розуміння предметної області, так і створення готового до використання програмного продукту з урахуванням вимог промислової експлуатації.

У першому розділі було зосереджено увагу на аналізі функціональності існуючої системи «НАКС-ПК» — як у контексті її апаратної складової, так і програмного забезпечення, яке забезпечує роботу в автоматизованому режимі. Значну увагу було приділено вивченню архітектурної побудови системи, способам збору і обробки інформації з вимірювальних каналів, аналізу структури бази даних, що акумулює результати гамма-аналізу та супутніх параметрів. Проведений критичний огляд програмних рішень, що вже існують для візуалізації даних із віддалених СКБД, дозволив зробити обґрунтований висновок про недостатність їх функціоналу для повноцінної інтеграції з системами на кшталт «НАКС-ПК», що стало мотивацією до створення власного застосунку.

Другий розділ був повністю присвячений технічним аспектам реалізації клієнтського ПЗ. На етапі проєктування архітектури були сформовані чіткі структурні компоненти застосунку, що відповідають принципам розподілу відповідальності та гнучкості. Зокрема, впроваджено розділення логіки між модулями взаємодії з Firebird і SQLite, підсистемами логіки оновлення даних, керування історією та інтерфейсом користувача. Обґрунтовано вибір мови C# та середовища Visual Studio як основи реалізації — з огляду на їх стабільність, багатофункціональність і відповідність вимогам до Windows-орієнтованого програмного забезпечення.

Алгоритмічне забезпечення клієнтського застосунку було реалізоване відповідно до визначених сценаріїв роботи: завантаження конфігурації при запуску, періодичне опитування серверів з оновленням даних, а також запити до історичної інформації за параметрами користувача. Усі SQL-запити були спроєктовані з урахуванням продуктивності, безпеки та відповідності структурі бази. Значну роль у забезпеченні надійності системи відіграє коректна обробка винятків та логування подій, що дозволяє проводити діагностику у випадку технічних збоїв.

Програмна реалізація завершилась створенням повноцінного Windows Forms-застосунку з українським інтерфейсом, адаптованого до використання на промислових підприємствах. Головна форма, форма перегляду історії та форма налаштувань були виконані у зручному стилі, що поєднує простоту, інформативність та стабільну інтеграцію з серверними ресурсами. Проведене функціональне тестування підтвердило відповідність реалізованого функціоналу поставленим технічним вимогам. Система показала стабільну роботу в умовах великого потоку даних, забезпечила відсутність зависань та аварій, а також правильну реакцію на граничні сценарії.

У підсумку можна зробити висновок, що цілі, поставлені у кваліфікаційній роботі, були досягнуті в повному обсязі. Розроблений клієнтський застосунок повністю відповідає вимогам до сучасного інженерного програмного забезпечення для виробництва, демонструє високий рівень функціональності, розширюваності та зручності користування. Архітектурна модель, алгоритмічні рішення та реалізація є основою для подальшого розвитку системи — зокрема, інтеграції в SCADA-комплекси, підтримки нових типів аналізаторів, а також перенесення частини функцій у хмарні або мобільні сервіси. Таким чином, робота має не лише навчальну, а й практичну цінність, будучи цілком готовою до використання у виробничому середовищі.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Радченко Ю. Системи збирання та обробки даних. — Дніпро: Наука і освіта, 2020. — 276 с.
2. Брауде Е., Гельбард М. Архітектура програмного забезпечення. — Львів: ЛНУ, 2020. — 370 с.
3. Firebird Project. Firebird Documentation [Електронний ресурс]. — Режим доступу: https://firebirdsql.org/
4. SQLite. Documentation [Електронний ресурс]. — Режим доступу: https://www.sqlite.org/docs.html
5. MSDN Library. Microsoft Docs: System.Data.SQLite [Електронний ресурс]. — Режим доступу: https://learn.microsoft.com/
6. Стефанишин Д.І. Алгоритми та структури даних : навч. посіб. / Д.І. Стефанишин. — Львів : Львівська політехніка, 2020. — 256 с.
7. Codd E. F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 1970.
8. Бен-Форт О. SQL. Основи програмування баз даних : пер. з англ. / О. Бен-Форт. — Київ : Діалектика, 2020. — 432 с.
9. Власенко С. Сучасні технології розробки програмного забезпечення. — Львів: ЛНУ, 2021. — 298 с.
10. ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE).
11. Троєлсен Е. C# 8.0 і платформа .NET Core 3.0 : навч. посіб. / Е. Троєлсен, Ф. Джепікс ; пер. з англ. — Київ : Діалектика, 2020. — 1040 с.
12. Методи і засоби тестування програмних систем: навч. посібник / П.П. Попов, В.О. Мартинюк. — Вінниця: ВНТУ, 2021. — 148 с.

13. Visual Studio Documentation [Електронний ресурс]. — Режим доступу: https://learn.microsoft.com/en-us/visualstudio/ — Назва з екрана. — Дата звернення: 02 черв. 2025.
14. Моркун Н. В., Завсегдашня І. В. Методичні вказівки до виконання кваліфікаційної роботи бакалавру для студентів спеціальності 122 “Комп’ютерні науки”. Кривий Ріг : Видавничий центр КНУ, 2021. 50 с.
15. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).
16. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).
17. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація).