Міністерство освіти та науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Системотехніки
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

ГЮІК 50 5400.0103 ПЗ
(позначення документа)

Розробка системи зберігання та аналізу якості звітності про виконання завдань виробничим персоналом (на прикладі гірничодобувного підприємства)
(тема)

Виконав:
Студент *2* курсу групи СПРм-19-1
Спеціальність *122 – Комп'ютерні науки*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Системне проектування*
(повна назва освітньої програми)
Дмитрієв О.В.
(прізвище, ініціали)
Керівник *проф. Мінухін С.В.*
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри системотехніки _____ *Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*

Кафедра *Системотехніки*

Рівень вищої освіти *другий(магістерський)*

Спеціальність *122 Комп'ютерні науки*

(код і повна назва)

Освітня програма *Системне проектування*

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Дмитрієву Олександру Володимировичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка системи зберігання та аналізу якості звітності про виконання завдань виробничим персоналом (на прикладі гірничодобувного підприємтва)* затверджена наказом по університету від "*02*" *листопада* 2020 р. № *1516Ст*
2. Термін подання студентом роботи до екзаменаційної комісії "16" *грудня* 2020 р.
3. Вихідні дані до роботи *Структура гірничодобувного підприємства, документообіг гірничодобувного підприємства,характер взаємозв'язків між структурними елементами підприємства, критерії якості звітності та правила перевірки звітів для кожного типу робіт. Вимоги до вхідних даних: усі дані збережено у відповідних таблицях бази даних, збережена їх реляційна цілісність. Вимоги до функціонування додатку: процес аналізу якості звітності відбувається у окремому потоці; результати аналізу можуть бути збережені до файлу Microsoft Excel.*

4. Перелік питань, що потрібно опрацювати в роботі: *4.1 Вступ. 4.2 Аналіз предметної області. Постановка задачі 4.2.1 Аналіз структури гірничодобувного підприємства. 4.2.2 Аналіз документообігу гірничодобувного підприємства. Визначення змісту звітів. 4.2.3 Аналіз вимог до завдань на гірничодобувному підприємстві. 4.2.4 Визначення взаємозв'язку між елементами структури гірничодобувного підприємства та вимогами до завдань.4.2.5 Аналіз існуючих методів вирішення проблеми. 4.2.6 Постановка задачі. 4.3 Розробка методів перевірки якості виконання завдань. 4.3.1 Визначення вимог до бази даних для зберігання звітів. 4.3.2 Критерії оцінки якості виконання завдань. 4.3.3 Розробка методів аналізу якості виконання завдань 4.3.4 Результати аналізу якості звітності про виконання завдань. 4.4 Розробка бази даних. 4.4.1 Розробка логічної моделі бази даних. 4.4.2 Розробка фізичної моделі даних. 4.4.3 Верифікація бази даних на основі формування запитів та аналіз результатів верифікації. 4.5 Розробка підсистем зберігання та оцінки якості звітів. Аналіз результатів. 4.5.1 Розробка підсистеми зберігання звітів про виконання завдань 4.5.2 Розробка підсистеми формування правил перевірки звітності про виконання завдань. 4.5.3 Розробка підсистеми*

аналізу якості звітності про виконання завдань. 4.5.4 Оцінка повноти запропонованого рішення

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *5.1 Узагальнена ієрархічна схема виробничих підрозділів гірничодобувного підприємства. 5.2 Узагальнена схема документообігу гірничодобувного підприємства. 5.3 Схема алгоритму перевірки якості звітів за критерієм наявності файлів. 5.4 Схема алгоритму перевірки якості звітів за критерієм своєчасності. 5.5 Логічна модель бази даних*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	*Отримання завдання атестаційної роботи*	*09.09.2020*	
2.	*Аналіз завдання, літератури та аналогів з теми атестаційної роботи*	*10.09.2020-30.09.2020*	
4.	*Розробка методів аналізу якості звітності*	*01.10.2020-20.10.2020*	
9.	*Оформлення пояснювальної записки та програмної документації*	*21.10.2020-04.12.2020*	
10.	*Оформлення графічної частини та презентаційних матеріалів комп'ютерного захисту*	*04.12.2020-07.12.2020*	
11.	*Представлення на рецензування*	*11.12.2020*	
12.	*Представлення атестаційної роботи ДЕК*	*16.12.2020*	

Дата видачі завдання "09" вересня 2020 р.

Студент Дмитрієв О.В.
(підпис)

Керівник роботи _____ проф.. Мінухін С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Атестаційна робота: 101 стор., 35 рис., 3 додатки, 32 джерел. Графічні матеріали: 5 плакатів.

БАЗА ДАНИХ, СИСТЕМА, АНАЛІЗ ЯКОСТІ, СТРУКТУРА ПІДПРИЄМСТВА, ПРАВИЛА ПЕРЕВІРКИ.

Мета роботи – розробка методу зберігання та аналізу якості звітності про виконання завдань виробничим персоналом з його реалізацією, що дозволить автоматизувати процес аналізу якості звітності та зробить зберігання звітності більш зручним, швидким та надійним.

Область застосування задачі – виробничі підприємства різних галузей, перш за все гірничодобувної.

У пояснювальній записці розглянуто структуру та взаємозв'язки між елементами гірничодобувного підприємства, проаналізовано документообіг та критерії якості звітності. Розроблено метод аналізу якості звітності та реалізовано його у програмному продукті з обґрунтуванням комплексу використаних технологій.

Програмний продукт реалізовано у середовищі Microsoft Visual Studio 2013, у якості системи керування базою даних використано Microsoft SQL Server 2017. Для створення програмного продукту використовувалася об’єктно-орієнтована мова програмування C#. Для функціонування програмного додатку необхідна 64-х бітна операційна система Windows 7 або більш нова зі встановленим .Net Framework 4.5.1.

ABSTRACT

Attestation work: 101 p. 35 pic. 32 sources, 3 applications. Graphic material attestation work contains 5 posters.

DATABASE, SYSTEM, QUALITY ANALYSIS, COMPANY STRUCTURE, TEST RULES.

The purpose of the work is to develop and implement the method that can automate the process of storage and quality assessment of reporting on assignments production staff (on the example of a mining enterprise).

The scope of the work is enterprises of different industries especially the mining industry.

The explanatory note describes structure and relationship between elements of a mining enterprise, analyzed document flow and quality criteria of reporting. Developed a method for analyzing the quality of reporting and implemented it in software.

The software is implemented in the Microsoft Visual Studio IDE. As server database managed system is selected Microsoft SQL Server 2017. C# object-oriented language was used to create the software. Software requires Windows 7 or higher operating system with .Net Framework 4.5.1 installed.

ЗМІСТ

ВСТУП

Проблема зберігання звітів про виконання завдань виробничим персоналом та аналізу їх якості є актуальною для підприємств будь якого розміру та типу. Для великих виробничих підприємств вона є більш вагомою, бо кількість звітів зростає разом із розміром підприємства та складністю виробничого процесу. Вирішуючи цю проблему потрібно відповісти на такі питання «Яким чином зберігати звіти?» та «Яким чином перевіряти якість (наявність, своєчасність, повноту) звітів?».

Відповідь на питання «Яким чином зберігати звіти?» дає змогу створити структуру даних, що максимально чітко відображає структуру цільового підприємства та зберігає усі істотні особливості взаємозв'язку між елементами структури підприємства.

Відповідь на питання «Яким чином перевіряти якість (наявність, своєчасність, повноту) звітів?» дасть змогу зрозуміти, які саме критерії відрізняють якісний звіт від неякісного та оцінювати звіти за цими критеріями.

Методи зберігання та оцінки якості звітності про виконання завдань виробничим персоналом було вирішено розглядати саме на прикладі гірничодобувного підприємства, бо на сьогоднішній день гірничодобувна промисловість швидко зростає, залучає багато інвестицій та перебуває у процесі пошуку нових, більш економічних, швидких та сучасних, методів вирішення проблем усіх рівнів [1-9]. З аналізу робот [1-9] однозначно видно, що гірничодобувна промисловість, зокрема вугільна в Україні, зараз перебуває у процесі розвитку та привертає дедалі більший інтерес інвесторів , що робить цю сферу промисловості дуже перспективною.

Як і будь-яка інша, гірничодобувна промисловість перебуває у постійному пошуку кращих (більш ефективних та економічних) рішень, у тому числі й у питаннях оснащення виробничого персоналу, зокрема впровадження кращого програмного забезпечення, яке реалізує більш досконалі методи збереження та обробки інформації. У роботі [10] розглядається вплив різних факторів на

ефективність роботи персоналу. З розгляду цієї роботи видно, що приблизно 10% ефективності праці виробничого персоналу залежить від якості обладнання, зокрема програмного забезпечення. Тож запровадження програмного забезпечення, що базується на покращених методах зберігання та обробки інформації, може суттєво підвищити ефективність роботи виробничого персоналу, а отже й підприємства в цілому.

У роботі [11] акцент зроблено на необхідності запровадження аналізу ефективності та правильності роботи виробничого персоналу з метою покращення процесу стратегічного планування та керування. Згідно з цією роботою, запровадження аналізу результатів роботи покращує такі показники, як ефективність та середню продуктивність праці виробничого персоналу, та, у деяких випадках, рівень охорони праці.

З наведеного вище слідкує, що пошук нових, більш досконалих методів зберігання та аналізу звітності про виконання завдань виробничим персоналом є актуальною, бо запровадження нових методів може підвищити продуктивність персоналу гірничодобувної промисловості. Розробку методу пропонується проводити саме на прикладі гірничодобувного підприємства, бо ця сфера промисловості України є швидко зростаючою та перспективною.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз структури гірничодобувного підприємства

Структура гірничодобувного підприємства буде розглядатися з метою розуміння принципу розподілення завдань між виробничим персоналом та знаходження взаємозв'язку між структурою підприємства та вимогами до виконання завдань.

Гірничодобувне підприємство у своєму складі зазвичай має шахти, кар'єри, дробильно-сортувальні, збагачувальні або агломераційні фабрики, що у свою чергу поділяються на ділянки, секції, відділи, майстерні та інші виробничі підрозділи. Крім цього, у склад підприємства можуть входити учбово-курсові комбінати, клуби, дитячі садки та інше [12].

Оскільки метою роботи гірничодобувного підприємства є видобуток цільової сировини (палива, руди, каміння тощо), є доцільним більш детально проаналізувати структуру тих елементів підприємства, що безпосередньо цим займаються, тобто його виробничих підрозділів [12].

До виробничих підрозділів відносяться цехи та підрозділи, що займаються проведенням гірничих виробок, видобуванням руди, її подрібненням, збагаченням та агломерацією [12].

Структура гірничодобувного підприємства є ієрархічною: є один кореневий елемент(рудник), що поділяється на менші дочірні елементи, а ті у свою чергу на ще менші і так далі. Найменшим елементом структури гірничодобувного підприємства є блок чи виробка, залежно від способу видобутку породи.

Існує три основних методів видобутку руди: відкритий, підземний, та комбінований відрито-підземний. Верхнім рівнем видобувного підрозділу гірничодобувного підприємства зазвичай є родовище або філіал, що є групою географічно близько розташованих родовищ. Кожне родовище поділяється на регіони. Регіони поділяються на горизонти. Горизонти поділяються на блоки та підблоки для відкритого способу видобутку або на виробки для закритого [12-13].

Окрім загальних елементів структури, що характерні для усіх способів видобутку, можуть буди присутні й спеціальні, що вказують на елементи, що присутні лише при конкретному способі видобутку. Наприклад, для відкритого способу видобутку характерним є наявність кар'єру, для змішаного – окремо виділення підземного та надземного рудника, а для закритого – ділянки та шахти [12-13].

Також структура може містити додаткові елементи, які відповідають за геологічні дослідження, розрахунок допоміжних показників, гаражі тощо. Наявність додаткових елементів та їх структура залежить від внутрішнього регламенту підприємства [12-13].

Виходячи з наведеного вище та приймаючи до уваги ієрархічну специфіку структури гірничодобувного підприємства та з метою виділення структури виробничих підрозділів, було розроблено ієрархічну, а саме деревоподібну схему структури підприємства (рисунок 1.1).

Кількість елементів кожного рівня залежить від масштабів видобутку та положення цільової руди у просторі. Так, якщо руда зосереджена в одному місці, то немає необхідності у створенні великої кількості шахтних стволів.

Подальший аналіз взаємозв'язків між елементами структури підприємства та вимогами до виконання завдань буде проводитися з урахуванням схеми, зображеної на рисунку 1.1.

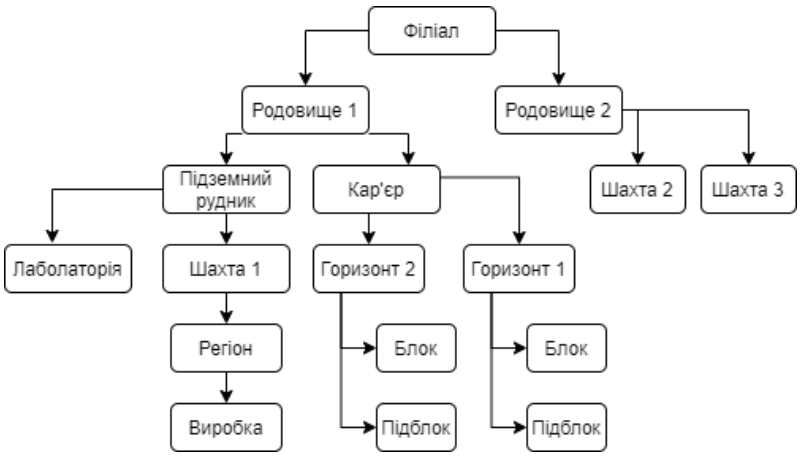


Рисунок 1.1 – Узагальнена ієрархічна схема виробничих підрозділів гірничодобувного підприємства

1.2 Аналіз документообігу гірничодобувного підприємства. Визначення змісту звітів

Для розуміння документообігу гірничодобувного підприємства необхідно розглянути процес видобутку руди з найменшого елементу структури рудника: блоку або виробки, – від планування до завершення.

Перш за все проводиться проектування гірничорозвідувальних робіт. Проектування – розробка проектної, конструкторської та іншої документації, призначеної для здійснення будівництва будь-якого об’єкту. Від якості проектування у великій мірі залежать темп технічний прогрес при виконанні видобувних робіт [13]. Результатом процесу проектування є створення переліку спеціальних документів – проекту.

Проект містить необхідні технічні й економічні розрахунки, схеми, графіки, пояснювальну записку, кошториси та інші матеріали [13].

Проектування зазвичай проводиться спеціалізованими проектними організаціями на основі завдання, що формується гірничодобувним підприємством. У завданні вказується найменування об’єкту, вид будівництва, місцезнаходження площадки будівництва, номенклатура та об’єми виробництва та інша необхідна інформація за вимогою проектної організації [13].

Після створення та затвердження проекту складається робочий проект, що включає: загальну частину, гірничотехнічну, гірничо-електромеханічну, будівничу частини, техніку безпеки, охорону довколишнього середовища, робочу документацію та інші документи за потреби [13].

Кожна з частин робочого проекту складається з певного переліку документів, що детально розглядаються у роботі [13]. Робочий проект створюється для визначення узагальненого бачення об’єму робіт, що мають бути зроблені, технічних та економічних показників підприємства, оцінки рентабельності проекту.

На основі робочого проекту створюються проекти виконання робіт (ПВР), що необхідні для найбільш ефективного виконання гірничодобувних робіт та

сприяють зниженню їх вартості, скороченню часу проведенні робіт та підвищенню безпеки праці. ПВР передбачають детальну розробку технології та порядку виконання робіт. ПВР створюються для конкретного родовища чи шахти/рудника. Як правило, ПВР оновлюється щомісячно або щоквартально для уточнення поточного стану виконання робіт та формування на його основі нового проекту виконання робіт [13].

На основі ПВР створюються типові проекти для забезпечення виробничих підрозділів необхідною проектною документацією на окремі елементи технічного призначення або на організаційні та технічні рішення, що часто повторюються на підприємстві. Типові проекти необхідні для налагодження виробництва нестандартних конструкцій чи елементів та прискорення процесу будівництва й видобутку [13].

Залежно від способу видобутку конкретних блоків формується перелік додаткової інформації. Наприклад, при видобутку блоку буровибуховим способом, створюється план буріння для закладання вибухівки, план її підриву та план вивезення породи. Ці плани складаються на тиждень та передаються підрядникам. Підрядниками, як правило, виступають сторонні підприємства, що займаються безпосередньо видобутком породи [13].

Більш детально про склад типових проектів наведено у роботі [13].

Безпосередньо перед початком видобувних робіт на основі типових проектів та ПВР створюються технологічні схеми та паспорти процесів видобувних робіт. У цих документах зазначені якісні та кількісні характеристики проведення робіт. Якісні характеристики вказують на організаційні та технічні сторони технології проведення робіт, а кількісні – характеризують виробничі процеси, структуру гірничих виробок або кар’єру, гірничо-геологічні умови проведення видобутку.

Паспорти процесів видобувних робіт, окрім наведеного вище, зазначають порядок проведення видобувних робіт, наприклад вказівки з закладання вибухівки, буріння свердловин, техніки безпеки тощо.

Усі зазначені вище документи створюються як персоналом гірничодобувного підприємства, так і сторонніми організаціями, та нормуються діючими нормативно-правовими документами й затверджуються у відповідних інстанціях. Зазвичай ці документи створюються для проміжних елементів структури: кар'єру, шахти, горизонту чи родовища, – для визначення порядку, темпу та планових результатів проведення видобувних робіт. Для елементів структури, на яких безпосередньо проводяться видобувні роботи (блоки, виробки, штольні тощо) будуть створюватися звіти за фактом виконання певних робіт чи завдань виробничим персоналом підприємства [14].

На основі паспортів процесів видобувних робіт проводяться безпосередньо видобувні роботи. Для прикладу буде розглянуто процес проходки штольні буровибуховим способом. Цей процес э досить складним та багатоетапним. Детальніше про нього розглянуто у роботі [13]. Для аналізу документообігу слід зазначити, що процес проходки штольні складається з п'яти етапів, що проводяться згідно з паспортом виконання видобувних робіт [13]:

– буріння шпурів;

– вибух шпурів;

– вивезення породи;

– закріплення;

– допоміжні роботи (розширення електромережі та водопроводу, геологічні та маркшейдерські спостереження та інше) за необхідністю.

За фактом проходження ділянки штольні складається звіт, у якому зазначаються усі необхідні дані: об'єми видобутої породи, кількість проведених вибухів, час витрачений на вентиляцію та вивезення породи, дати початку та кінця робіт на поточній ділянці штольні. Точний перелік звітних даних також задається у паспорті виконання робіт та може додатково корегуватися підприємством. Звіт зазвичай складається для кожного етапу робіт окремо, тобто після видобутку ділянки штольні буде складено, щонайменше, чотири звіти [14].

Окрім безпосереднього видобутку руди, проводяться допоміжні роботи, що включають у себе збір проб ґрунту та їх аналізу, топозйомки кар'єру, створення

проектного та реального зображення родовищ, облік відвалів(місця зберігання відпрацьованої породи), перерахунок планових та аналіз фактичних показників тощо [13].

На виконання певного типу робіт видається відповідне завдання виробничому персоналу. У завданні зазначається термін завершення робіт та посилання на відповідний паспорт процесу видобутку. Завдяки посиланню на відповідний паспорт, виробничий персонал має чітке уявлення про характер завдання, необхідні підготовчі роботи, особливості виконання робот на конкретній ділянці та яку звітну інформацію необхідно надати при завершенні виконання завдання [13].

Також слід зазначити, що більшість робіт є ітеративними, тобто повторюються через певний інтервал часу до повного завершення. Прикладом таких робіт також слугує процес видобутку штольні буровибуховим способом. За одну ітерацію буріння, підриву та вивезення породи видобувається лише невелика частина штольні, тому ці роботи мають повторюватися доки запланована ділянка штольні не буде видобута повністю [13-14].

Іншим важливим фактором є те, що роботи на конкретному елементі структури проводяться не постійно. Наприклад, після видобутку певного блоку або виробки, роботи на цих елементах завершуються. Або деякі шахти чи штольні можуть бути тимчасово заморожені чи законсервовані з різних причин. Крім повного припинення робіт на елементі, можуть буди припинені лише деякі види робіт. Наприклад, роботи з видобутку зупиняються на час проведення розвідних та геологічних робіт[13-14]. Отже перевірка якості звітності про виконання завдань конкретного елементу структури має проводитися лише для тих періодів часу, коли елемент був активний та на ньому проводилися відповідні типи робот.

Як було зазначено вище, за результатами кожної з цих робіт оформлюється звіт, що потрібно зберігати для подальшого аналізу. Перелік документів, з яких складається звіт, варіюється для кожного окремого підприємства та формується з урахуванням методу видобутку родовища (закритий чи відкритий), методу видобутку саме породи (буровибуховий чи комбайновий) та програмного

забезпечення, яке використовується на підприємстві. Перелік необхідних документів має бути зазначений у паспорті процесу видобутку. Також важливим є той факт, що планові показники можуть відрізнятися від фактичних, тож доцільно буде мати змогу зберігати обидва варіанти для подальшого їх порівняння та корегування плану.

Частота складання звітів також зазначається у паспорті процесу видобутку, та може залежати від часу(наприклад, щотижнево чи щомісячно) або від прогресу видобутку(після повного видобутку блоку чи після проходження ділянки виробки певної довжини) [13-14].

Наявність суттєвих відмінностей у документообігу в різних підприємствах, а саме відмінність у переліку документів, що мають додаватися до проекту, паспорту процесу видобутку, звітів про виконання робіт тощо, вказує на необхідність створення універсального методу зберігання документів.

Крім універсальності, має бути можливість зберігати звіти для кожного окремого елементу структури та групувати звіти за типами проведених робіт. Наприклад, для кожної ділянки штольні, видобутої буровибуховим способом, повинні зберігатися звіти про виконанні роботи з буріння, вибуху, вивезення породи у двох екземплярах: план проведення робіт та їх фактичний результат. Зазвичай планові показники включено у проект проведення робіт, але можуть буди винесені до окремого документу. Оскільки цей процес є ітеративним, має зберігатися звіт про результати виконання кожної ітерації.

З наведеного вище можна зробити висновок, що конкретний документообіг конкретного гірничодобувного підприємства залежить від багатьох факторів, тож метод зберігання та аналізу якості звітності має бути максимально універсальнім. Проте можна виділити основні етапи документообігу, що є на кожному гірничодобувному підприємстві. Проаналізувавши наведене вище, розроблено узагальнену схему документообігу гірничодобувного підприємства (рисунок 1.2).

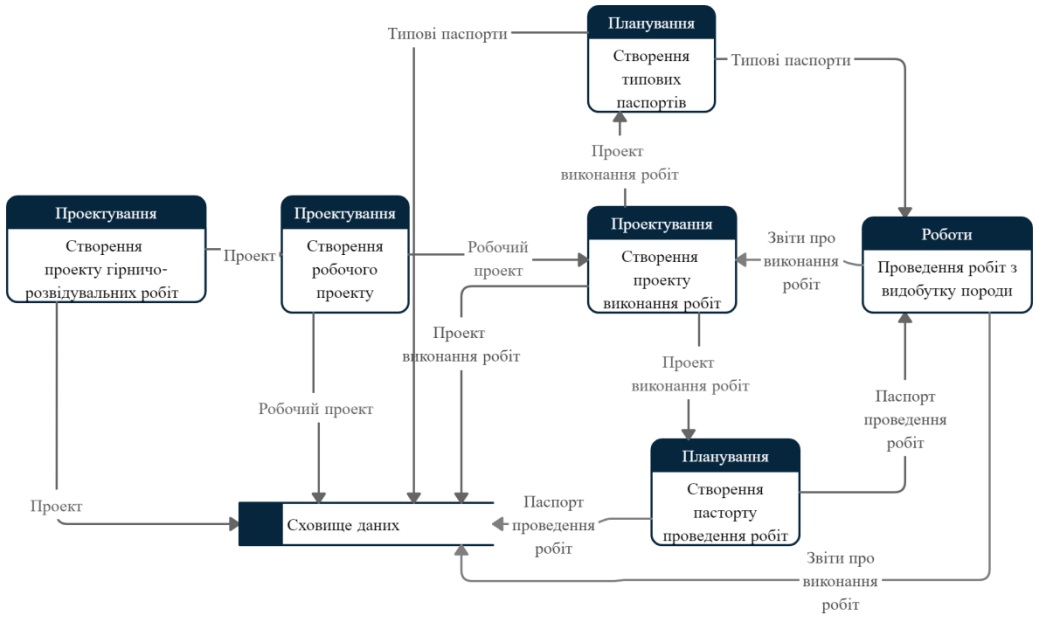


Рисунок 1.2 – Узагальнена схема документообігу гірничодобувного підприємства

1.3 Аналіз вимог до завдань на гірничодобувному підприємстві

З наведеного вище аналізу документообігу, можна зробити висновок, що на різні елементи структури підприємства видаються різні завдання з урахуванням типу робіт, що мають буди проведені. Наприклад, для блоків чи виробок видаються завдання на визначення порядку буріння та вивозу породи, а для горизонтів – завдання визначення кількості блоків, їх положення у просторі та розміру. Для родовища – завдання створення проекту виконання робіт, аналізу ґрунту, розрахунок цільових показників тощо.

Виконання кожного виду робіт завершується створенням звіту. Без правильно заповненого звіту завдання не може вважатися виконаним. Перелік документів та вимог до їх змісту варіюється в кожному підприємстві та для кожного виду завдань, але на окремо узятому підприємстві є досить чітким та структурованим та створюється під час складання паспорту процесу видобутку, бо від цього залежить ефективність виконання завдань та їх аналізу.

Як було вказано у розділі 1.2, звіт складається або періодично (щомісячно, щотижнево тощо), або за фактом виконання певного обсягу робот (повний

видобуток конкретного блоку або видобуток частини виробки певної довжини). У першому випадку звіт має складатися щонайменш один раз за зазначений період. У другому випадку зазвичай існують терміни виконання запланованого обсягу робіт, тож звіт має бути складено не пізніше цього терміну, не зважаючи на те, чи завершено запланований обсяг робіт.

Подання звіту у чіткі терміни дає змогу на їх основі корегувати плани на наступний період та перераховувати планові показники.

Виходячи з наведеного вище існує два критерії до виконання завдань:

– своєчасність складання звіту;

– повнота заповненого звіту.

Головною вимогою для усіх завдань є своєчасність виконання. Наприклад, якщо план видобутку блоків створюється на тиждень, то на кінець кожного тижня має бути створено новий план для продовження робіт.

Повнота заповнення звіту залежить від вимог конкретного підприємства до конкретних видів завдань. Проте перелік необхідних документів на конкретному підприємстві є чітким, структурованим та, як правило, задає вимоги до кількості документів конкретного формату та їх назв.

Виходячи з наведеного вище можна зробити висновок, що якісним вважається звіт, що було вчасно складено та який містить усі необхідні документи з правильними назвами. При виконанні цих умов, робота зі звітною інформацію є швидкою та ефективною.

1.4 Визначення взаємозв’язку між елементами структури гірничодобувного підприємства та вимогами до завдань

Відповідно до зазначеного у розділі 1.2, на різні елементи структури видаються різні завдання. Різні види завдань мають різний формат звітної інформації та терміни завершення.

Наприклад, роботи з видобутку породи для конкретного блоку мають проводитися щоденно, а звіт про результати виконання цих робіт має складатися

щомісячно чи по факту завершення видобутку блоку але не пізніше зазначеного терміну для використання цього звіту при створенні наступного плану. Таким чином для завдань, що видаються для блоків, вимогами буде щомісячне складання звіту та наявність у звіті необхідних документів, що будуть зазначати, наскільки було видобуто блок, скільки фактичної породи та руди було з нього вивезено та скільки ітерацій буріння та вибуху було пройдено за місяць тощо. Повний перелік звітної інформації задається у паспорті процесу видобутку.

Для елементів структури, що відповідають кар'єрам, мають щорічно проводитися роботи з уточнення положення кар'єру, його розміру та планових показників. Отже, щорічно має складатися звіт з результатами проведення цих робіт. Таким чином вимогами для завдань, що видаються для кар'єру, буде щорічне складання звітів та наявність у звіті документів, що відображують уточнені положення, розміри та планові показники кар'єру та інше.

Кожен набір звітів для кожного елементу структури уявляє собою зліпок фактичного стану елементу на момент складання звіту. Оскільки звіт, як правило, містить досить повний та різнобічний опис стану елементу, доцільним буде називати набір звітної інформації для конкретного елементу структури моделлю цього елементу на певний момент часу.

Для окремого виду завдань(буріння, підрив, топозйомка тощо) мають, з відповідною періодичністю, складатися звіти, тобто створюватися моделі. Оскільки різні типи робіт мають різний перелік звітної інформації, буде доцільно групувати моделі за типами цих робіт та запровадити термін «тип моделі». Таким чином для моделей одного типу, тобто для звітів про виконання завдань одного типу, будуть задаватися однакові вимоги для виконання.

Також з наведеного вище слідкує, що однакові за типом елементи структури (блоки, горизонти, кар'єри тощо) мають однаковий перелік робіт, а відповідно однакові завдання та переліки звітної інформації. Доцільно буде запровадити термін «категорія структурного елементу». Таким чином, елементи структури, однакові за своїм призначенням та суттю, будуть мати однакову категорію.

З наведеного вище видно, що взаємозв’язок між елементами структури та вимогами до виконання завдань є досить чітким та однозначним. Кожний елемент структури належить до певної категорії. Залежно від категорії, на елементі структури мають проводитися відповідні види робіт. Після проведення певного виду робіт має складатися звіт відповідно до вимог цього типу робіт. Схему взаємозв’язків зображено на рисунку 1.3.

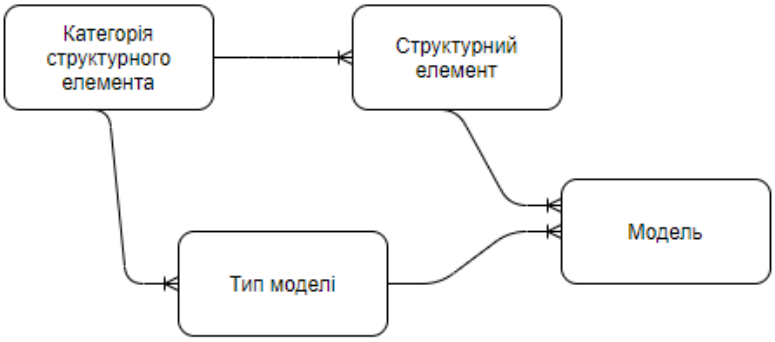


Рисунок 1.3 – Схема взаємозв’язку елементу структури з вимогами до завдань

1.5 Аналіз існуючих методів вирішення проблеми

На сьогоднішній день найбільш розповсюдженим є метод зберігання звітів у вигляді файлів на комп’ютері, сервері підприємства, чи хмарному сховищі. Для організації процесу складання звітів у наведених варіантах використовується ієрархічна система папок, кожна з яких є конкретним елементом структури підприємства.

Перевагою цих методів є простота та зрозумілість, адже кожен знайом з файловою системою.

Недоліком зберігання звітів на своєму сервері чи комп’ютері також є необхідність створення резервних копій для запобігання можливості їх втрати або пошкодження. При використанні хмарного сховища від стороннього провайдера відповідальність за збереження даних перекладається на провайдера. Іншим недоліком є швидкість, адже при завантаженні папки у файловому менеджері,

завантажується деяка кількість додаткової інформації, як то розмір файлів, дата створення та останньої зміни тощо. Також істотним недоліком цього підходу є складність організації паралельного доступу до файлів.

Проте при зберіганні даних у хмарному сховищі втрачається можливість повного контролю над даними. Можливості маніпуляції з даними обмежена тими функціями, що надає провайдер. Також при використанні хмарного сховище виникає проблема кількості даних, адже мова йде про великі об'єми даних, кількість яких постійно збільшується. Не усі провайдери можуть надати можливість зберігати необхідну кількість даних, або плата за це буде надто великою. Не менш важливим недоліком хмарних сховищ є необхідність стабільного та широкого доступу до мережі Інтернет, що не завжди можливо на гірничодобувних підприємствах, особливо безпосередньо на місці роботи [15].

Також можуть буди використані стороні програмні продукти, що дозволяють або оптимізувати процес роботи з файловою системою, або замінити його повністю на роботу з іншим програмним забезпеченням.

Цікавим варіантом є програмний продукт «Скан-Архив», що дозволяє зберігати електронні копії документів з прив'язкою до бази даних «1С», що значно спрощує процес знаходження відповідної звітної інформації для подальшого її аналізу [16]. Проте бази даних «1С» майже не використовуються на гірничодобувних підприємствах.

Суттєвим недоліком усіх наведених вище методів зберігання звітів є неможливість пошуку необхідних елементів структури за їх категорією чи моделей за їх типом.

Універсальних систем аналізу якості виконання завдань, які б перевіряли своєчасність, регулярність та повноту складання звітів, у загальному доступі знайдено не було. Існують окремі рішення, створені на базі деяких установ та підприємств, доступні лише усередині підприємств.

Також існує ряд організацій, що займаються створенням подібних систем на основі баз «1С». Такі рішення користуються попитом серед гірничодобувних підприємств, що мають у своєму складі декілька філіалів, на кожному з яких

історично склались різні вимоги до звітної інформації. Проблеми цього роду виникають при об’єднанні підприємств або за умови, що один з філіалів був запроваджений значно пізніше інших.

У якості альтернативного рішення можна розглянути різні програмні продукти, призначені для планування задач. Прикладами можуть слугувати Jira [17], Notion[18], Trello[18]. Наведені системи надають можливість створювати терміни виконання завдань та складання звітів, зазначати перелік дій, що мають буди виконані для завершення завдань та додавати файли у якості звітів для виконання завдань. Але вони не дають змоги задавати перелік необхідних файлів для звіту, не мають можливості групувати завдання у ієрархічну структуру та мають жорсткі обмеження щодо розміру доданих файлів.

З наведеного вище можна зробити висновок, що натепер не існує систем зберігання та аналізу звітів про виконання завдань, що задовольняють умовам гірничодобувного підприємства.

1.6 Постановка задачі

Об’єктом розробки э система зберігання та аналізу якості звітності про виконання завдань виробничим персоналом (на прикладі гірничодобувного підприємства).

Вхідними даними для роботи системи є звітність про виконання завдань, що створюється відповідно до процесу документообігу підприємства (див. 1.2) та структура підприємства, відповідно до якої буде зберігатися звітність (див. 1.1).

Вихідними даними є результат аналізу якості звітності про виконання завдань, який може бути збережений у доступному для користувача форматі та використаний для запровадження заходів з покращення якості звітності та ефективності праці виробничого персоналу.

Проблемами, що потрібно вирішити у процесі розробки, є відсутність чітких критеріїв до виконання завдань, за якими можна оцінити якість звітності (перелік необхідних документів та терміни виконання завдань варіюються для

кожного підприємства), складна система взаємозв’язків між елементами структури та критеріями до виконання завдань, та відсутність єдиного шаблону структури гірничодобувного підприємства.

Отже, у процесі розробки системи необхідно розробити відповідну універсально структуру бази даних, що дала б змогу зберігати:

- ієрархічну структуру підприємства;

- звіти про виконання завдань відповідно до типів робіт, що виконувалися у рамках цього завдання, та елементів структури підприємства, на яких ці завдання виконувалися;

- критерії до звітів відповідно до типу проведених робіт;

- усі взаємозв’язки між елементами структури підприємствами, звітами та критеріями до виконання завдань (див. 1.4).

Також необхідно розробити методи аналізу якості виконання завдань за критеріями, розглянутими у розділі 1.3. Результат аналізу повинен однозначно вказувати на проблемні елементи структури підприємства та типи робіт, що були виконані не повністю (не усі необхідні документи присутні у звіті з виконання завдання) або не були виконані у зазначений термін(звіт не було завантажено до системи).

Поєднання методів аналізу якості звітності та бази даних, у якій зберігається звітність, дасть змогу створити універсальну систему, що вирішить наведені вище проблеми та дасть змогу більш ефективно аналізувати якість звітності та запроваджувати заходи з її покращення.

2 РОЗРОБКА МЕТОДІВ ПЕРЕВІРКИ ЯКОСТІ ВИКОНАННЯ ЗАВДАНЬ

2.1 Визначення вимог до бази даних для зберігання звітів

Узагальненою вимогою до бази даних для зберігання звітів є урахування усіх особливостей взаємозв’язків між елементами структури, моделями, типами моделей та категоріями структурних елементів, що були визначені у розділі 1.4.

З наведеного у розділі 1.4 слідкує, що центральним елементом у структурі зберігання звітів має бути структурний елемент рудника, бо завдання виробничому персоналу видаються безпосередньо на елементи структури рудника та звіт про виконання завдання складається для кожного елементу структури окремо.

Оскільки структура рудника є ієрархічною, кожен елемент структури, окрім кореневого, має однозначно визначати елемент вищого рівня, до якого належить. Кореневий елемент не має вищого або батьківського елементу. Також кожен елемент структури має зберігати інформацію про свою категорію, для визначення переліку робот, що мають на ньому проводитися, та свою назву, для однозначної ідентифікації.

Також у базі даних мають зберігатися категорії структурних елементів та типи моделей.

Категорія структурного елементу має назву для однозначної ідентифікації та слугує для визначення переліку робіт, що будуть проводитися на конкретному елементі структури, тобто пов’язувати структурні елементи та типи моделей.

Як зазначено у розділі 1.4, тип моделі уявляє собою конкретний вид робіт, що мають бути проведені на структурному елементі відповідної категорії. Таким чином, у типі моделей має зберігатися його назва, періодичність виконання робіт та посилання на категорію структурного елементу. Перелік необхідних звітних документів з зазначенням їх кількості, доцільно буде винести до окремої сутності бази даних. має мати назву, для визначення типу та змісту необхідного

документу, маску назви документу, для можливості перевірки правильності назви та формату документів.

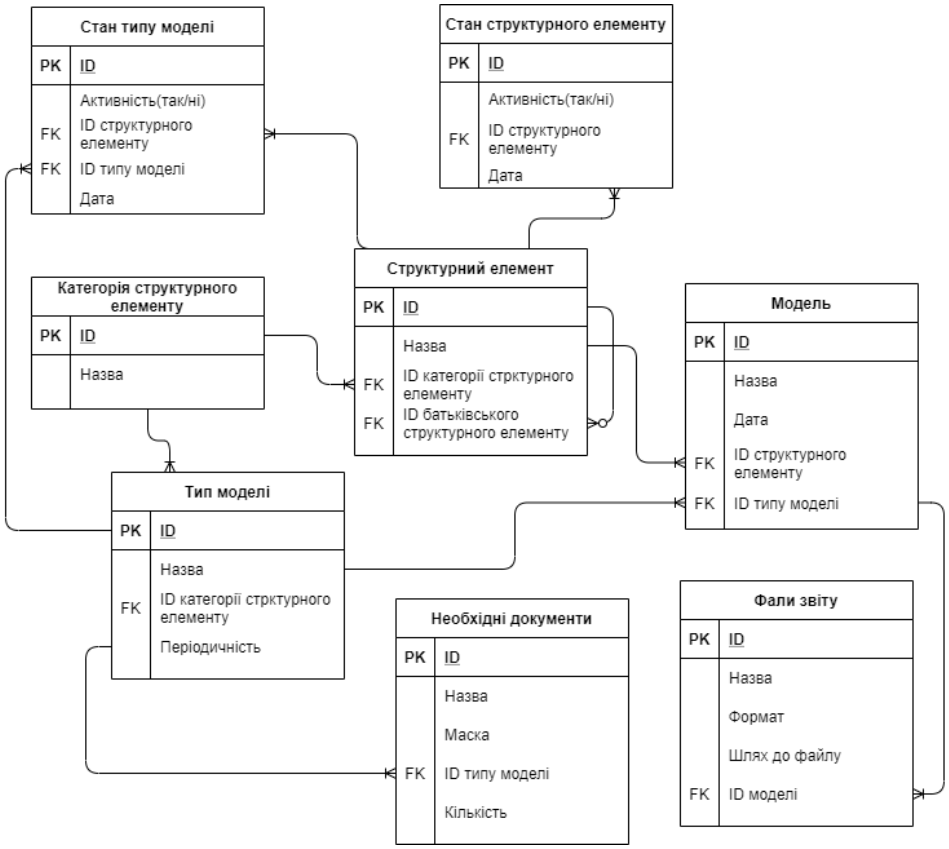
Звітна інформація буде зберігатися у моделях, які описані у розділі 1.4. Модель може мати ім’я, для уточнення змісту звітної інформації, та обов’язково має дату для перевірки своєчасності складання звіту. Також модель має посилання на тип моделі (для визначення виду робіт, звіт про виконання котрих у ній зберігається) та на елемент структури, на якому ці роботи проводилися.

Для зберігання переліку документів, що додано до звіту, має бути створено сутність файлу, що матиме назву, формат, посилання на модель, тобто на звіт до якого належить. Оскільки зберігати файли безпосередньо у базі даних є недоцільно, то необхідна додаткова інформація для знаходження файлу у системі. У найпростішому випадку це може бути шлях до файлу у файловій системі серверу, на якому він зберігається.

Також при аналізі документообігу гірничодобувного підприємства у розділі 1.2 було визначено, що елементи структури не є постійно активними. Та не усі типи робіт постійно проводяться на відповідному елементі структури. Інформація про періоди активності елементів структури також має зберігатися у базі даних. Доцільним буде зберігати не усю історію активності, а лише ті дати, коли роботи на елементі структури були розпочаті, припинені чи продовжені. Таку саме інформацію потрібно зберігати й для типів моделей, але за прив’язкою до структурного елементу. Прив’язка потрібна, для можливості припинення певного виду робіт лише на конкретному елементі структурному елементі.

З наведеного вище, та з урахуванням взаємозв’язків елементів, визначених у розділі 1.4, створено узагальнену схему (рисунок 2.1) бази даних, що є бажаною для повної реалізації методу.

Схема надає лише загальне уявлення про зв’язки між елементами та зміст елементів та не є логічною чи фізичною моделлю бази даних.



(FK-foreign key, зовнішній ключ; PK – primary key, головний ключ; ID – identifier, ідентифікатор)

Рисунок 2.1 – Узагальнена схема бази даних

З наведеного вище, та з огляду створеної схеми бази даних однозначно видно, що для реалізації системи необхідно використовувати технології реляційних баз даних, що мають підтримку зовнішніх ключів [19].

2.2 Критерії оцінки якості виконання завдань

Критерії оцінки якості виконання завдань ґрунтуються на вимогах до виконання завдань, що були визначені у розділі 1.3. З наведеного у розділі 1.3 слідкує, що існує дві вимоги до виконання завдань на гірничодобувному підприємстві, які і є критеріями якості виконання завдання: звіт складено та внесено до системи у заданий термін, звіт має увесь перелік необхідних документів.

Термін виконання завдання, складання звіту та внесення звіту до системи визначається відповідно до типу робіт, що мають бути виконання у рамках завдання та зазначається у відповідному паспорті процесу видобувних робіт.

За терміном завершення та складання звіту завдання можна поділити на два типи: періодичні та не періодичні чи процесові.

Періодичні завдання виконується щонайменш один раз за зазначений період (тиждень, місяць тощо). З такою ж періодичністю складається звіт. Прикладом таких завдань є створення плану проведення робіт на наступний зазначений період.

Не періодичні чи процесові завдання видаються на повний видобуток блоку чи його частини, одноразові завдання, або видобуток ділянки виробки певної довжини. Звіт про виконання цих завдань складається за фактом повного їх виконання, але не пізніше зазначеного у плану терміну за умови відставання від плану.

Таким чином, якісно виконаним завданням за критерієм терміну є завдання, звіт з про виконання якого складається:

– щонайменш одного разу за період для періодичного завдання;

– за фактом завершення завдання, але не пізніше запланованого терміну для неперіодичного завдання.

Якісно виконаним завданням за критерієм переліку документів вважається завдання, звіт про виконання якого містить усі необхідні документи у достатній кількості.

З урахуванням описаних взаємозв’язків у розділі 1.4, можна запровадити наступне визначення якісно виконаного завдання:

Якісно виконаним вважається завдання, за фактом виконання якого у заданий термін було створено та додано до відповідного структурного елементу підприємства звіт про виконання робіт відповідного типу, що містить увесь перелік необхідних документів.

Наступна розробка методу зберігання та оцінки якості звітності про виконання завдань виробничим персоналом буде засновуватися на цьому визначенні.

2.3 Розробка методів аналізу якості виконання завдань

Аналіз якості виконання завдань за критерієм переліку необхідних документів є досить простою та є послідовною перевіркою усіх моделей, а саме набору доданих до них файлів, на відповідність заданому переліку необхідних документів.

Таким чином для кожної окремої моделі процес перевірки якості за критерієм наявності необхідних документів матиме наступні кроки, відповідно до розділів 2.2, 2.1, 1.4 та 1.3:

– визначення переліку необхідних документів, відповідно до типу моделі;

– знаходження усіх файлів моделі;

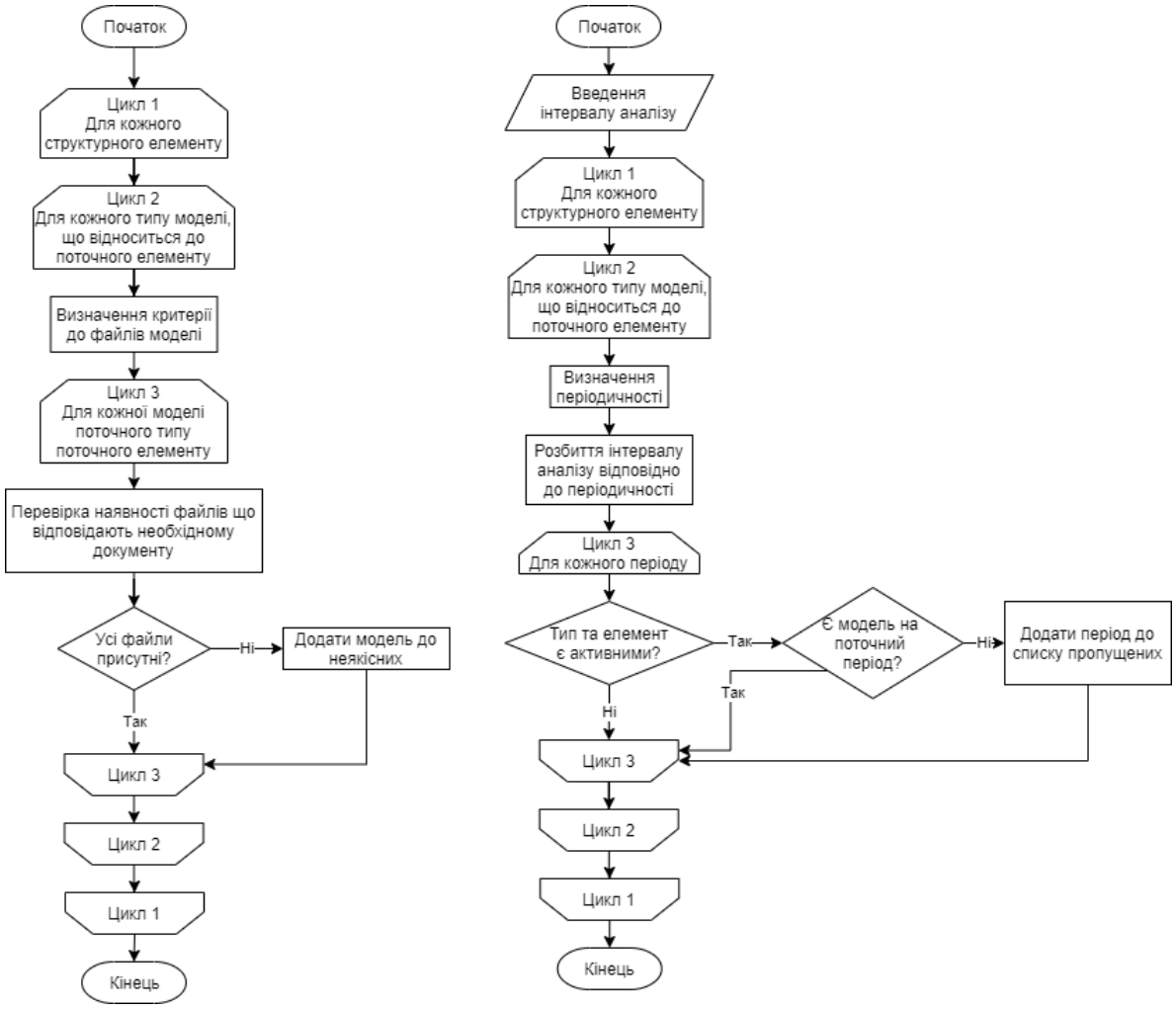
– для кожного необхідного документу проводиться пошук відповідних файлів за маскою назви;

– якщо необхідні файли знайдені у достатній кількості, то модель є якісною за критерієм наявності необхідних документів;

– якщо не усі необхідні файли були знайдені, або знайдені у недостатній кількості, то модель є неякісною за критерієм наявності необхідних документів.

Іншим підходом до процесу аналізу якості моделей за критерієм наявності необхідних документів є орієнтованість на структурні елементи. Його відмінність полягає у порядку вилучення даних з бази даних. Так, у наведеному вище процесі спочатку вилучалася інформація про усі наявні моделі, після чого для кожної моделі виконувалися наведені вище кроки. У підході зі сторони структурних елементів спочатку вилучається інформація про усі структурні елементи, після чого для кожного пов’язаного з ним типу моделі, вилучається інформація про моделі та перелік необхідних документів. Саме цей підхід було обрано для подальшого розгляду, бо він краще поєднується з перевіркою якості моделей за

критерієм своєчасності. Схема алгоритму перевірки цього критерію наведена на рисунку 2.2 (а).



(а) *(б)*

Рисунок 2.2 – Схема алгоритму перевірки якості моделі за критерієм наявності необхідних документів (а) та за критерієм своєчасності (б)

Перевірка якості виконання завдань за критерієм своєчасності є дещо складнішою. Для неперіодичних завдань звіт має бути складено за фактом виконання, але не пізніше зазначеного терміну. Таким чином для перевірки якості неперіодичних завдань достатньо перевірити наявність звіту та порівняти заплановану дату виконання завдання з датою складання звіту(датою створення моделі).

Для періодичних завдань перш за все необхідно визначити мінімальний необхідний перелік періодів, через які можуть виконуватися завдання. У розділах 1.2, 1.3 було вказано наявність щотижневих, щомісячних та щорічних робіт. Також у роботі [13] зазначаються завдання, що виконуються щоквартально та щовахтно (раз у два тижні).

Отже, завдання можуть мати наступні періодичності:

– тижнева: завдання виконується щонайменш раз на тиждень. Кожний місяць має чотири тижні;

– вахтна: завдання виконується щонайменш раз на два тижні або двічі на місяць;

– місячна: завдання виконується щонайменш раз на місяць;

– квартальна: завдання виконується щонайменш раз на квартал;

– річна: завдання виконується щонайменш раз на рік.

Цей перелік дозволяє повністю покрити потреби підприємства з перевірки періодичних завдань.

Процес аналізу якості виконання завдань за критерієм термінів складання звітів містить наступні етапи:

– визначення інтервалу часу, на якому проводиться аналіз;

– визначення періодичності складання звітів для кожного типу робіт кожного структурного елементу;

– розбиття інтервалу проведення аналізу на відповідні періоди (тижні, місяці, вахти тощо) для кожного структурного елементу та кожного типу робіт;

– перевірка, чи є структурний елемент та тип моделі для цього структурного елементу активними у конкретний період. Якщо вони не активні, то переходимо до наступного періоду (тижня, вахти тощо);

– перевірка наявності моделі конкретного типу конкретного структурного елементу у конкретний період. Якщо модель є, то завдання є якісним за критерієм терміну складання звіту.

Для більш зрозумілого уявлення про процес аналізу якості звітів про виконання завдань виробничим персоналом за критерієм своєчасності було створено схему алгоритму цього процесу (див. рисунок 2.2 (б)).

Після аналізу алгоритмів перевірки обох критеріїв, було розроблено загальний алгоритм перевірки якості виконання завдань виробничим персоналом. Схему алгоритму наведено на рисунку 2.3.

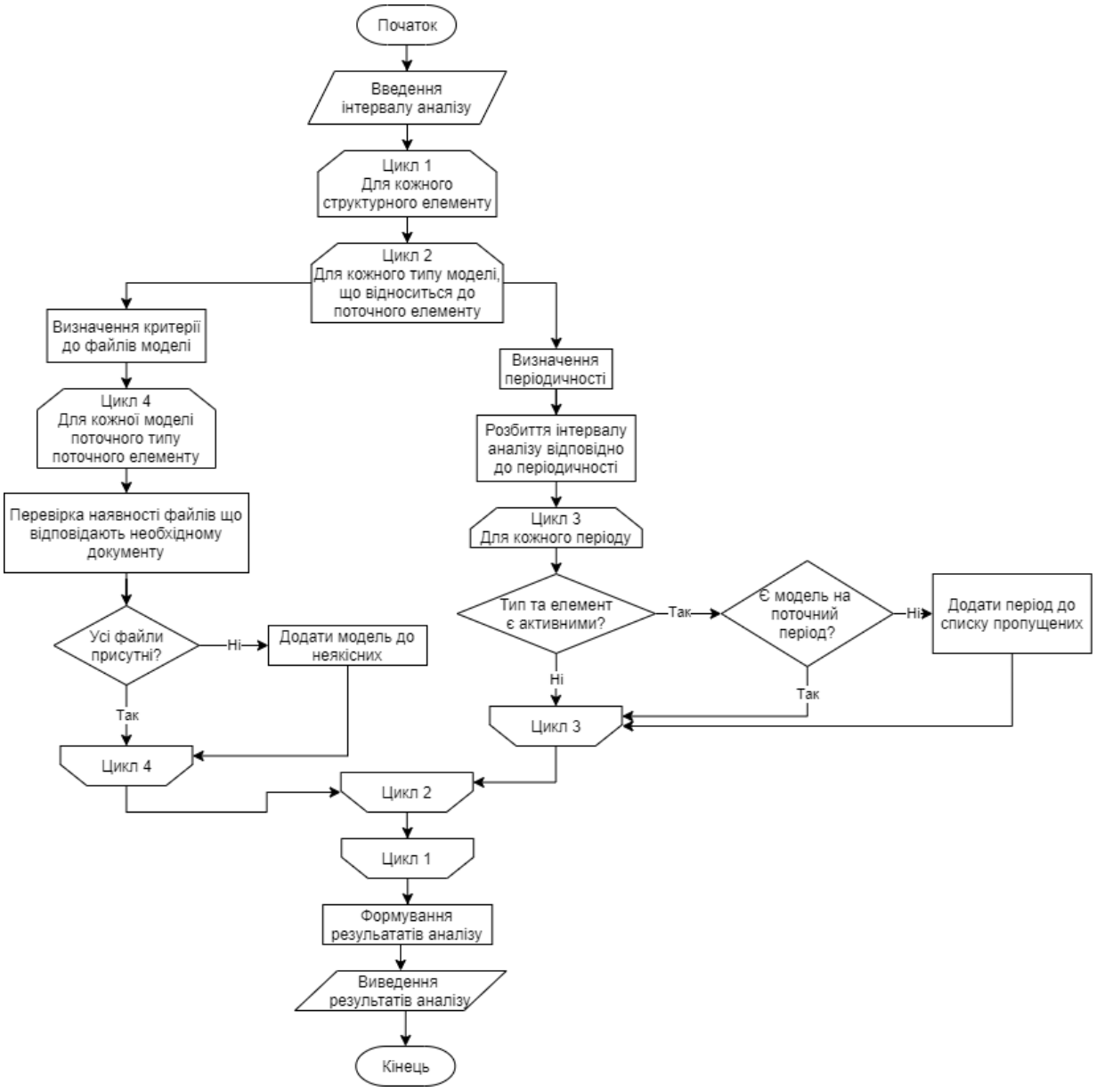


Рисунок 2.3 – Загальний алгоритм аналізу якості виконання завдань виробничим персоналом

2.4 Результати аналізу якості звітності про виконання завдань

Результати проведеного аналізу мають надавати чітке уявлення про якість звітності з виконання завдань виробничим персоналом, а саме чітко вказувати на ті структурні елементи та види робіт на цих елементах, звіти з виконання яких не є якісними.

З визначення, запровадженого у розділі 2.2, слідкує, що у результатах аналізу якості звітності про виконання завдань мають відображатися для кожного структурного елементу:

– періоди, у які не було створено моделі для відповідних типів моделей, якщо завдання є періодичним;

– типи моделей, що не мають моделей, якщо завдання не є періодичним;

– моделі, що не мають файлів або перелік файлів не відповідає критеріям, зазначеним у відповідному типі моделі.

Оскільки аналіз проводиться для кожного структурного елементу, буде доцільним у результатах вказувати повний шлях у ієрархії до елементу, результати якого зображуються. Наприклад, для елементу «Кар’єр» (див. рисунок 1.1) буде вказано наступний шлях: Філіал\Родовище 1\Кар’єр. Окрім структурних елементів, необхідно вказати часовий проміжок, на якому проводився аналіз, тип моделі та його періодичність. Також доцільним буде вказати базу даних, з якої були вилучені дані для аналізу.

Результати аналізу мають бути згруповані за структурним елементом та типом моделі. Потреба у такому групуванні є наслідком взаємозв’язку структурних елементів, типів моделей, моделей та вимог до моделей, що були описані у розділах 2.1 та 1.4. Завдяки такому групуванню можна буде чітко розділити моделі, файли яких перевірялися за однаковим переліком необхідних документів, що мають однакову періодичність та відносяться до одного структурного елементу.

Для візуалізації результатів аналізу якості звітності про виконання завдань, було обрано табличне подання. Такий вибір було зроблено, бо табличне подання є

легким для аналізу людиною, структурованим, має можливість для доповнення новими даними за необхідністю, та може зберігатися у файлах звичного для користувача формату. Також табличне подання дає змогу вказати усі необхідні данні та згрупувати результати вказаним вище способом. Для створення прикладу звіту було обрано програму Microsoft Excel. Приклад табличного подання результатів аналізу якості наведено на рисунку 2.4.

	A	B	C	D	E
1					
2	**База даних:**	.\SQL_2019\TestBase			
3					
4	**Струтурний елемент**	**Період**		**Тип моделі**	**Періодичність**
5		**з**	**по**		
6	Філіал\Родовище 1\ Кар'єр 1	01.01.2019	22.12.2020	Проходка за тиждень	Тижднева
7					
8	**Період**	**Вид помилки**	**Модель**		
9	3 тиждень жовтень 2020	Відсутні моделі			
10	2 тиждень вересень 2020	Відсутні моделі			
11	3 тиждень червень 2019	Неправильний набір файлів	16.06.2019		
12	4 тиждень червень 2019	Модель без файлів	30.06.2019		
13	3 тиждень серпень2019	Модель без файлів	22.08.2019		
14	4 тиждень серпень 2019	Модель без файлів	30.08.2019		
15	2 тиждень вересень 2019	Неправильний набір файлів	09.09.2019		
16			12.09.2019		

Рисунок 2.4 – Приклад виведення результатів аналізу якості звітності про виконання завдань виробничим песоналом

Проаналізувавши приклад результатів роботи з рисунку 2.4, можна сказати, що у структурному елементі «Підземний рудник» не усі звіти про виконання завдань типу «Проходка за тиждень» за період з 01.01.2019 по 22.12.2020 є якісними.

А саме:

– звіти про виконання щотижневих завдань не були складені для 2-го тижня вересня 2020 та 3-го тижня жовтня 2020;

– звіти, що складені за 3-й тиждень червня 2019 та 2-й тиждень вересня 2019, не мають усіх необхідних файлів;

– звіти, що складені за 3-й й 4-й тиждень серпня 2019 та 4-й тиждень червня 2019, не мають файлів зовсім.

На основі аналізу цих результатів необхідно провести відповідні заходи з поліпшення результатів праці персоналу.

3 РОЗРОБКА БАЗИ ДАНИХ

Розробка бази даних буде проводитися з урахуванням особливостей бази даних, що використовується у програмному забезпеченні Mine Advisor, до якого буде інтегровано розроблювану систему.

Mine Advisor – це програмне забезпечення, розроблене компанією SightPower Inc. Це програмне забезпечення розроблене для керування (заповнення даними, модифікування даних та вилучення з даних корисної інформації) спеціалізованими базами даних для гірничодобувних підприємств [20-21].

Розроблювану систему було вирішено інтегрувати до Mine Advisor, бо це програмне забезпечення має розвинену інфраструктуру, що включає до себе такі необхідні компоненти як: менеджер підключень до бази даних, сторінку підключення до бази даних, власне сховище файлів й інтерфейс взаємодії з ним та набір базових класів й компонентів для створення нових майстрів й включення їх до інтерфейсу користувача. Окрім безпосередньо необхідних для розроблюваної системи компонентів, Mine Advisor також має велику кількість допоміжних функцій: візуалізація даних у 2D та 3D; імпорт та конвертація даних різноманітного формату для подальшого їх відображення, зберігання та обробки; створення фонових процесів обробки даних та інтерфейс взаємодії з ними (зчитування прогресу,пауза, зупинка та продовження виконання)[21].

3.1 Розробка логічної моделі бази даних

Розробка логічної моделі бази даних була виконана за допомогою CASE-засобу ERwin Data Modeler за стандартом мови графічного моделювання баз даних IDEF1x [22].

Розроблену логічну модель бази даних зображено на рисунку 3.1.

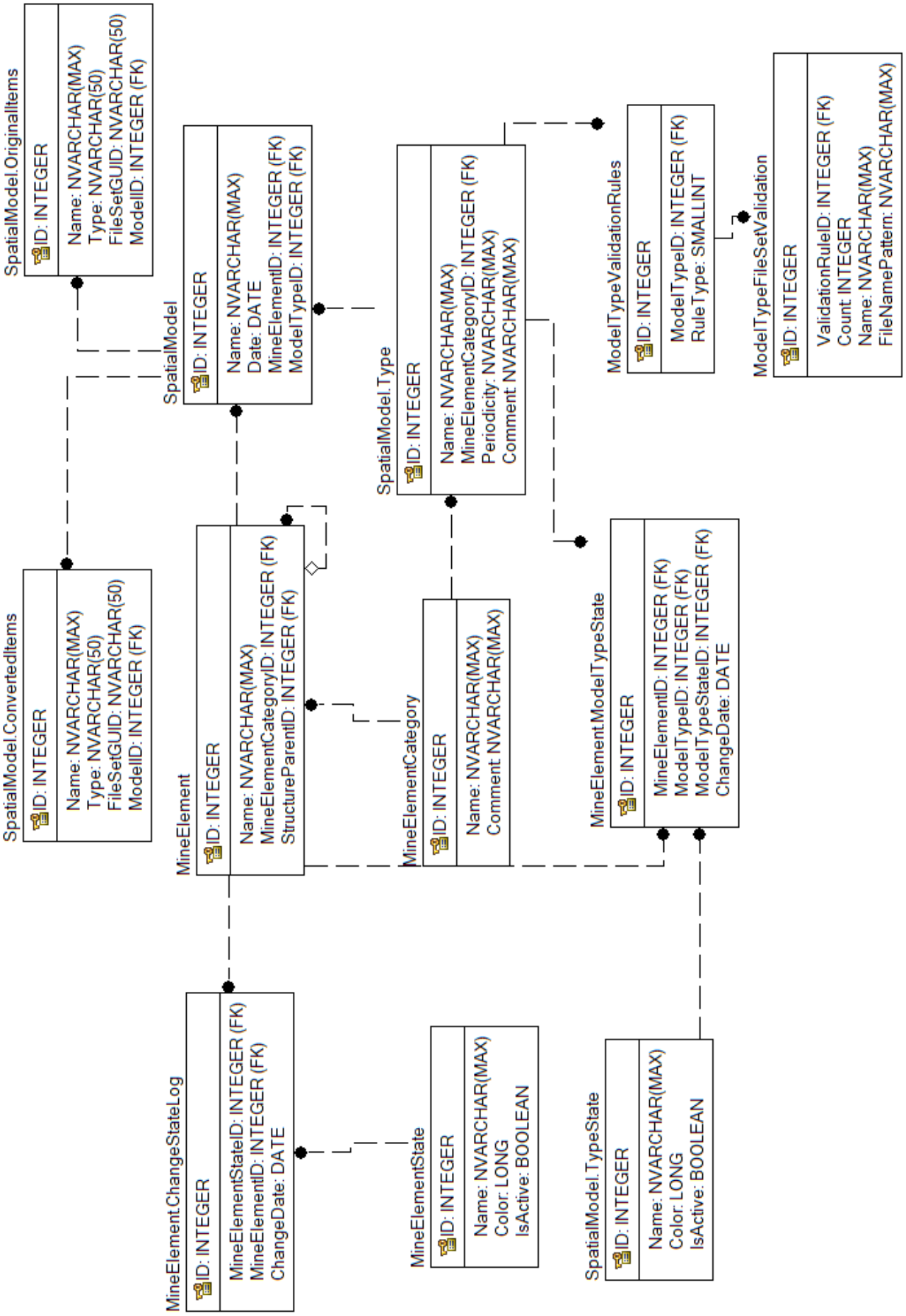


Рисунок 3.1 – Логічна модель бази даних

Центральним елементом бази даних виступає сутність MineElement, що є відображенням елементу структури підприємства. Сутність MineElement вже присутня у базі даних Mine Advisor та має велику кількість додаткових атрибутів. Додаткові атрибути не відображені на логічній моделі бази даних, бо вони не використовуються у роботі розроблюваної системи.

Сутність MineElement має наступні атрибути:

– ID – ціле 32-х бітне число. Є основним ключем (PK, Primary Key);

– Name – рядок символів юні коду. Зберігає назву структурного елементу;

– MineElementCategoryID – ціле 32-х бітне число. Є зовнішнім ключем (FK, Foreign Key) та слугує для зв’язку сутності MineElement з сутністю MineElementCategory. Вказує категорію структурного елементу;

– StrictireParentID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності MineElement з собою. Вказує батьківський структурний елемент. Слугує для створення ієрархічної структури.

Сутність MineElementCategory є відображенням категорії структурного елементу та слугує для взаємозв’язку між структурними елементами одного типу(горизонти, блоки, родовища тощо) та видами робіт, що будуть на них проведені. Ця сутність також вже присутня у базі даних Mine Advisor, та її додаткові атрибути також не відображені. Сутність має наступні атрибути:

– ID – ціле 32-х бітне число, PK;

– Name – рядок символів юнікоду. Зберігає назву категорії структурного елементу;

– Comment – рядок символів юнікоду. Зберігає коментар.

Сутність SpatialModel.Type є відображенням типу робіт та має наступні атрибути:

– ID – ціле 32-х бітне число, PK;

– Name – рядок символів юнікоду. Зберігає назву типу робіт;

– Comment – рядок символів юнікоду. Зберігає коментар.

– MineElementCategoryID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності ModelType з сутністю MineElementCategory. Вказує категорію структурного елементу, для якої роботи цього типу будуть виконуватися;

– Periodicity– рядок символів юнікоду. Зберігає періодичність виконання робіт у форматі JSON (JavaScript Object Notation).

Сутність SpatialModel є відображенням звіту про виконання робіт та має наступні атрибути:

– ID – ціле 32-х бітне число, PK;

– Name – рядок символів юнікоду. Зберігає назву звіту;

– Date – дата. Зберігає дату складання звіту;

– ModelTypeID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності Model з сутністю ModelType. Вказує на тип робіт, для якого було складено звіт;

– MineElementID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності Model з сутністю MineElement. Вказує на елемент структури підприємства, для якого було складено звіт.

Оскільки Mine Advisor надає можливість імпорту та конвертації різноманітних даних до системи у вигляді слоїв з подальшою можливістю їх відображати у 3D та 2D [21], було вирішено створити можливість додавання цих імпортованих даних до звіту. Це дасть змогу зберігати не тільки документальне відображення стану елементу структури підприємства, а й відображати цей стан у 3D та 2D.

Таким чином запропонована у розділі 2.1 сутність файлу була розділена на дві сутності: SpatialModelOriginalItems та SpatialModelConvertedItems. Обидві сутності мають однакові атрибути:

– ID – ціле 32-х бітне число, PK;

– Name – рядок символів юнікоду. Зберігає назву файлу;

– ModelID – ціле 32-х бітне число, FK. Слугує для зв’язку з сутністю Model. Вказує на звіт, до якого цей файл належить;

– FileSetGUID – рядок символів юнікоду. Зберігає GUID (Globally Unique Identifier) фалу, що створюється під час імпорту до Mine Advisor та слугує унікальним ідентифікатором файлу у системі;

– Type – рядок символів юнікоду. Зберігає тип файлу.

Розподілення на дві сутності необхідне для полегшення подальшого пошуку конвертованих та оригінальних файлів та модифікації цих сутностей за необхідністю.

Усі наведені вище сутності використовуються для безпосереднього зберігання звітності з урахуванням особливостей системі підприємства та взаємозв’язків між елементами структури, видами робіт та звітами.

Сутності MineElementState та SpatialModel.TypeState є відображеннями можливих станів структурного елементу підприємства та типу робіт відповідно. Вони мають наступні атрибути:

– ID – ціле 32-х бітне число, PK;

– Name – рядок символів юнікоду. Зберігає назву файлу;

– Color – ціле 64-х бітне число. Зберігає колір стану, що буде використаний для маркування у інтерфейсі користувача;

– IsActive – логічне поле(може приймати лише значення «інтина» або «лож»). Вказує на те, чи проводяться роботи при присвоєнні цього стану.

Сутність MineElement.ChangeStateLog слугує для зберігання історії зміни станів структурних елементів підприємства та реалізує зв’язок «багато до багатьох»[19]. Має наступні атрибути:

– ID – ціле 32-х бітне число, PK;

– MineElementID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності MineElement.ChangeStateLog з сутністю MineElement. Вказує на елемент структури підприємства, стан якого змінюється;

– MineElementStateID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності MineElement.ChangeStateLog з сутністю MineElementState. Вказує на новий стан структурного елементу підприємства;

– ChangeDate – дата. Зберігає дату зміни стану структурного елементу підприємства.

Сутність MineElement.ModelTypeState слугує для зберігання історії зміни станів типів робіт для відповідних елементів структури підприємства та реалізує зв’язок «багато до багатьох»[19]. Має наступні атрибути:

– ID – ціле 32-х бітне число, PK;

– MineElementID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності MineElement.ModelTypeState з сутністю MineElement. Вказує на елемент структури підприємства, стан якого змінюється;

– ModelTypeStateID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності MineElement.ModelTypeState з сутністю ModelTypeState. Вказує на новий стан типу робіт;

– ModelTypeID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності Model з сутністю ModelType. Вказує на тип робіт, для якого було змінено стан;

– ChangeDate – дата. Зберігає дату зміни стану типу робіт для конкретного структурного елементу.

Сутність ModelTypeValidationRules є відображенням конкретного критерію якості звітності для конкретного типу робіт. Створена для можливості додавання нових критеріїв якості звітності та встановлення власного переліку критеріїв для кожного типу робіт. Має наступні атрибути:

– ID – ціле 32-х бітне число, PK;

– ModelTypeID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності ModelTypeValidationRules з сутністю ModelType. Вказує на тип робіт, для якого цей критерій якості звітності встановлено;

– RuleType – ціле 16-ти бітне число. Вказує на те, який саме критерій якості звітності встановлено, наприклад критерій наявності документів чи наявності конвертованих файлів.

Сутність ModelTypeFileSetValidation є відображенням вимог до переліку необхідних документів, що мають бути додані до звіту про виконання завдань конкретного типу. Має наступні атрибути:

– ID – ціле 32-х бітне число, PK;

– ValidationRuleID – ціле 32-х бітне число, FK. Слугує для зв’язку сутності ModelTypeFileSetValidation з сутністю ModelTypeValidationRules. Вказує на критерій якості, до якого належить. Матиме вплив на результат перевірки якості звітності лише якщо у відповідній сутності ModelTypeValidationRules атрибут RuleType вказує на необхідність перевірки за критеріям наявності відповідних документів;

– Count – ціле 32-х бітне число. Вказує на мінімальну кількість файлів, назва яких має відповідати регулярному виразу (regular expression), що зберігається у атрибуті FileNamePattern;

– Name – рядок символів юнікоду. Зберігає назву файлу;

– FileNamePattern – рядок символів юнікоду. Зберігає регулярний вираз для перевірки назви файлів у звіті.

Таким чином, було розроблено модель реляційної бази даних у третій нормальній формі [19], відповідає вимогам, що були висунуті у розділі 2.1, та враховує усі особливості взаємозв’язків між елементами структури підприємства та критеріями до якості звітності, що були розглянуті у розділі 1. Модель бази даних надає можливість подальшої зміни критеріїв якості звітності та ретельного налаштування вимог до звіту для кожного типу робіт, та структурного елементу.

3.2 Розробка фізичної моделі даних

Фізична модель бази даних розроблялася для Microsoft SQL Server (MSSQL Server) з використанням ERwin Data Modeler.

MSSQL Server – це система керування реляційними базами даних (РСКБД), що активно розроблюється та підтримується компанією Microsoft. Основною мовою MSSQL Server є T-SQL(Transact-SQL, Transact Structured Query Language) [23].

T-SQL – процедурне розширення мови SQL, що включає у себе додаткові функції та оператори, реалізація яких відсутня у SQL[23].

Для розробки фізичної моделі бази даних було обрано MSSQL Server, бо цю РСКБД було використано для створення основної бази даних Mine Advisor.

Процес перетворення логічної моделі бази даних на фізичну є повністю автоматизованим завдяки використанню ERwin Data Modeler. Проте отримана таким чином фізична модель потребує деяких змін, а саме:

а) створення послідовності для автоматичного заповнення головних ключів (PK) для кожної сутності;

б) зміни назв обмежень зовнішніх ключів (Foreign Key Constraint) з автоматично створених на більш змістовні;

в) уточнення типів даних деяких атрибутів:

– SMALLINT на TINYINT;

– BOOLEAN на BIT;

– DATE на DATETIME.

Після проведення цих змін було отримано фізичну модель бази даних для MSSQL Server та мови T-SQL, зображену на рисунку 3.2.

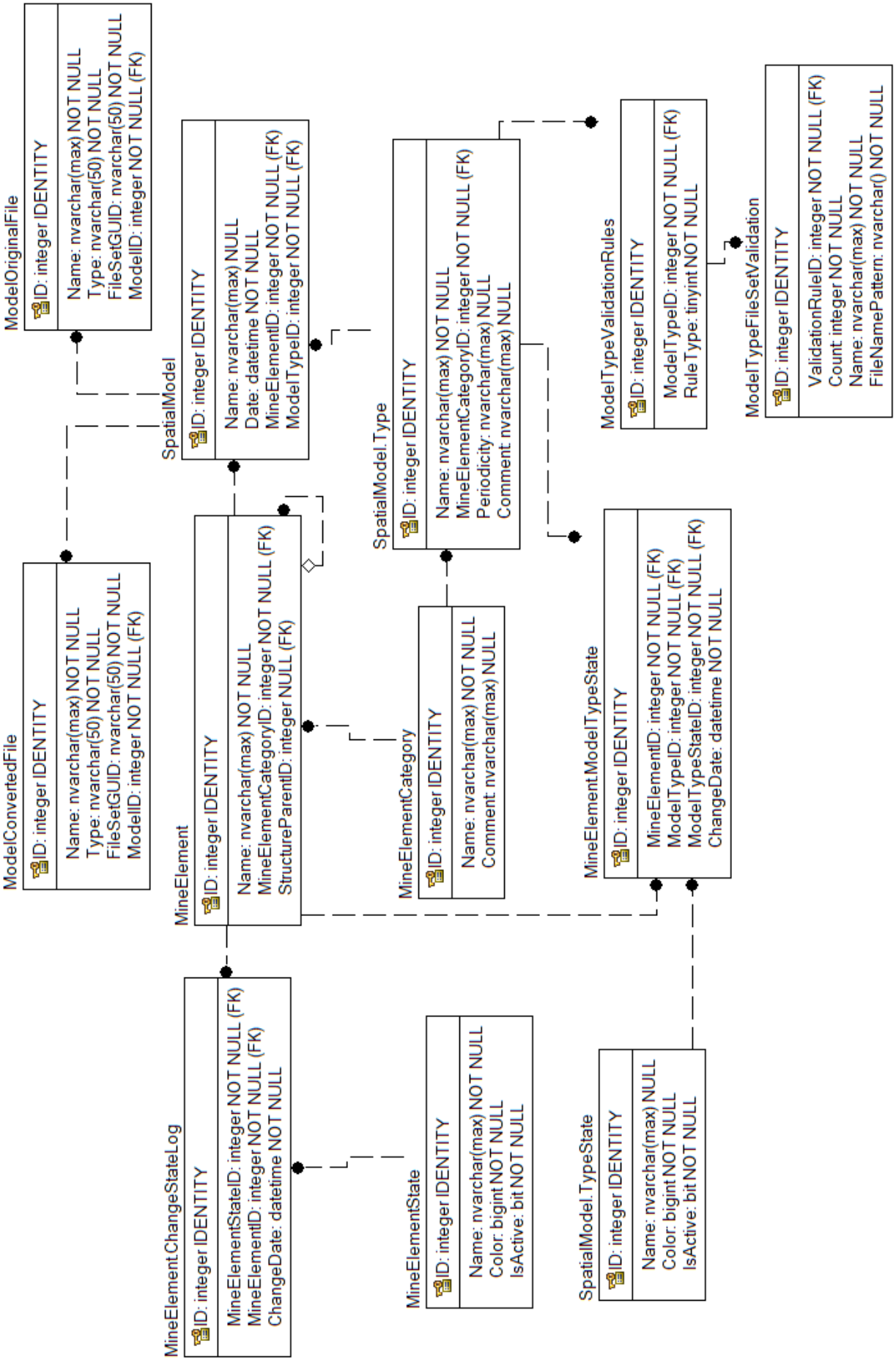


Рисунок 3.2 – Фізична модель бази даних

На основі фізичної моделі бази даних з використанням ERwin Data Modeler розроблено скрипт мовою T-SQL, який може бути використаний для створення нової або модифікації існуючої бази даних. Оскільки сутності MineElement та MineElementCategory вже присутні у базі даних Mine Advisor, їх створення було виключено зі скрипту.

Внесення змін до структури бази даних Mine Advisor виконується завдяки розробленому компанією Sight Power Inc допоміжному додатку RunCreateDBScripts.exe, інтерфейс якого наведений на рисунку 3.3.

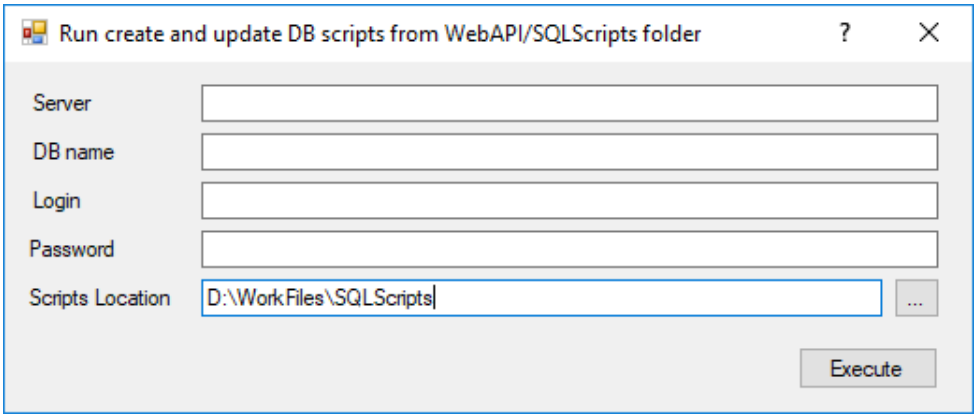
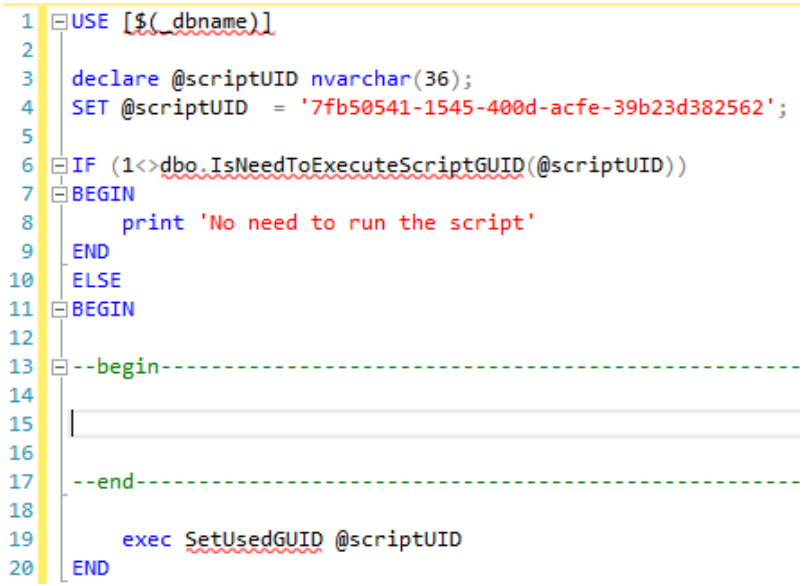


Рисунок 3.3 – інтерфейст допоміжного додатку RunCreateDBScripts.exe

Для роботи цей додаток потребує введення адреси MSSQL Server, на якому розташована цільова база даних, назви бази даних, логіну та паролю користувача та розташування папки, у якій розташовані файли скриптів, що необхідно виконати. Файли скриптів повинні мати формат .sql, а їх назви повинні починатися з порядкового номеру, наприклад: 001_firtsScript.sql.

Скрипт повинен створюватися за шаблоном, який наведено на рисунку 3.4.



```
1  USE [$(_dbname)]
2
3  declare @scriptUID nvarchar(36);
4  SET @scriptUID  = '7fb50541-1545-400d-acfe-39b23d382562';
5
6  IF (1<>dbo.IsNeedToExecuteScriptGUID(@scriptUID))
7  BEGIN
8      print 'No need to run the script'
9  END
10 ELSE
11 BEGIN
12
13 --begin----------------------------------------------------
14
15
16
17 --end------------------------------------------------------
18
19     exec SetUsedGUID @scriptUID
20 END
```

Рисунок 3.4 – Шаблон скрипта для модифікування бази даних Mine Advisor

У рядку 1 скрипт переключає контекст на базу даних, назва якої передається як параметр командної строки при виконанні скрипту у додатку RunCreateDBScripts.exe.

У рядках 3-4 створюється змінна @scriptUID, якій присвоюється значення GUID для цього скрипту.

У рядках 6-11 робиться перевірка на те, чи був цей скрипт виконаний для поточної бази даних. Для цього у функцію IsNeedToExecuteScriptGUID у якості параметру передається змінна @scriptUID. Якщо цей скрипт вже був виконаний для поточної бази даних, то на екран виводиться напис «No need to run the script» та його виконання завершується.

Текст скрипту, що повинен вносити зміни безпосередньо до структури бази даних, має починатися після коментаря «begin», що знаходиться у рядку 13, та завершуватися коментарем «end».

Після успішного виконання основної частини скрипту викликається функція SetUsedGUID з параметром @scriptUID, що вносить GUID поточного скрипту до переліку виконаних для поточної бази даних.

Такий шаблон дозволяє уникнути повторного виконання скрипту для однієї бази та дозволяє відстежувати версію бази даних за переліком виконаних скриптів. Перевірка версії бази даних проводиться під час запуску майстрів Mine

Advisor та не дозволяє продовжити роботу, якщо версія бази не відповідає версії додатку.

Таким чином, після успішного оновлення структури бази даних Mine Advisor, вона містить усі необхідні сутності, розглянуті у розділі 3.1.

3.3 Верифікація бази даних на основі формування запитів та аналіз результатів верифікації

Перевірка працездатності бази даних буде виконана на даних, що були надані компанією Sight Power Inc та є реальними даними золотодобувної компанії «Алтыналмас» (АО «АК Алтыналмас»). Кількість записів у основних таблицях наведено на рисунку 3.5.

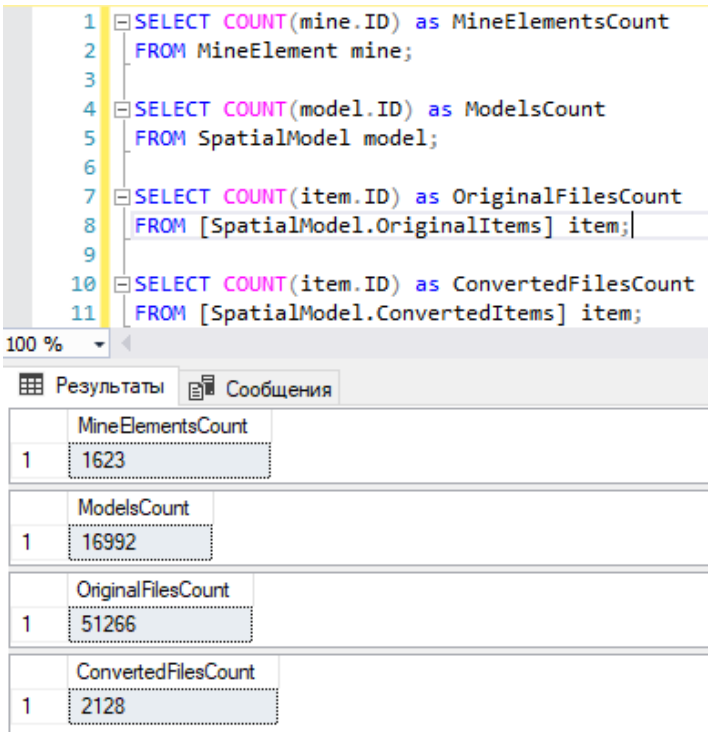
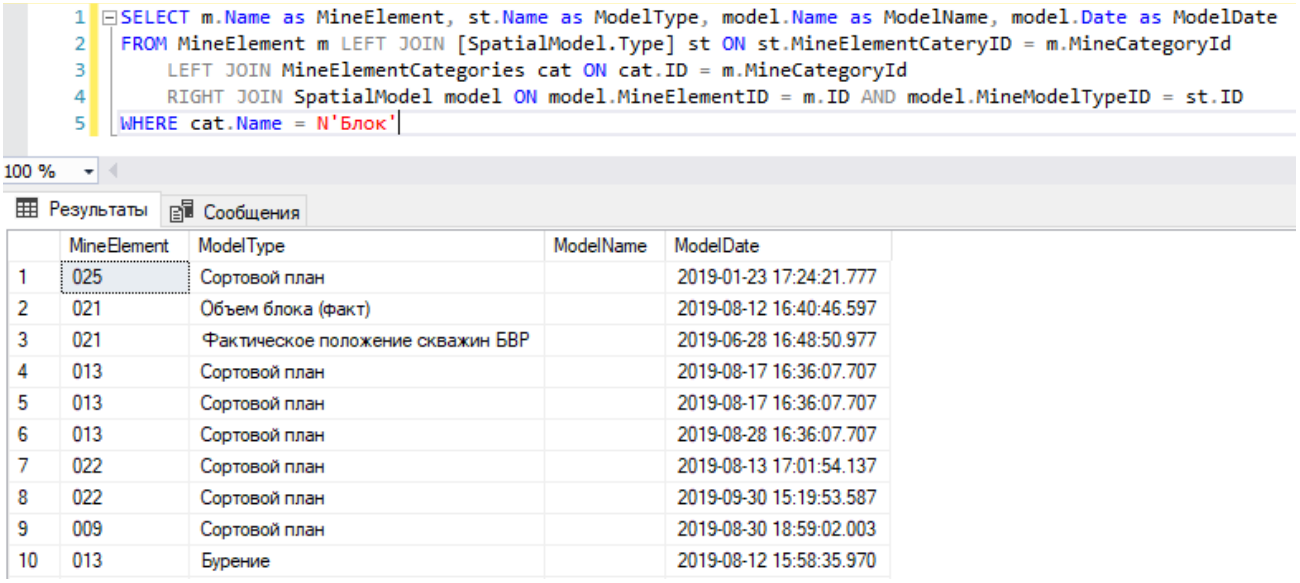


Рисунок 3.5 – Кількість записів у основних таблицях

Перш за все необхідно перевірити правильність створених взаємозв’язків між сутностями. Для цього буде використано запит на вилучення з бази даних назв структурних елементів, що мають категорію «Блок»; типів робіт, які

проводилися на цих структурних елементах; назв й дат звітів, що були складені за результатом виконання робіт. Текст запиту та частина результатів його виконання приведені на рисунку 3.6.



```
SELECT m.Name as MineElement, st.Name as ModelType, model.Name as ModelName, model.Date as ModelDate
 FROM MineElement m LEFT JOIN [SpatialModel.Type] st ON st.MineElementCateryID = m.MineCategoryId
     LEFT JOIN MineElementCategories cat ON cat.ID = m.MineCategoryId
     RIGHT JOIN SpatialModel model ON model.MineElementID = m.ID AND model.MineModelTypeID = st.ID
WHERE cat.Name = N'Блок'
```

100 %

Результаты | Сообщения

	MineElement	ModelType	ModelName	ModelDate
1	025	Сортовой план		2019-01-23 17:24:21.777
2	021	Объем блока (факт)		2019-08-12 16:40:46.597
3	021	Фактическое положение скважин БВР		2019-06-28 16:48:50.977
4	013	Сортовой план		2019-08-17 16:36:07.707
5	013	Сортовой план		2019-08-17 16:36:07.707
6	013	Сортовой план		2019-08-28 16:36:07.707
7	022	Сортовой план		2019-08-13 17:01:54.137
8	022	Сортовой план		2019-09-30 15:19:53.587
9	009	Сортовой план		2019-08-30 18:59:02.003
10	013	Бурение		2019-08-12 15:58:35.970

Рисунок 3.6 – Запит на вилучення з бази даниих інформації про звіти, що були складені для структурних елементів категорії «Блок» зі зазначенням типу робіт, для яких було складено звіти

Запит виконано успішно та його результати відповідають введеним до системи даним. Усього у результаті виконання запиту було повернено 1157 рядків, а середній час виконання запиту складає приблизно 200 мс (тут і далі замір часу проводився з використаннями Microsoft SQL Server Management Studio). Даний результат є прийнятним, бо не передбачається часте виконання таких складних запитів під час експлуатації системи.

Під час експлуатації розроблюваної системи передбачається багаторазове виконання запитів для відображення структури підприємства у інтерфейсі користувача.

Для вилучення з бази даних усіх комбінацій структурних елементів підприємства та типів проведених на них робіт використовується запит, приведений на рисунку 3.7.

```
SELECT m.ID as MineID, t.ID as TypeID, m.StructureParentID as ParentID
FROM MineElement m LEFT JOIN [SpatialModel.Type] t ON t.MineElementCateryID = m.MineCategoryId
```

100 %

Результаты | Сообщения

	MineID	TypeID	ParentID
1	1	3	NULL
2	1	7	NULL
3	1	8	NULL
4	1	9	NULL
5	1	10	NULL
6	1	13	NULL
7	1	43	NULL
8	2	1	1

Рисунок 3.7 – Запит на вилучення з бази даних усіх комбінацій структурних елементів та їх типів пробіт з зазначенням батьківстького структурного елементу

За результатами цього запиту можливо побудувати усю ієрархічну структуру підприємства та вказати, які типи робіт було проведено на кожному структурному елементі підприємства. За результатами цього запиту було повернено 7702 рядка, а середній час виконання складав 190 мс.

Також передбачається можливість виконувати фільтрацію структури підприємства за різноманітними ознаками, наприклад за категорією структурного елементу, назвою структурного елементу чи типом проведених робіт. Наприклад, необхідно вилучити з бази структурні елементи категорії «Блок» та відобразити лише типи робіт «Буріння» та «Вибух». Приклад запиту з фільтрацією наведено на рисунку 3.8.

```
SELECT c.ID as MineID, t.ID as TypeID, c.StructureParentID as ParentID
FROM MineElement c LEFT JOIN [SpatialModel.Type] t ON t.MineElementCateryID = c.MineCategoryId
    LEFT JOIN SpatialModel s ON s.MineModelTypeID = t.ID AND s.MineElementID = c.ID
    LEFT JOIN MineElementCategories cat ON cat.ID = c.MineCategoryId
    WHERE ( c.MineCategoryId IN(23) )
            AND ( ( s.MineModelTypeID IN (34,19) ) OR ( t.ID IS NULL ) OR ( t.ID IN (34,19) ) )
    GROUP BY c.ID, c.StructureParentID, t.ID;
```

Рисунок 3.8 – Запит на вилучення з бази даних комбінацій структурних елементів та типів робіт з урахуванням фільтрації

За результатом виконання запиту було повернено 244 рядки, а середній час виконання складав 190 мс. Результатом цього запиту є комбінації структурних

елементів та типів робіт, що мають бути відображені у структурі підприємства після фільтрації. Разом із результатами попереднього звіту це дозволить створити повну структуру підприємства та залишити у ній лише ті елементи, що необхідно відобразити.

Аналіз результатів виконання основних запитів вказує на те, що створена структура бази даних відповідає усім вимогам, наведеним у розділі 2.1 та враховує усі взаємозв’язки між структурними елементами підприємства, розглянутими у розділах 1.1-1.4.

Усі запити виконано успішно, а час їх виконання є прийнятним, адже процес експлуатації розроблюваної системи не передбачає виникнення великого навантаження на базу даних.

4 РОЗРОБКА ПІДСИСТЕМ ЗБЕРІГАННЯ ТА ОЦІНКИ ЯКОСТІ ЗВІТІВ. АНАЛІЗ РЕЗУЛЬТАТІВ

Розробка підсистем зберігання та оцінки якості звітів буде проводитися для інтеграції у програмне забезпечення Mine Advisor. Основною технологією розробки підсистем є .Net Framework версії 4.5.1, а саме наступні його складові:

– C# (C Sharp) – об’єктно-орієнтована мова програмування;

– Windows Presentation Foundation (WPF) – система для створення та відображення клієнтських додатків у платформі .Net[24];

– Language Integrated Query (LINQ) – мова інтегрованих запитів, що надає змогу писати структуровані та безпечні SQL-подібні запити до локальних колекцій об’єктів [25];

– LINQ to SQL – технологія для перетворення запитів мовою LINQ на запити мовою SQL [26];

– Entity Framework – засіб об’єктно-реляційного відображення (O/RM, Object-Relational Mapper). Дозволяє створювати звичайні C# класи, що є відображенням сутностей у базі даних, створювати об’єкти доступу до бази даних та підтримувати цілісність даних [27];

– Telerik UI for WPF – набір компонентів та інструментів для створення інтерфейсу користувача, що розширюють можливості стандартних елементів WPF [28].

Даний перелік технологій дозволяє створювати програмні продукти будь-якої складності, основним призначенням яких є надання користувачеві інтерфейсу доступу до бази даних та інструментів вилучення з неї корисної інформації.

4.1 Розробка підсистеми зберігання звітів про виконання завдань

Підсистема зберігання звітів складатиметься з трьох складових:

– інструменту керування простими сутностями бази даних. До простих сутностей було віднесено статуси типів моделей, статуси структурних елементів, категорії структурних елементів;

– інструменту для керування структурою підприємства, що дозволить додавати, видаляти та модифікувати елементи структури підприємства;

– інструменту для завантаження до системи звітів про виконання завдань, що дозволить додавати, видаляти та модифікувати моделі, додавати до них оригінальні та конвертовані файли.

Усі інструменти, що використовуються у Mine Advisor складаються з наступних компонентів:

– майстра (wizard), який реєструється у системі Mine Advisor та відповідає за відображення відповідної кнопки у меню навігації, запуск, регулювання й завершення виконання інструменту та за необхідності запускають фоновий процес розрахунків;

– сторінок (page), що є складовими майстру. Вони створюються за паттерном Model-View-View Model (MVVM)[24] та об’єднують у собі інтерфейс користувача(View), логіку обробки введених користувачем даних (View Model), та логіку перевірки результатів роботи (Model).

Під час запуску інструменту майстер створює необхідні сторінки та організує їх у необхідному порядку, створюючи таким чином маршрут (route). Маршрут може буди як строго послідовним (робота наступної сторінки базується на результатах роботи поточної) так і більш складним (може розгалужуватись за необхідністю).

Наведені вище складові підсистеми зберігання звітів про виконання завдань будуть додані у вигляді сторінок до вже існуючого у Mine Advisor інструменту «Менеджер цифрового рудника»(Digital Mine Manager).

4.1.1 Розробка сторінки керування простими сутностями бази даних

Сторінка керування простими сутностями також вже присутня у системі Mine Advisor. Завдяки цій сторінці можливо модифікувати класи, що є відображенням сутностей бази даних та реалізують інтерфейс IAuxRecord:

```
public interface IAuxRecord : System.ComponentModel.INotifyPropertyChanged
                        ,
SightPower.XGIP.Plugins.Database.Common.IIdentified
                        , SightPower.XGIP.Plugins.Common.Components.INamed
{
     IEnumerable<Object> GetUserRecords();
     int GetUserRecordsCount();
}
```

Класи, що реалізують цей інтерфейс повинні мати: властивість Name типу string, доступну для запису та читання (об’явлена у інтерфейсі INamed); властивість ID типу int, доступну для читання (об’явлена у інтерфейсі IIdentified); реалізацію функції GetUserRecords, що має повертати перелік усіх об’єктів, пов’язаних з поточним об’єктом, що також є відображенням сутностей бази даних та через які неможливо видалити з бази даних сутність, що відображена на поточний об’єкт; реалізацію функції GetUserRecordsCount, що повертає кількість описаних вище об’єктів; подію PropertyChanged типу PropertyChangedEventHandler, що сигналізує про зміну значень властивостей об'єкту (об’явлена у інтерфейсі INotifyPropertyChanged).

Завдяки реалізації цього інтерфейсу, View Model сторінки керування простими сутностями бази даних може запобігти виникненню помилки під час видалення сутності, а View сторінки може без змін прив’язок даних взаємодіяти з об’єктами, що редагуються на даний момент.

Під час розробки системи було вирішено обмежити кількість можливих станів типів моделей та структурних елементів підприємства до десяти, враховуючи стан «Не задано». Така кількість станів є достатньою для вимог підприємства та не створить непорозумінь. Перевірку кількості станів робитиме View Model сторінки.

Стан «Не задано» було вирішено не зберігати у базі даних та зробити його доступним тільки для читання. Це дасть змогу однозначно визначати, чи задано стан відповідного елементу.

Також необхідно додати перевірку на унікальність назв станів та їх кольорів для уникнення плутанини.

З урахуванням наведеного вище, до існуючої сторінки керування простими сутностями бази даних та класів MineElementState та SpatialModel_TypeState необхідно внести наступні зміни:

– додати класи MineElementState та SpatialModel_TypeState до переліку доступних для редагування класів сторінки;

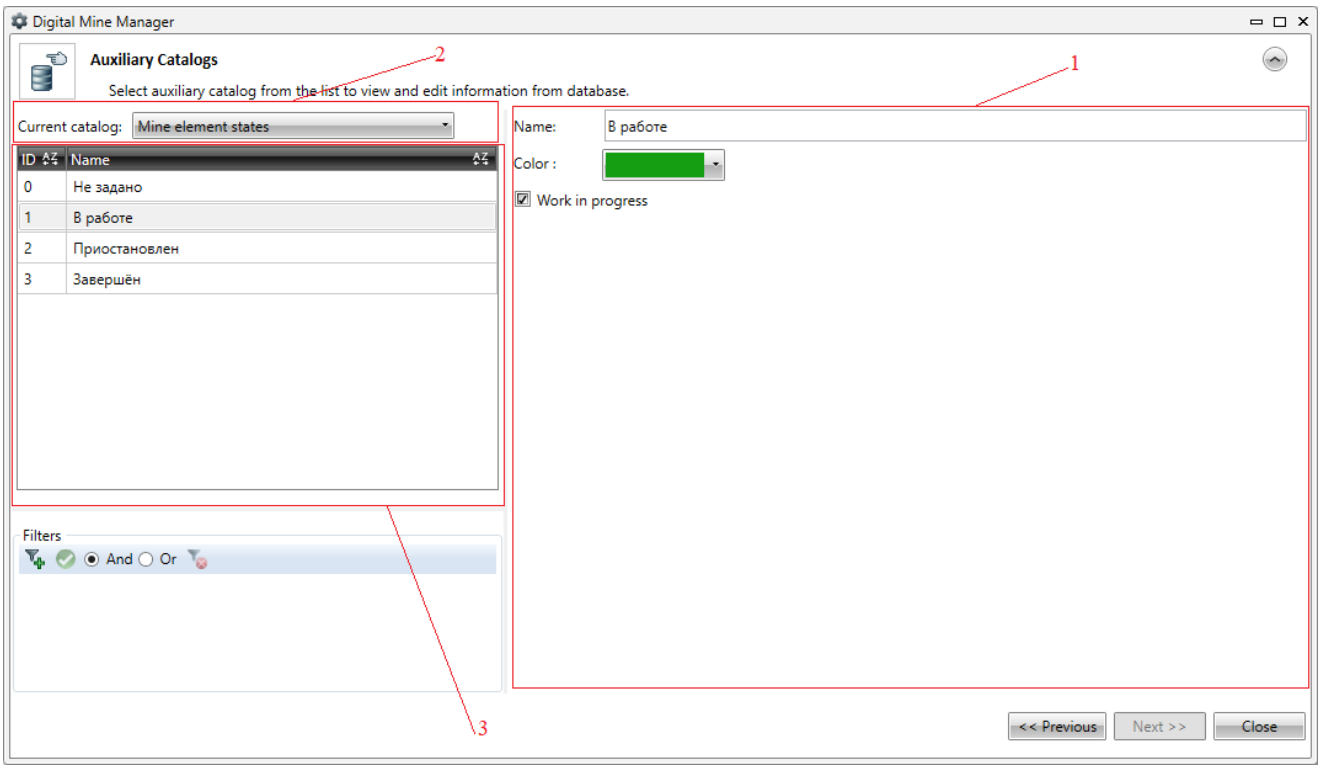
– розробити інтерфейс для редагування об’єктів класів MineElementState та SpatialModel_TypeState у View сторінки;

– додати необхідну перевірку правильності введених даних для об’єктів класів MineElementState та SpatialModel_TypeState у Model сторінки;

– модифікувати класи MineElementState та SpatialModel_TypeState відповідно до вимог сторінки (реалізувати інтерфейс IAuxRecord);

– модифікувати усі складові сторінки відповідно до особливостей роботи з класами MineElementState та SpatialModel_TypeState.

Інтерфейс сторінки керування простими сутностями бази даних з редактором сутності стану структурного елементу підприємства наведено на рисунку 4.1.



1 – редактор обраної сутності, 2 – випажаючий список з переліком доступних типів сутностей для редагування, 3 – перелік сутностей обраного типу

Рисунок 4.1 – Інтефейс сторінки керування простими сутностями бази даних

Для визначення переліку доступних для редагування типів сутностей Model сторінки має масив AuxWorkingTypes типу Type[]. Значення цього масиву задається в коді та є незмінним під час роботи з системою. Таким чином необхідно лише додати необхідні типи до визначення цього масиву.

Інтерфейс редагування сутностей станів є однаковим для MineElementState та SpatialModel_TypeState. Для зміни інтерфейсу редагування відповідно до обраного типу сутності використовуються шаблони (Data Template), що задають інтерфейс редактору, та тригери стилів (Style.Triggets, DataTrigger), що виставляють необхідний шаблон.

У ході розробки було вирішено задати палітру доступних кольорів для станів структурних елементів підприємства та типів моделей, щоб усі кольори були контрастними та було легко відрізнити один стан від іншого. Також це полегшить процес перевірки унікальності кольорів станів. Для зберігання обраних

кольорів було створено файл XML(eXtensible Markup Language). Зміст файлу зображено на рисунку 4.2.

```
<?xml version="1.0" encoding="utf-8" ?>
<MineElementAndModelTypeStateColors>
  <Colors xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <!--all values are writen in hex notation without '#' sumbol-->
    <!--'ff' is added infront of all values to represent A(opasity)-->
    <Color>FFD3980A</Color>
    <Color>ffeae114</Color>
    <Color>ff169E13</Color>
    <Color>ff1ec9e8</Color>
    <Color>ff842eba</Color>
    <Color>fff375f3</Color>
    <Color>ff17f1d8</Color>
    <Color>ffa6e541</Color>
    <Color>ffb40a1b</Color>
    <Color>ff000000</Color>
  </Colors>
</MineElementAndModelTypeStateColors>
```

Рисунок 4.2 – XML-файл з кольорами станів структурних елементів підприємства та типів моделей

Для роботи с цим файлом було створено клас, який має необхідні атрибути для зчитування XML у його поля:

```
public class MineElementAndModelTypeStateColors
{
    [XmlArray]
    [XmlArrayItem("Color")]
    public List<string> Colors { get; set; }
}
```

Такий підхід до зберігання кольорів полегшує процес додавання чи зміни кольорів та робить можливим використання однакового набору кольорів у різних частинах коду.

Перелік доступних кольорів для стану оновлюється при кожній зміні обраного для редагування елемента та при зміні його власного кольору. З переліку усіх кольорів виключаються ті, що вже назначені для інших станів, завдяки чому стани будуть мати унікальний колір.

Також було вирішено створити три стани за замовчуванням: «В работе»(активний стан), «Приостановлен»(не активний стан) та «Завершён»(не

активний стан). Для цього було створено новий скрипт для бази даних, що заносить ці данні до неї.

Стан «Завершён», як і стан «Не задано», було вирішено зробити доступним тільки для читання. Це заборонить користувачеві видалити чи змінити його та дасть змогу не враховувати структурні елементи підприємства та типи моделей, робота на яких вже завершена. Для цього у класах MineElementState та SpatialModel_TypeState була створена властивість IsReadOnly та визначено частковий (partial) метод OnLoaded(), що визивається коли сутність була завантажена з бази даних до об'єкту відповідного класу. Код цих доповнень є аналогічним для обох класів станів та виглядає наступним чином:

```
partial void OnLoaded()
{
    if (this.Name == "Завершён")
        this.IsReadOnly = true;
}
private bool isReadOnly = false;
public bool IsReadOnly
{
    get { return this.isReadOnly; }
    set
    {
        if(this.isReadOnly != value) {
            this.SendPropertyChanging();
            this.isReadOnly = value;
            this.SendPropertyChanged("IsReadOnly");
        }
    }
}
```

Перевірка стану «Завершён» лише за ім'ям є робочою, бо цей стан було додано до бази даних за допомогою скрипту та буде додано перевірку на унікальність назв станів.

Стан «Не задано» створюється під час роботи системи та, як було сказано вище, не зберігається у базі даних і є доступним тільки для читання. Для цього у класах MineElementState та SpatialModel_TypeState було створено статичні поля, що відповідають за назву та колір стану за замовчуванням, а також створюють

об’єкт класу стану за замовчуванням. Поля для класу MineElementState наведено у лістингу 4.1. Поля класу SpatialModel_TypeState є аналогічними:

Лістинг 4.1 – Статичні поля класу MineElementState для створення стану «Не задано»

```
public static string DefaultName = "Не задано";
public static long DefaultColor = 4287203721;
public static long FinishedStateColor = 4278190080;
public static MineElementState DefaultState = new MineElementState()
{
    Color = DefaultColor,
    Name = DefaultName,
    IsActive = false,
    IsReadOnly = true
};
```

Для додавання стану за замовченням в перелік доступних для редагування об’єктів було перевизначено(override) функцію GetObjectsCollection у Model сторінки та створено клас ObservableCollectionToTableWithDefaultItem<T,T>, що дає змогу за рахунок зміни простої колекції об’єктів автоматично переносити ці зміни до контекстну даних (DataContext), який підключено до цільової бази даних. Для заборони видалення та модифікування об’єктів, що доступні тільки для читання, було перевизначено функцію CanDeleteElemet у Model сторінки та модифіковано XAML-розмітку View сторінки.

Усі описані вище зміни дають змогу зчитувати перелік станів з бази даних, модифікувати їх, додавати чи видаляти та зберігати зміни у базі даних.

Останнім кроком є додавання перевірки правильності введених даних. Для цього необхідно додати до функції IsSubmitAllowed, що викликається безпосередньо перед зберіганням змін до бази даних та перевіряє правильність введених користувачем даних, у Model сторінки рядки, наведені на рисунку 4.3:

```
287 if (res && typeof(MineElementState) == this.TypedVM.SelectedType)
288 {
289     if (this.LocalContext.ActualData<MineElementState>().GroupBy(x => x.Name.Trim()).Select(x => x.Count()).Any(x => x > 1))
290     {
291         MessageHelper.Current.ShowSimpleDialogLocalized("ATP_uniqueMineElementStateNames", "DM_AuxTablePageCaption"
292                                                         , MessageDialogButtons.Ok, MessageLevel.Warning);
293         res = false;
294     }
295 }
296 if (res && typeof(SpatialModel_TypeState) == this.TypedVM.SelectedType)
297 {
298     if (this.LocalContext.ActualData<SpatialModel_TypeState>().GroupBy(x => x.Name.Trim()).Select(x => x.Count()).Any(x => x > 1))
299     {
300         MessageHelper.Current.ShowSimpleDialogLocalized("ATP_uniqueModelTypeStateNames", "DM_AuxTablePageCaption"
301                                                         , MessageDialogButtons.Ok, MessageLevel.Warning);
302         res = false;
303     }
304 }
305 if (res && typeof(MineElementState) == this.TypedVM.SelectedType)
306 {
307     if (this.LocalContext.ActualData<MineElementState>().Count() > 9)
308     {
309         MessageHelper.Current.ShowSimpleDialogLocalized("ATP_tooMuchMineElementStatesWithDelete", "DM_AuxTablePageCaption"
310                                                         , MessageDialogButtons.Ok, MessageLevel.Warning);
311         res = false;
312     }
313 }
314 if (res && typeof(SpatialModel_TypeState) == this.TypedVM.SelectedType)
315 {
316     if (this.LocalContext.ActualData<SpatialModel_TypeState>().Count() > 9)
317     {
318         MessageHelper.Current.ShowSimpleDialogLocalized("ATP_tooMuchModelTypeStatesWithDelete", "DM_AuxTablePageCaption"
319                                                         , MessageDialogButtons.Ok, MessageLevel.Warning);
320         res = false;
321     }
322 }
```

Рисунок 4.3 – Перевірка правильності введених даних для станів типів моделей та структурних елементів підприємства

На рисунку 4.3 у рядках 287-304 виконується перевірка на унікальність назв станів, а у рядках 305-322 виконується перевірка на кількість станів.

В усіх перевірках використовується функція ActualData<T>. Ця функція створена для вилучення з бази даних та поточного контексту даних усіх сутностей заданого типу, враховуючи ще не збережені видалення та додавання. Використання її під час перевірки правильності введених даних дозволяє запобігти конфліктам, що виникають під час одночасної роботи декількох користувачів з однією базою даних.

Також для запобігання цих конфліктів та збереження обмеження на максимальну кількість станів аналогічна перевірка робиться перед додаванням нового стану.

Після проведення усіх вищеназваних змін сторінка керування простими сутностями бази даних є придатною для роботи з сутностями станів структурних елементів підприємства та типів моделей.

4.1.2 Розробка сторінки керування структурними елементами підприємства

Mine Advisor вже містить у собі сторінку керування структурними елементами підприємства, тому її потрібно лише модифікувати відповідно до нових вимог:

– додати відображення кольору поточного стану структурного елементу підприємства;

– додати можливість змінювати поточний стан структурного елементу підприємства;

– додати можливість редагувати історію змін станів структурного елементу підприємства.

Структура підприємства на сторінці керування структурою підприємства відображується у вигляді дерева з використанням компоненту MineAdvisor CustomTreeView. Приклад відображення структури підприємства з відображенням кольору поточного стану елементів структури наведено на рисунку 4.1.

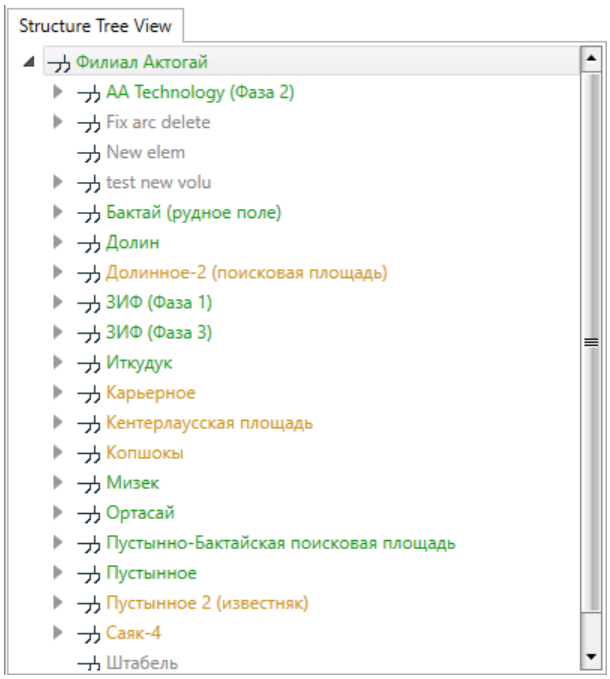


Рисунок 4.4 – Дерево структури підприємства з відображенням кольору поточного стану

Для відображення кольору поточного стану структурного елементу у дереві було створено спеціальний інтерфейс:

```
public interface IColoredTreeViewItem { long TreeViewColor { get; } }
```

До автоматично створеного за допомогою Entity Framework класу MineElement було додано реалізацію цього інтерфейсу, та додаткові функції, що повертають поточний стан структурного елементу на сьогоднішній день та на обрану дату:

```
public MineElementState GetActualState()
{
    return this.GetActualStateForDate(DateTime.Now);
}

public MineElementState GetActualStateForDate(DateTime date)
{
    MineElement_ChangeStateLog res =
    this.MineElement_ChangeStateLogs.OrderByDescending(x =>
x.ChangeDate)
    .FirstOrDefault(x => x.ChangeDate.Date <= date.Date);
    return res == null ||  res.MineElementState == null ?
    MineElementState.DefaultState : res.MineElementState;
}

public long TreeViewColor
{
    get { return this.GetActualState().Color; }
}
```

Для визначення кольору та поточного стану структурних елементів, стан яких ще не було задано, використовуються статичні поля класу MineElementState, наведені у лістингу 4.1.

Усе наведене вище дає змогу однозначно визначити поточний стан структурного елементу та його колір у вигляді 64-ох бітного цілого числа(long).

Безпосереднє відображення структури підприємства відбувається завдяки компоненту Mine Advisor – SimpleTreeView. Цей компонент є надбудовою над стандартним елементом Telerik UI for WPF – RadTreeView. Для відображення

кольору елементу було змінено шаблон відображення елементу дерева – RadTreeViewItem. Для цього було модифіковано XAML-документ (eXtensible Application Markup Language) розмітки SimpleTreeView, а саме додано наступні рядки до стилю елементу TextBlock, що відображає ім’я поточного елементу дерева:

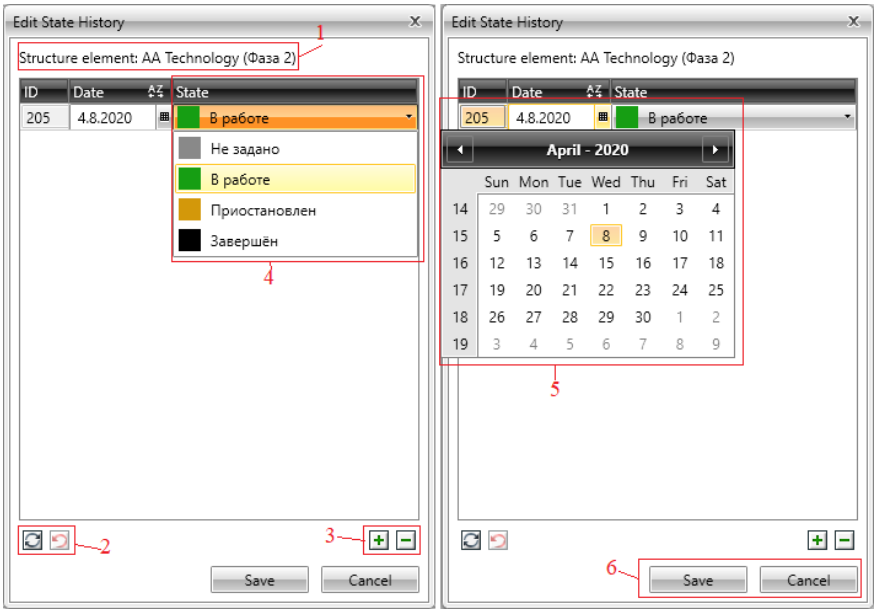
```
<Setter Property="Foreground" Value="{Binding Path=TreeViewColor,
Mode=OneWay, Converter={StaticResource xIntToSolidBrushConverter}}"/>
```

У цих рядках для властивості (Property) Foreground (колір шрифту) виставляється значення у вигляді прив’язки даних (data binding) до властивості TreeViewColor, що дозволяє за допомогою внутрішніх засобів WPF автоматично обновляти значення властивості Foreground при зміні властивості TreeViewColor. Також у цій прив’язці даних задається конвертер(Converter), що перетворює значення типу long на значення типу System.Windows.Media.SolidColorBrush з відповідним кольором.

Усе це дозволяє автоматично оновлювати колір елемента у дереві структури підприємства при зміні поточного стану елементу структури підприємства.

Наступним кроком було розроблено компонент редагування історії змін станів структурного елементу підприємства. Він має відображатися у вигляді діалогового вікна.

Для можливості відображення у вигляді діалогового вікна клас компоненту було успадковано від класу Telerik.Windows.Controls.RadWindow. Цей клас успадковує клас WindowBase, що є базовим класом для вікон у WPF, та розширює його функціонал. Інтерфейс компоненту зображено на рисунку 4.5.



1 – назва структурного елементу, історія зміни статусів якого редагується, 2 – кнопки «оновити» та «відмінити», 3 – кнопки «додати» та «виділити», 4 – випадаючий список (ComboBox) з усіма можливими станами, 5 – календар (DatePicker) для вибору дати зміни стану, 6 – кнопки «зберегти» та «відмінити»

Рисунок 4.5 – Інтерфейс вікна редагування історії змін станів структурних елементів

Компонент також розробляється за паттерном MVVM, де Model це сторінка майстру або інший інструмент, що буде створювати та викликати його. Оскільки можливість редагування історії статусів необхідна й для типів робіт(типів моделей), то для View Model було створено універсальний інтерфейс:

```
    public interface IEditStateHistoryDialogVM: IDisposable
    {
        bool OnDialogOk();
        bool OnDialogCancel();
        void SortStates(Telerik.Windows.Controls.SortingState
sortingState);

        ICommand AddItemCommand { get; }
        ICommand DeleteItemCommand{ get; }
        ICommand DiscardChangesCommand{ get; }
        ICommand RefreshCommand { get; }

        IEnumerable<StateHistoryItemWrap> ChangeHistory { get; }
        StateHistoryItemWrap SelectedHistoryItem { get; set; }

        string Header { get; }
```

```
        bool NeedToRefresh { get; }
    }
```

Реалізація цього інтерфейсу для створення View Model необхідна, бо прив’язки даних, що визначені у XAML, використовують саме ті властивості, що зазначено у інтерфейсі, а код програмної частини (code-behind), використовує визначені у інтерфейсі методи.

Для уніфікації роботи безпосередньо із сутністю поточного стану було створено інтерфейс IStateChangeDescriber:

```
    public interface IStateChangeDescriber : IIdentified
    {
        DateTime ChangeDate { get; set; }
        object NewState { get; set; }
    }
```

Цей інтерфейс дозволяє однаково працювати як зі станами структурних елементів підприємства, так і зі станами типів моделей. Реалізація цього інтерфейсу для класів MineElement_ChangeStateLog та MineElement_ModelTypeState є аналогічною: властивість NewState є додатковим етапом у присвоєнні значення до відповідної властивості класу.

Також слід зазначити, що стан структурного елементу підприємства чи стан типу моделей для конкретного структурного елементу підприємства може змінюватися лише одного разу на добу та не може бути змінено на той самий стан. Для перевірки цього у класах MineElement_ModelTypeState та MineElement_ChangeStateLog було створено функції, що обробляють масив змін станів відповідних елементів та присвоюють властивостям, що відповідають за взаємозв’язок між сутностями бази даних, помилкових елементів значення null. Далі наведено приклад такої функції у класі MineElement_ChangeStateLog, функція у класі MineElement_ModelTypeState є аналогічною:

```
public static void RemoveExtraRecords(MineElement_ChangeStateLog[]
                                                        statesToProcess)
{
    if (statesToProcess == null || statesToProcess.Length == 1) {
        return;
```

```
    }
    statesToProcess = statesToProcess.OrderBy(x => x.ChangeDate).ToArray();
    MineElement_ChangeStateLog currentState =

      statesToProcess.FirstOrDefault();
    for (int i = 1; i < statesToProcess.Length; i++) {
       if (statesToProcess[i].NewMineElementStateID ==
           currentState.NewMineElementStateID) {
          statesToProcess[i].MineElement = null;
           continue;
       }
       else if (currentState.ChangeDate.Date ==
                 statesToProcess[i].ChangeDate.Date) {
          currentState.MineElementState
=statesToProcess[i].MineElementState;
          statesToProcess[i].MineElement = null;
          MineElement_ChangeStateLog previousState =
                      statesToProcess.LastOrDefault(x => x.MineElement !=
                      null && x.ChangeDate < currentState.ChangeDate);
         if (previousState != null && previousState !=
                currentState && currentState.NewMineElementStateID ==
                previousState.NewMineElementStateID) {
               currentState.MineElement = null;
               currentState = previousState;
           }
           continue;
       }

        currentState = statesToProcess[i];
    }
}
```

Використання цієї функції після зміни історії станів структурного елементу підприємства або типу моделі дозволяє уникнути описаних вище проблем та полегшує подальший аналіз даних.

Для зміни поточного стану структурного елементу підприємства на сторінку керування структурою підприємства було додано спеціальний випадаючий список, зображений на рисунку 4.6.

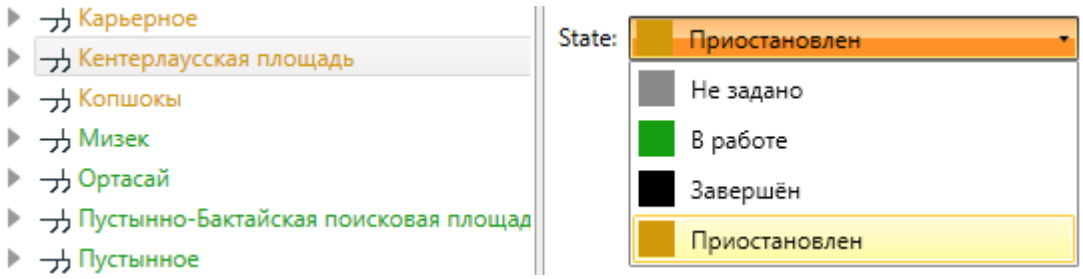


Рисунок 4.6 – Зміна поточного стану структурного елементу підприємства

Для правильної обробки зміни поточного стану у View Model сторінки було додано спеціальну властивість, що бобробляє зміну поточного стану структурного елементу та за необхідністю створює новий об’єкт класу MineElement_ChangeStateLog.

Після внесення наведених вище змін, сторінка керування структурою підприємства є завершеною та може використовуватися у розроблюваній системі.

4.1.3 Розробка сторінки завантаження до системи звітів про виконання завдань

Сторінка завантаження до системи звітів про виконання завдань також буде додана до майстру «Менеджер цифрового рудника»(Digital Mine Manager).

У цій сторінці мають бути реалізовані наступні функції:

– додавання, видалення та модифікування (зміна назви та дати, видалення файлів) звіту;

– редагування історії зміни станів типу робіт;

– додавання до звіту оригінальних та конвертованих файлів з переліку вже доданих до системи Mine Advisor;

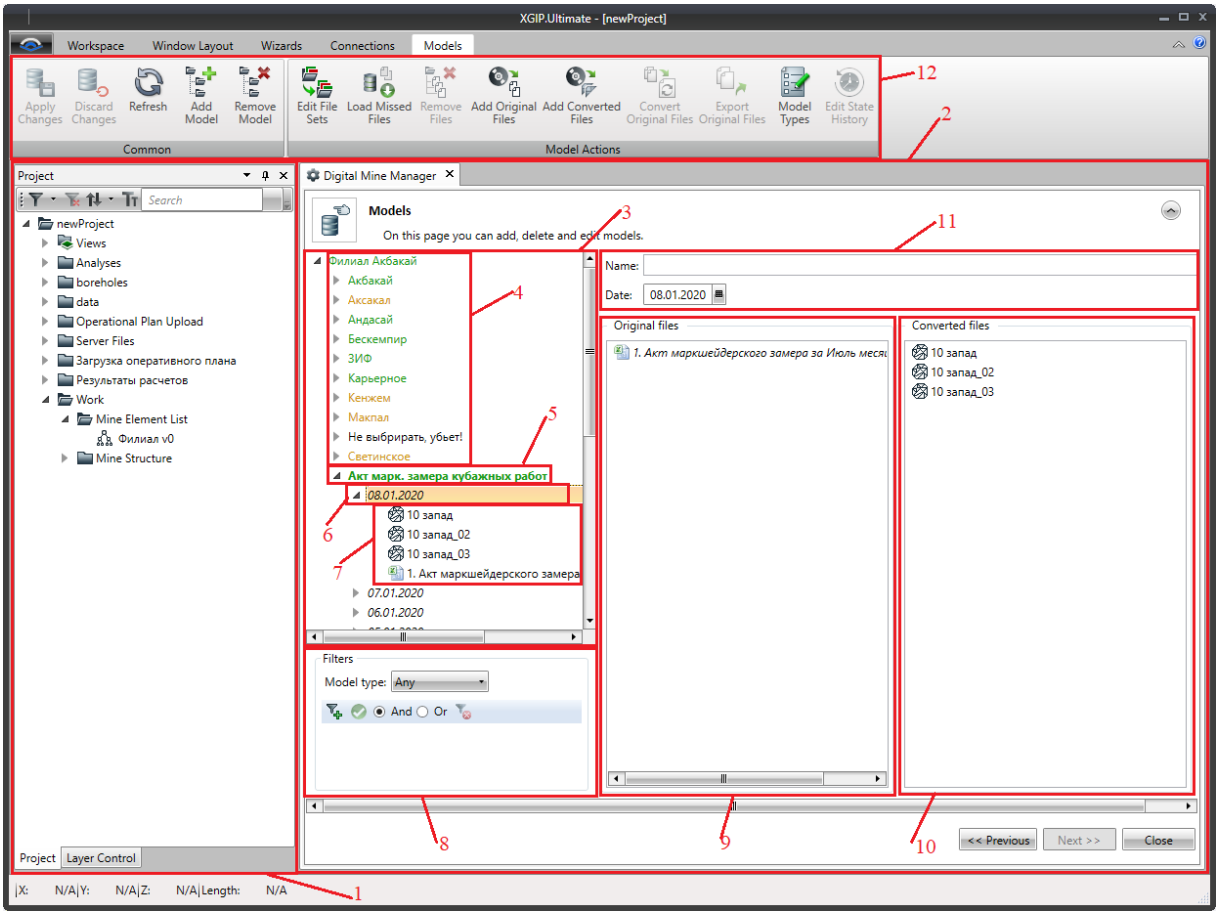
– додавання оригінальних та конвертованих файлів до звіту безпосередньо з файлової системи.

Для можливості додавання звіту до необхідного структурного елементу та типу робіт, необхідно відображати структуру підприємства аналогічно зображеній на рисунку 4.4, та додати до неї відображення типів робіт, звітів та файлів звітів. Усе це буде зроблено у деревовидній структурі, приклад якої зображено на

рисунку 4.7. У ході розробки сторінки завантаження звітів було вирішено створити новий компонент для відображення структури підприємства з усіма необхідними елементами, що буде підтримувати можливість фільтрації елементів за типом робіт та назвою звітів та буде швидшим за компонент CustomTreeView, який використовується на сторінці керування структурою підприємства.

Також розробка окремого компоненту є доцільною, бо передбачається створення нових інструментів, у яких цей компонент буде використано.

Надалі у роботі цей компонент буде називатися «дерево звітів».



1 – дерево проекту Mine Advisor; 2 – сторінка завантаження звітів до системи; 3 – дерево структури підприємства(дерево звітів); 4 – структурні елементи підприємства; 5 – тип робіт, що відноситься до структурного елементу «Филиал Акбакай»(виділено жирним); 6 – звіт про виконнаня робіт(виділен курсивом); 7 – файли звіту; 8 – панель фльтрів; 9 – перелік ориганальних файлів обраного для редагування звіту; 10 – перелік конвертованих файлів обраного для редагування звіту; 11 – поля редагування назви та дати звіту; 12 – панель інструментів (ribbon) сторінки завантаження звітів

Рисунок 4.7 – Інтерфейс сторінки завантаження до системи звітів про виконання завдань

Для забезпечення швидкості роботи дерева звіту було вирішено реалізувати два підходи:

– асинхронний доступ до бази даних для можливості паралельного завантаження декількох гілок дерева не у головному потоці додатку;

– DTO (Data Transfer Object) – шаблон проектування при якому класичні об’єкти доступу до даних (об’єкти класів, що були автоматично створені за допомогою Entity Framework та доступ до яких відбувається за допомогою Data Context) замінюються на простіші об’єкти, що мають лише данні та позбавлені будь-якої логіки [29].

Для асинхронного доступу до бази даних було розроблено спеціальну інфраструктуру, яку зображено на рисунку 4.8.

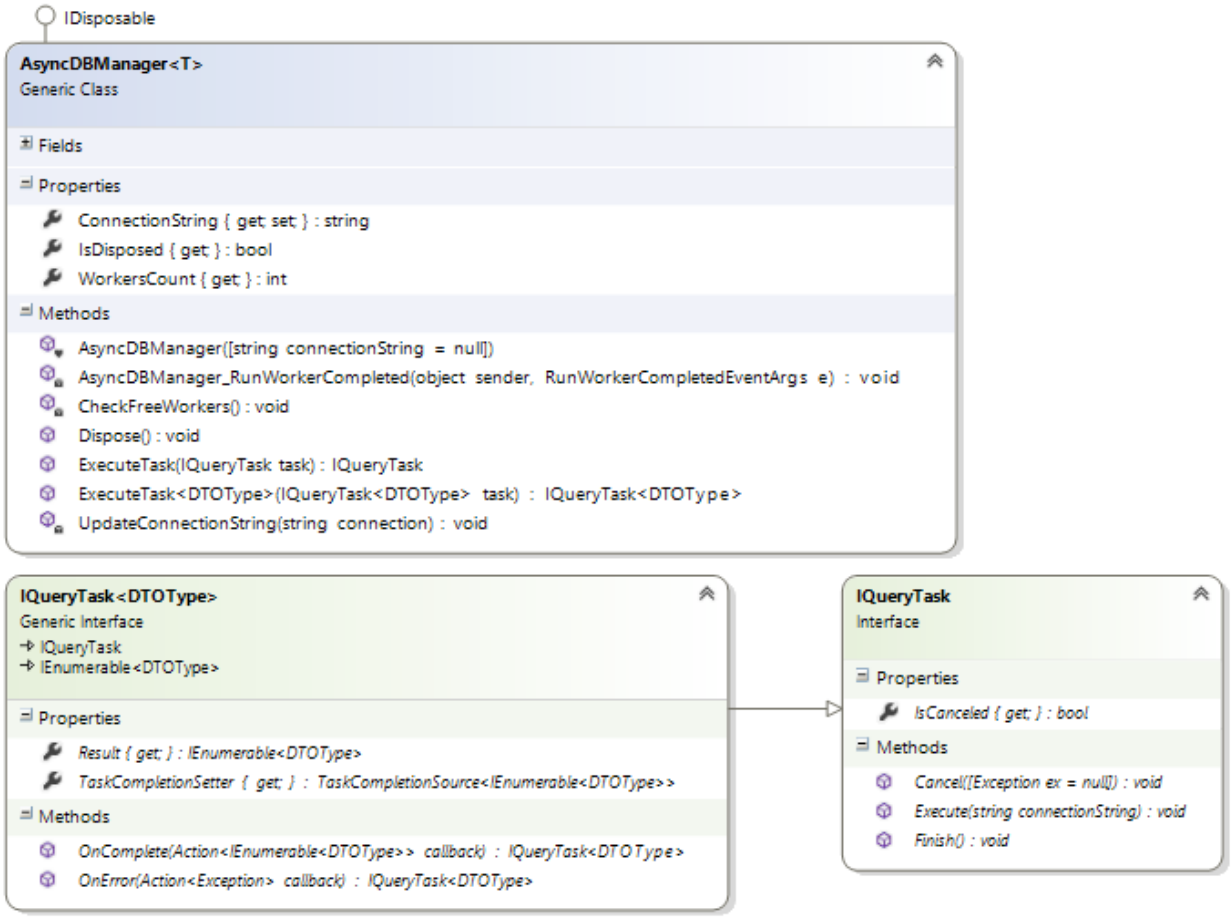


Рисунок 4.8 – Інфраструктура асинхронного доступу до бази даних

Вона складається з керуючого класу (AsyncDBManager) та класів задач (повинні реалізувати інтерфейс IQueryTask). AsyncDBManager містить у собі чергу (Queue) об’єктів типу IQueryTask, які необхідно виконати. Для паралельного виконання використовується клас AsyncDBWorker, що успадковує клас System.ComponentModel. BackgroundWorker. Цей клас дозволяє у окремому потоці виконати необхідну роботу.

Кількість AsyncDBWorker, що використовується у AsyncDBManager, на одиницю менше за кількість логічних ядер процесору, але не менше одного. Це дозволяє максимально ефективне паралельне виконання задач, яке не буде перешкоджати роботі головного потоку користувацького інтерфейсу.

Для зручності роботи з IQueryTask<DTOType> були розроблені спеціальні статичні методи розширення у статичному класі QueryTaskHelper, що дозволяють застосовувати оператор await[25] до об’єктів типів IQueryTask<DTOType> [30].

Основні властивості та сигнатури методів класу AsyncDBWorker та сигнатури методів розширення IQueryTask<DTOType> зображено на рисунку 4.9.

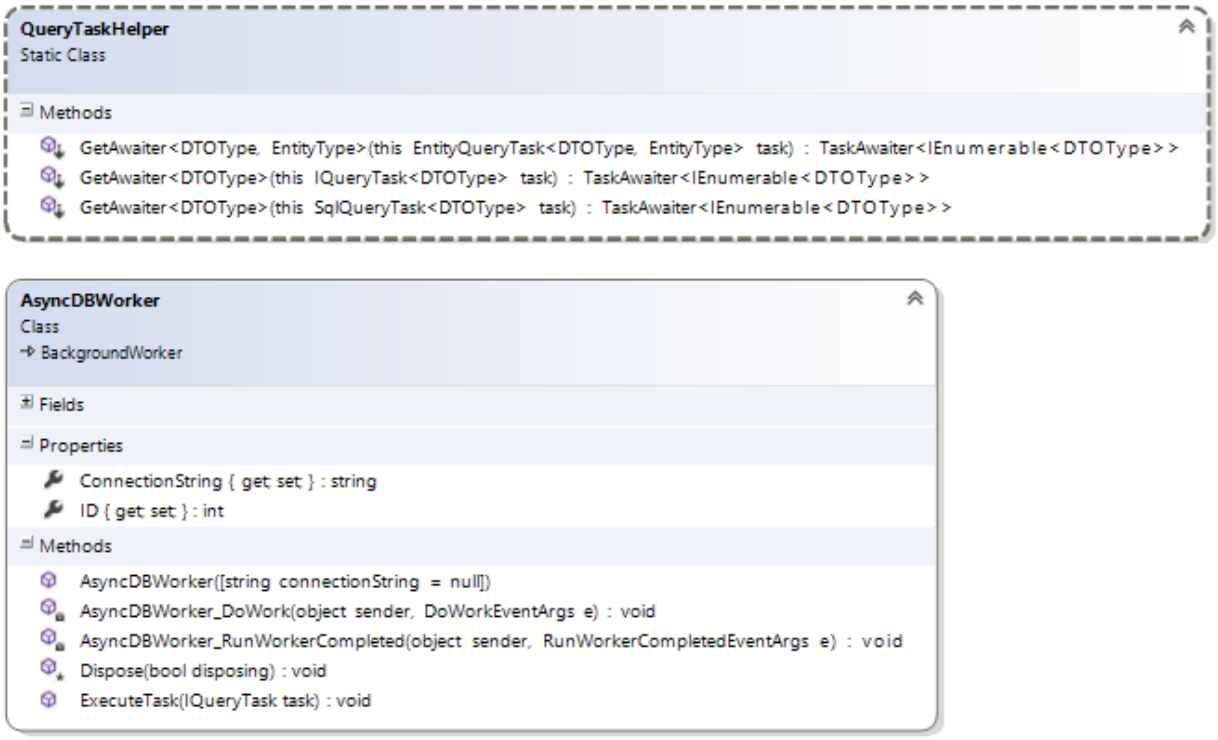


Рисунок 4.9 – Основні властивості та сигнатури методів класу AsyncDBWorker та сигнатури методів розширення IQueryTask<DTOType>

Для безпосереднього вилучення даних з бази даних було розроблено два класи: EntityQueryTask<DTOType, EntityType> та SqlQueryTask<DTOType> (див. рисунок 4.10)

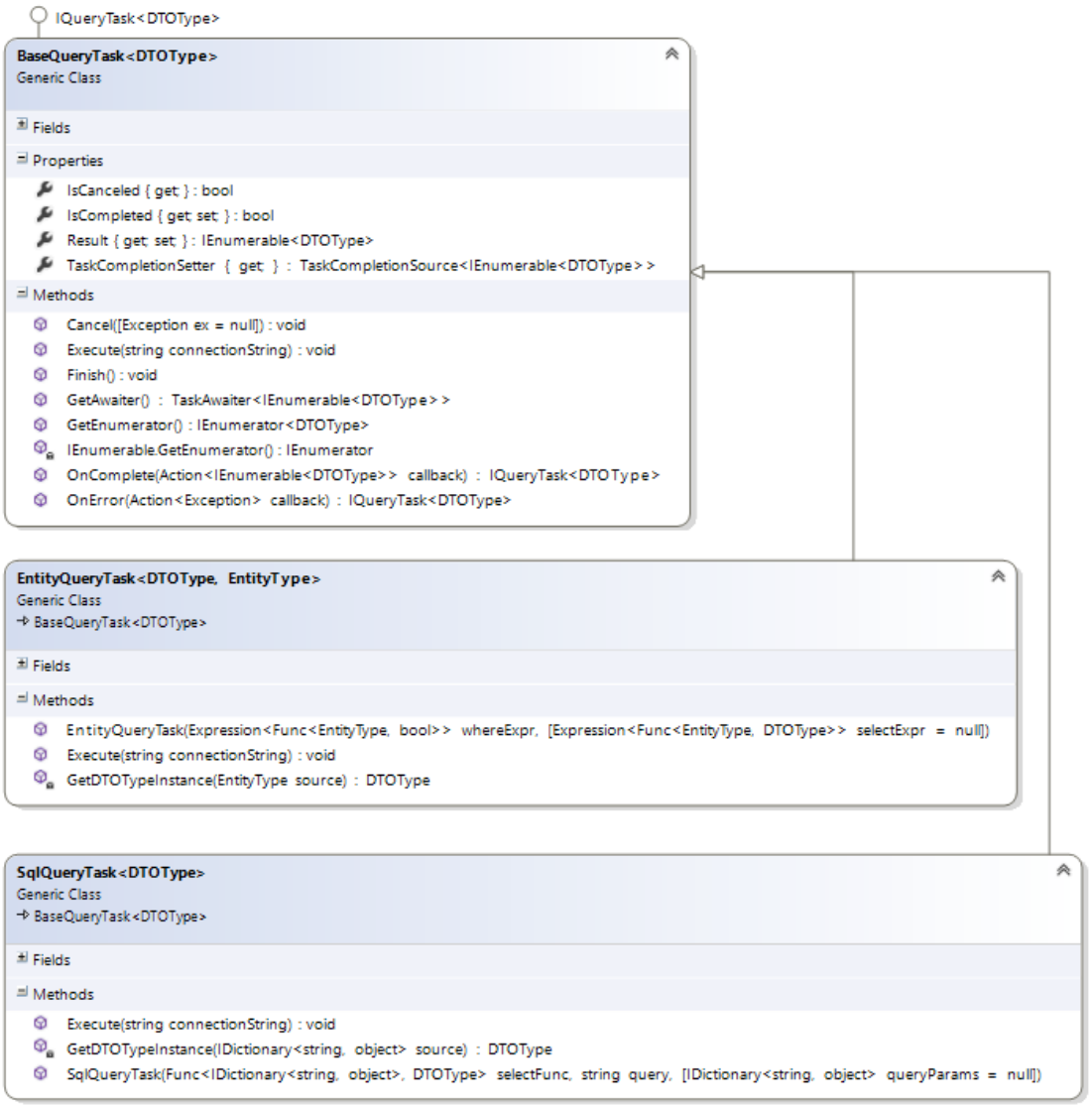


Рисунок 4.10 – Основні поля та сигнатури методів класів BaseQueryTask<DTOType>, EntityQueryTask<DTOType, EntityType> та SqlQueryTask<DTOType>

BaseQueryTask<DTOType> – базовий клас для задач, що має усі необхідні методи для виконання задачі, отримання її результату та обробки помилок при виконанні.

EntityQueryTask<DTOType, EntityType> – успадковує клас BaseQueryTask<DTOType> та використовується для виконання LINQ запитів з використанням DataContext та автоматичного перетворення отриманих об’єктів типу EntityType на DTO-об’єкти типу DTOType.

SqlQueryTask<DTOType> – успадковує клас BaseQueryTask<DTOType> та використовується для виконання звичайних SQL запитів.

Для DTO класів було розроблено два базових класи: DTOBaseClass та EditableDTO<EntityType>(див. рисунок 4.11).

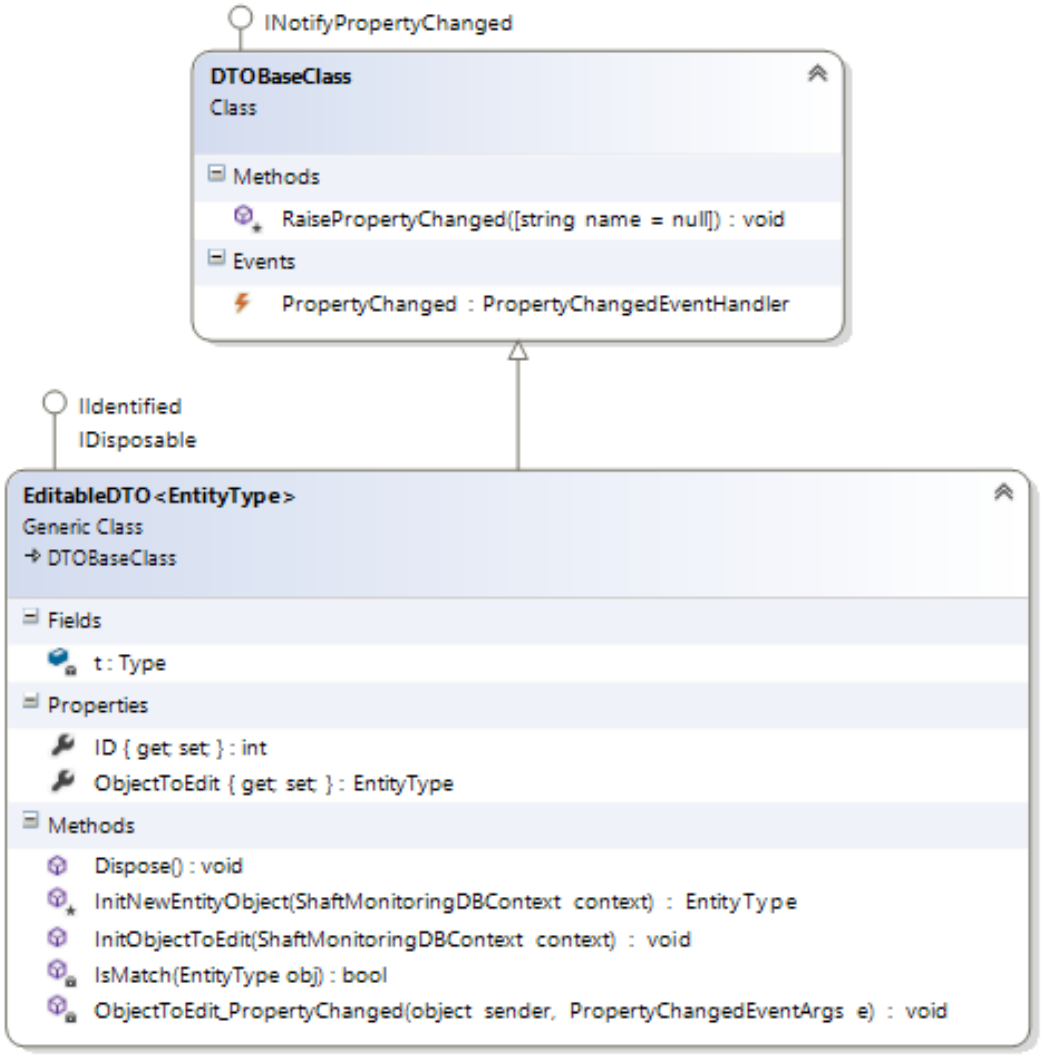


Рисунок 4.11 – Базові класи для DTO: DTOBaseClass та EditableDTO<EntityType>

DTOBaseClass містить лише реалізацію інтерфейсу INotifyPropertyChanged та слугує для упорядкування усіх DTO класів, що будуть створені у подальшому.

EditableDTO<EntityType> успадковує DTOBaseClass та є DTO класом, що може за необхідності модифікувати початковий об’єкт. Цей функціонал є необхідним для можливості редагування звітів.

DTO класи були створені для усіх класів, що є складовою структури підприємства. Вони мають однаковий перелік властивостей з початковим класом, але вже не є частиною DataContext, тому робота з ними більш швидка та проста.

Описана вище інфраструктура класів дозволяє паралельно виконувати декілька запитів до бази даних, кожен з яких оформлений у якості задачі. Це значно прискорює процес завантаження деревовидної структури та зменшує об’єм завантажених до пам’яті даних за рахунок використання DTO класів.

Також передбачена можливість редагування початкових об’єктів за необхідністю.

Оскільки виконання задач відбувається у додатковому потоці, це розвантажує головний потік додатку, у якому обробляється взаємодія з інтерфейсом, та дозволяє працювати з іншими елементами інтерфейсу поки іде завантаження даних з бази даних.

Безпосередньо дерево звітів складається з чотирьох компонентів:

– інтерфейс, що є розширенням компоненту RadTreeView з додаванням спеціальних властивостей залежності (Dependency Property), та функціоналом для завантаження дерева;

– контекст дерева – клас, що керує процесом завантаження дерева, містить допоміжні функції та зберігає копії значень властивостей залежностей для доступу до них з додаткових потоків;

– набір класів-обгорток (wrap) над створеними вище DTO тих компонентів, що відображаються у дереві. Класи-обгортки необхідні для керування процесом завантаження дочірніх елементів, створення додаткових властивостей для відображення та розширення функціоналу дерева за необхідності (наприклад, додавання контекстного меню);

– будівника запитів (Query Builder) що створює SQL-запити для фільтрації дерева звітів за багатьма критеріями: категорія структурного елементу, тип робіт, назва та дата моделі тощо.

Усі ці компоненти використовують розглянуту вище систему асинхронного доступу до бази даних, що робить роботу дерева звітів максимально швидкою.

Для редагування історії зміни статусів типів моделей буде використовуватися компонент, розглянутий у пункті 4.1.2. Для нього необхідно лише створити клас View Model, що реалізує інтерфейс IEditStateHistoryDialogVM та враховує специфіку роботи з сутністю MineElement_ModelTypeState.

Оскільки усі додані до звіту файли мають одразу завантажуватися до сховища Mine Advisor, було розроблено клас SpatialModelFileUploader, який контролює процес завантаження файлів до сховища та створення сутностей файлів моделей у базі даних.

Для розвантаження головного потоку додатку створюється процес Mine Advisor, що виконує розрахунки у додатковому потоці та дозволяє слідкувати за перебігом завантаження, зупиняти, продовжувати та переривати завантаження.

Безпосередньо завантаження файлів у сховище виконує компонент IRepositoryClient, який є сінглтоном та може бути викликаний у будь-якій частині додатку. У результаті завантаження файлів до сховища IRepositoryClient повертає GUID створеної сутності, який буде використано для створення сутності файлу у базі даних.

Основні поля та сигнатури методів класу SpatialModelFileUploader наведені на рисунку 4.12.

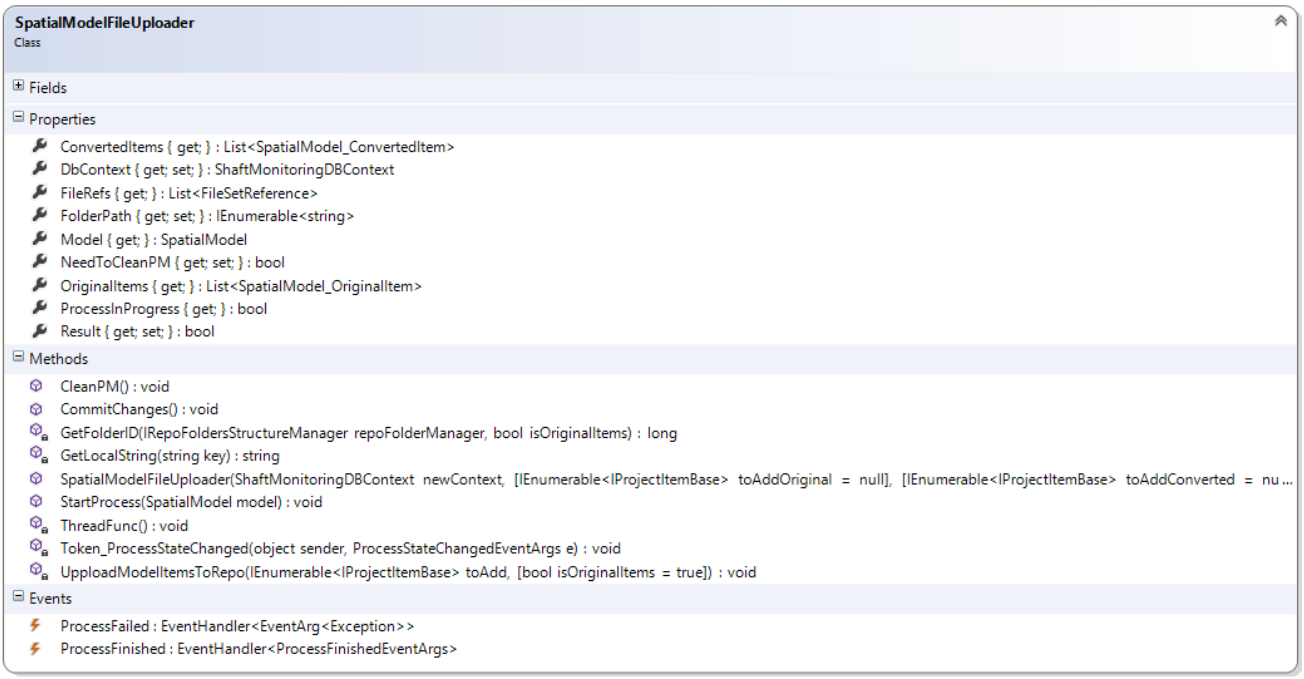
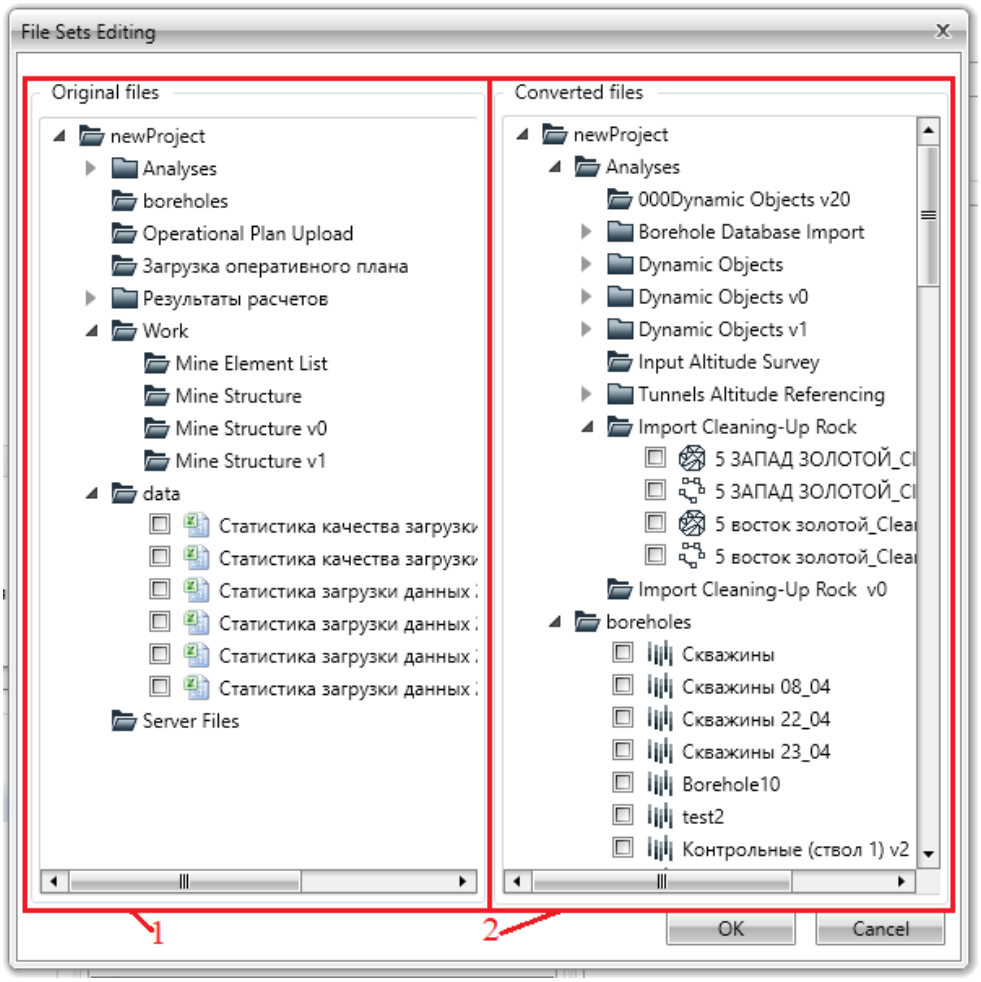


Рисунок 4.12 – Основі поля та сигнатури методів класу SpatialModelFileUploader

Додавання до звіту оригінальних та конвертованих файлів з переліку вже доданих до системи Mine Advisor відбуватиметься за допомогою класу LayerLoaderView, що дозволяє відобразити дерево проекту Mine Advisor(рисунок 4.7(1)) з фільтрацією. Два об’єкти цього класу будуть відображатися у діалоговому вікні: один з оригінальними файлами, другий з конвертованими. Приклад діалогового вікна зображено на рисунку 4.13.

Після вибору необхідних для додавання до звіту файлів, створюється сутність класу SpatialModelFileUploader, до якої передаються: обраний звіт, перелік обраних оригінальних файлів, перелік обраних конвертованих файлів та контекст даних, до якого необхідно додати нові сутності файлів моделей.



1 – дерево проекту тільк з оригінальними файлами; 2 – дерево проекту тільки з конвертованими файлами

Рисунок 4.13 – Діалогове вікно додавання до звіту файлів, що вже було завантажено до системи Mine Advisor

Завантаження оригінальних файлів безпосередньо з файлової системи відбувається за допомогою стандартного діалогового вікна вибору файлів (див. рисунок 4.14).

Після вибору бажаних файлів відбувається їх реєстрація у системі Mine Advisor та завантаження до дерева проекту Mine Advisor. Далі додані файли передаються до об’єкту класу SpatialModelFileUploader з усіма додатковими даними та починається процес завантаження.

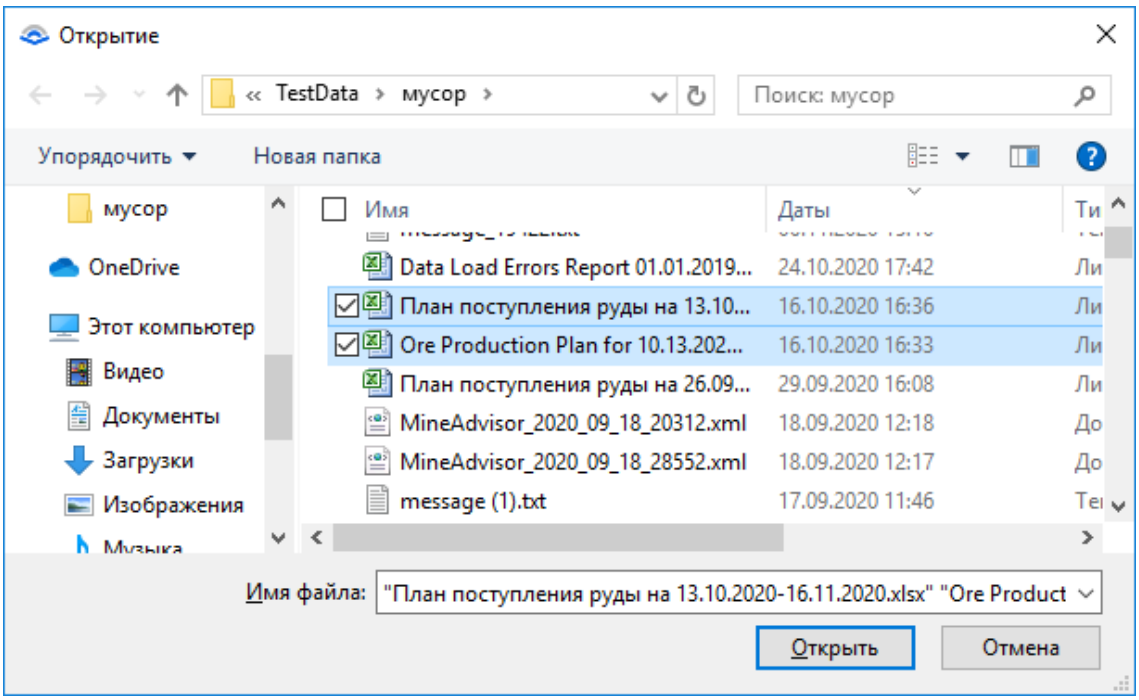


Рисунок 4.14 – Діалогове вікно вибору оригінальних файлів для завантаження до звіту

Для імпорту конвертованих файлів з файлової системи використовується компонент Mine Advisor – IImportManager. Цей компонент дозволяє імпортувати до системи Mine Advisor велику кількість різноманітних типів даних, частину з яких наведено на рисунку 4.15.

Після вибору файлів починається процес імпорту. Він відбувається, як і процес завантаження файлів до сховища, у додатковому потоці та реєструється у системі Mine Advisor для можливості керування.

Після завершення імпортування усіх необхідних файлів починається процес їх завантаження до сховища.

Після завершення завантаження, як оригінальних, так і конвертованих файлів, гілка дерева звітів, у якій знаходиться обраний звіт, оновлюється починаючи з типу робіт. Таким чином результати завантаження файлів видно одразу.

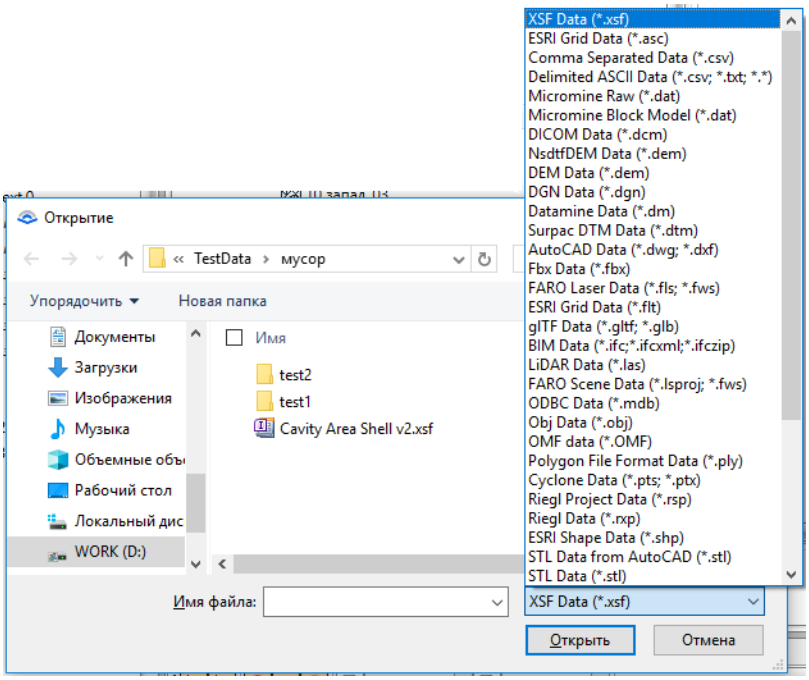


Рисунок 4.15 – Частина типів даних, доступних для імпорту до системи Mine Advisor

Виклик усіх наведених вище інструментів відбувається за допомогою команд та дескриптору панелі інструментів.

Команди створюються на View Model, та є класами, що реалізують інтерфейс ICommand. Такі класи мають два обов’язкових методи: CanExecute, що повертає значення типу bool та визначає, чи можливо зараз виконати команду, та Execute, що виконує дії команди.

Дескриптор панелі інструментів є компонентом системи Mine Advisor та дає змогу налаштовувати панель інструментів для сторінок майстрів: визначати перелік кнопок для запуску інструментів, розбивати їх на групи, давати назви, описи та зображення (рисунок 4.7(12)).

Процес додавання звіту до бази даних є наступним: створюється нова сутність типу SpatialModel, якій задаються значення за замовчуванням(сьогоднішня дата та назва «Новый элемент/New element») та виставляються зв’язки із сутностями структурного елементу підприємства та типу робіт, що були обрані. Після цих дій новий звіт є доданим до поточного контексту даних та буде збережено у базі даних під час наступного зараження змін.

Процес видалення звіту складається лише з додавання поточного звіту до переліку звітів, що будуть видалені при наступному збереженні змін у поточному контексті даних.

Для відображення додавання та видалення звітів у дереві звітів без необхідності перезавантажувати дерево, було розроблено спеціальні методи у класі-обгортці для типу робіт:

```
void RemoveChildren(object toRemove);
void RemoveChildrens(IEnumerable<object> toRemove);
void AddChildren(object toAdd);
void AddChildrens(IEnumerable<object> toAdd);
```

Ці методи редагують колекцію дочірніх елементів у класі-обгортці для типу робіт та викликають оновлення відображення усіх звітів, що належать до цього типу робіт та конкретного структурного елементу.

Редагування даних звітів(зміна дати створення та назви) відбувається за допомогою прив’язки даних у коді XAML-розмітки View сторінки.

Таким чином було розроблено сторінку завантаження до системи звітів про виконання завдань, що надає усі необхідні інструменти для додавання, видалення та редагування звітів, а також додавання та видалення оригінальних та конвертованих файлів.

4.2 Розробка підсистеми формування правил перевірки звітності про виконання завдань

Підсистема формування правил базується на типах робіт та правилах, що з ними пов’язані. Правила поділяються на дві групи відповідно до критеріїв якості, розглянутих у розділі 2.2: правила перевірки періодичності створення та завантаження звітів до системи та правила перевірки переліку файлів звіту.

Для створення та редагування типів робіт буде розроблено окрему сторінку майстру «Менеджер цифрового рудника (Digital Mine Manager)». Ця сторінка

даватиме змогу додавати, видаляти й модифікувати типи робіт та створювати для них правила перевірки звітності.

Оскільки було вирішено, що періодичність у типі робіт зберігається у вигляді JSON, то необхідно розробити класи, що відповідали б за зчитування цього JSON та давали змогу визначати необхідні дані (наприклад, дати початку та кінця періоду) для кожної з необхідних періодичностей.

Правила для переліку файлів звітів зберігаються безпосередньо у відповідних сутностях бази даних (див. розділ 3.1), тому необхідно лише створити редактор для них на сторінці редагування типів робіт.

До критеріїв оцінки якості виконання завдання, розглянутих у розділі 2.2, було вирішено додати ще один, що враховує специфіку роботи системи Mine Advisor – наявність доданих до звіту конвертованих файлів. Правило для цього критерію належить до групи правил перевірки переліку файлів звіту, тому зберігатиметься у тих самих сутностях бази даних. Конвертовані файли зазвичай є просторовим(3D) зображенням структурного елементу підприємства, тому є доцільним їх додавання до звітів з робіт, що заміряють положення кар’єру, розмітку блоків для видобутку тощо.

4.2.1 Розробка класів періодичностей типів робіт

Відповідно до розділу 2.3 усього у системі буде п’ять періодичностей складання звітів : тижнева, вахтна, місячна, квартальна, річна. Для кожної з них буде розроблено відповідний клас, що матиме можливість повертати дати початків та кінців періодів, у які необхідно завантажувати звіти та локалізовані назви цих періодів. Також буде створено додатковий клас, що визначатиме відсутність періодичності.

Оскільки перелік необхідних методів для усіх періодичностей є одноковим, було розроблено базовий клас PeriodicityManagerBase, що реалізує інтерфейс IModelTypePeriodisity:

```
    public interface IModelTypePeriodisity: INamedReadOnly,
                                            INotifyPropertyChanged
    {
        IEnumerable<string> AvailableNames{get;}
        IEnumerable<string> AvailableShortNames { get; }
        string Type { get;}
        int[] PeriodDelimeters { get; }
        int MinModelsCount { get; set; }
        Dictionary<int, string> GetLocalizedShortNames(int index);
        Dictionary<int, string> GetLocalizedNames(int index);
        string PreferredName(DateTime date);
        DateTime GetPeriodStartDate(DateTime date);
        DateTime GetNextPeriodStartDate(DateTime date);
        string GetPeriodName(DateTime date,
System.Globalization.CultureInfo
                             locale = null);
        string GetPeriodShortName(DateTime date,
                             System.Globalization.CultureInfo locale =
null);
        void UpdateNames();
    }
```

Усі класи періодичностей є спадкоємцями класу PeriodicityManagerBase та за необхідністю перевизначають його функції.

Для створення періодичностей на основі JSON було розроблено два класи: ModelTypePeriodisityTypes та TypePeriodicityManagerFabric. Приклад JSON для тижневої періодичності, що зберігається у базі даних:

```
{ "Type":"SMM_weekPeriodisity" }
```

Клас ModelTypePeriodisityTypes має властивість Type, яка дозволяє зчитувати JSON для періодичності, та перелік константних рядків, що визначають доступні типи періодичностей.

TypePeriodicityManagerFabric використовується як фабрика об’єктів періодичностей, що створює їх на основі JSON. Для цього було розроблено наступну функцію:

```
public static IModelTypePeriodisity GetPeriodisityManager(string jString)
{
    if(string.IsNullOrEmpty(jString)) {
        return new NonePeriodicityManager();
```

```
    }

    ModelTypePeriodisityTypes typeDescryber =

      JsonConvert.DeserializeObject<ModelTypePeriodisityTypes>(jString);
    IModelTypePeriodisity result = null;
    switch (typeDescryber.Type)
    {
        case "SMM_nonePeriodisity":
            result = new NonePeriodicityManager();
            break;
        case "SMM_weekPeriodisity":
            result = new WeekPeriodicityManager();
            break;
        case "SMM_rotationPeriodisity":
            result = new RotationPeriodicityManager();
            break;
        case "SMM_monthPeriodisity":
            result = new MonthPeriodicityManager();
            break;
        case "SMM_quarterPeriodisity":
            result = new QuarterPeriodicityManager();
            break;
        case "SMM_halfYearPeriodisity":
            result = new HalfYearPeriodicityManager();
            break;
        case "SMM_yearPeriodisity":
            result = new YearPeriodicityManager();
            break;
        default:
            throw new
InvalidOperationException(ResourcesManager.Current.Local.GetString("SMM_inv
alidPeriodisityTypeJson"));
    }
    return result;
}
```

Ця функція приймає у якості параметру JSON рядок, що зчитується у об’єкт класу ModelTypePeriodisityTypes. Якщо властивість Type цього об’єкту відповідає якомусь із доступних типів періодичностей, то створюється об’єкт відповідного класу періодичності.

Також клас TypePeriodicityManagerFabric має статичну властивість PeriodisityTypes, що повертає перелік об’єктів усіх доступних класів періодичностей.

Для можливості зміни періодичності типу робіт через інтерфейс користувача було розроблено додаткову властивість у класі SpatialModel_Type – PeriodicityObject, та функція PeriodicityObject_PropertyChanged, що обробляє зміну значення цієї властивості.

```
public IModelTypePeriodisity PeriodicityObject {
    get {
        this.periodicityObject = TypePeriodicityManagerFabric
                                 .GetPeriodisityManager(this.Periodisity);
        return this.periodicityObject;
    }
    set {
      if(value != null &&
value.GetType()==this.periodicityObject.GetType()){
          return;
      }
      if (this.periodicityObject != null) {
          this.periodicityObject.PropertyChanged -=
                                      PeriodicityObject_PropertyChanged;
      }
      this.periodicityObject = value;
      if(this.periodicityObject != null) {
          this.periodicityObject.PropertyChanged +=

     PeriodicityObject_PropertyChanged;

this.PeriodicityObject_PropertyChanged(this.periodicityObject,null);
      }
      else {
          this.Periodisity = String.Empty;
     }
     this.SendPropertyChanged("PeriodicityObject");
    }
}

private void PeriodicityObject_PropertyChanged(object sender,
                           System.ComponentModel.PropertyChangedEventArgs
e)
{
    var toSerialize = new { Type = this.periodicityObject.Type};
    this.Periodisity = Newtonsoft.Json.JsonConvert
                                      .SerializeObject(toSerialize);
}
```

Частина get властивості PeriodicityObject повертає новий об’єкт поточної періодичності, який створюється у методі GetPeriodisityManager (див. лістинг 4.2).

Частина set властивості PeriodicityObject робить необхідні перевірки, підписується на подію PropertyChanged нового об’єкту та викликає функцію PeriodicityObject_PropertyChanged для зміни значення властивості Periodicity та збереження змін у базі даних. Завдяки підписці на подію PropertyChanged при зміні поточного об’єкту періодичності буде також викликатися функція PeriodicityObject_PropertyChanged, що зробить автоматичним оновлення значення у базі даних.

Функція PeriodicityObject_PropertyChanged створює новий об’єкт анонімного типу [25] з властивістю Type, у яку записується тип поточного об’єкту періодичності. Далі цей об’єкт перетворюється на JSON та записується до бази даних.

Таким чином було розроблено та модифіковано усі необхідні компоненти для роботи з періодичністю типів робіт. Завдяки зберіганню даних періодичності у JSON не має необхідності змінювати структуру базу даних при додаванні нових властивостей до періодичності, а розроблені класи полегшують роботу з періодичністю.

4.2.2 Розробка сторінки редагування типів робіт

Сторінка редагування типів робіт повинна давати можливість додавати, видаляти та модифікувати типи робіт, додавати, видаляти та модифікувати правила перевірки звітів. Також сторінка повинна перевіряти правильність введених даних, а саме виконання наступних умов:

– усі типи робіт повинні мати назву та категорію структурного елементу, до якої вони відносяться;

– назви усіх типів робіт мають бути унікальними;

– усі правила для переліку файлів звіту повинні мати назву та маску для перевірки назви файлів;

– назви усіх правил перевірки мають бути унікальними в рамках одного типу робіт;

– усі правила перевірки звітів повинні мати тип робіт, до якого вони належать.

Перевірка виконання цих умов буде проводитися безпосередньо перед збереженням змін до бази даних з відображенням відповідного повідомлення про помилку у заповнені даних.

Крім цього існують додаткові умови, що обмежують можливості редагування типу робіт :

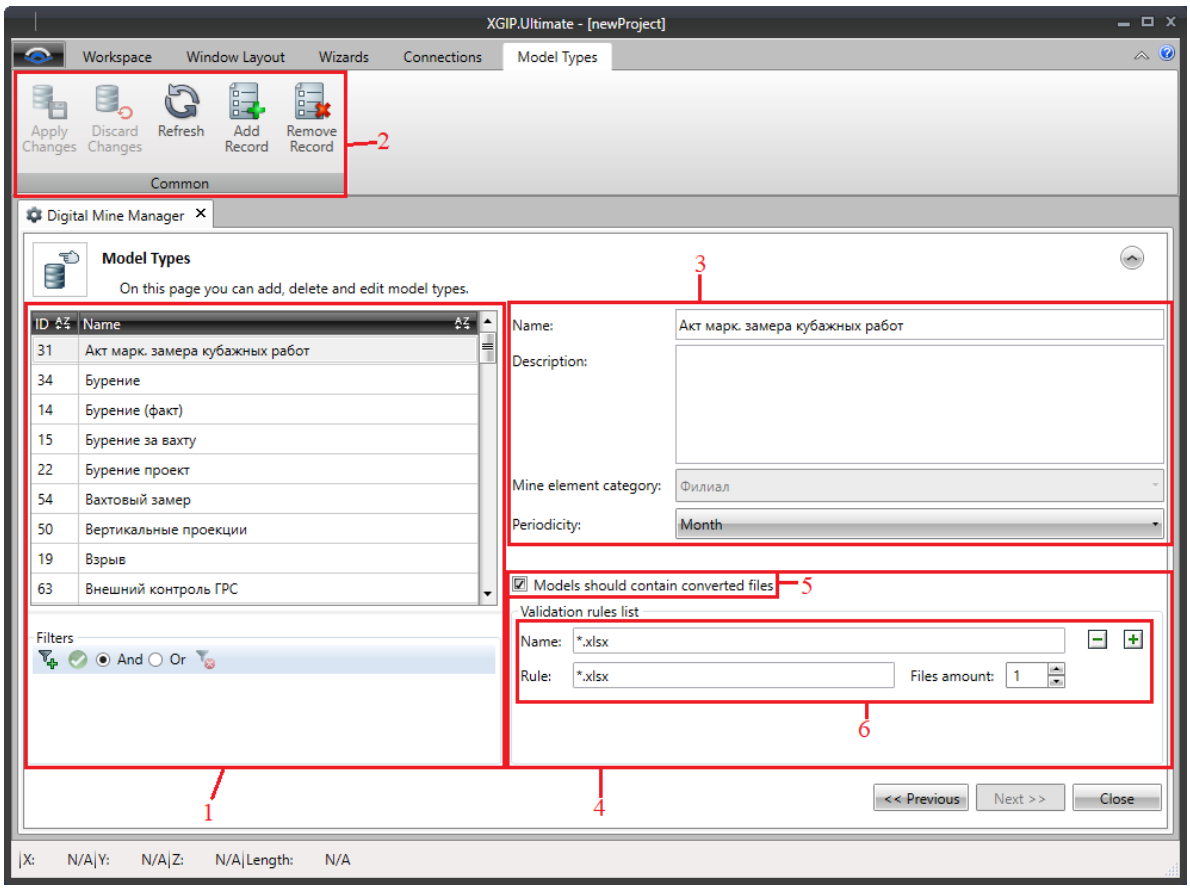
– тип робіт не може бути видалений, якщо вже були створені звіти з цього типу робіт;

– категорія структурного елементу, до якої відноситься тип робіт, не може бути змінена якщо вже були створені звіти з цього типу робіт;

– при зміні категорії структурного елементу типу робіт необхідно перевіряти наявність історії змін станів цього типу робіт та, за її наявності, повідомляти користувача про повне її видалення.

Ці обмеження та попередження необхідні для запобігання втрати користувачем звітів, бо при зміні категорії структурного елементу типу робіт вже створені звіти не будуть відображатися у дереві звітів, а історія зміни станів втратить сенс, адже стани типів робіт зазначаються для кожного структурного елементу підприємства, на якому вони проводяться.

Приклад інтерфейсу сторінки редагування типів робіт наведено на рисунку 4.16.



1 – таблиця усіх типів робіт з панеллю фільтрації; 2 – панель інструментів; 3 – поля для редагування типу робіт; 4 – поля для редагування правил до звітів; 5 – чекбокс(CheckBox) для зазначення необхідності додавання конвертованих файлів до звіту; 6 – поля для редагування правил до переліку файлів звіту

Рисунок 4.16 – Приклад інтерфейсу сторінки редагування типів робіт

Для відображення таблиці з типами робіт та панелі фільтрації (рисунок 4.16(1)) було використано компонент Mine Advisor – GridView, що є розширенням компоненту Telerik – RadGridView. Цей компонент вже налаштований для відображення колекції елементів, що задається за допомогою прив'язки даних, та має функціонал для сортування цієї колекції. Панель фільтрації забезпечує можливість фільтрації типів робіт за ID та назвою.

Панель інструментів, як і на сторінці завантаження до системи звітів про виконання завдань, створюється завдяки дескриптору панелі інструментів та команд (див. розділ 4.1.3).

Поля редагування типу робіт (рисунок 4.16(3)) створені завдяки стандартним компонентам: TextBox для редагування текстових властивостей та

Telerik RadComboBox для редагування категорії структурних елементів та періодичності.

Для заборони зміни категорії структурного елементу у класі SpatialModel_Type було створено властивість IsChangeMineCategoryAlowed:

```
public bool IsChangeMineCategoryAlowed
{
    get { return !this.SpatialModels.Any(); }
}
```

Та створено прив’язку даних властивості RadComboBox IsEnabeld(«є увімкненим») до властивості IsChangeMineCategoryNotAlowed.

Для перевірки наявності історії зміни станів типу робіт під час зміни категорії структурного елементу було створено подію(event) PropertyChangedCheck та спеціальний клас аргументів події PropertyChangedCheckEventArgs. Ця подія викликається під час зміни категорії структурного елементу типу робіт. Клас PropertyChangedCheckEventArgs має поле IsCanceled типу bool, що вказує, чи відмінено зміну значення властивості.

View Model сторінки підписується на цю подію та при її виклику запитує у користувача підтвердження зміни категорії структурного елементу типу робіт, попереджуючи про видалення усієї історії змін станів.

Для роботи з різними типами перевірки набору файлів звіту було створено перерахування(enum) ValidationRules:

```
public enum ValidationRules
{
    ConvertedFilesRequired = 1,
    FileSetRequirements = 2
};
```

Значення цього enum зберігаються у полі RuleType сутності ModelTypeValidationRule та слугують для визначення способу перевірки звітів. Наприклад, якщо значення RuleType дорівнює FileSetRequirements, то робиться

перевірка переліку файлів звіту за правилами, зазначеними у пов’язаних сутностях ModelTypeFileSetValidation.

Для визначення того, чи необхідні конвертовані файли у звіті використовується CheckBox(рисунок 4.16(5)), властивість якого IsChecked визначається прив’язкою даних до властивості класу SpatialModel_Type – AreConvertedFilesRequired:

```
public bool AreConvertedFilesRequired {
  get {
    return this.ModelTypeValidationRules.Any(x =>
(ValidationRules)x.RuleType
                                    ==
ValidationRules.ConvertedFilesRequired);
  }
  set {
    if(this.onCreatedRullesSet == null) {
      this.onCreatedRullesSet = this.ModelTypeValidationRules.ToList();
    }
    if(value) {
      if(this.ModelTypeValidationRules.Any(x => (ValidationRules)x.RuleType
                                    ==
ValidationRules.ConvertedFilesRequired)){
        return;
      }
      ModelTypeValidationRule newRule = new ModelTypeValidationRule() {
           RuleType = (int)ValidationRules.ConvertedFilesRequired,
           SpatialModel_Type = this};
      this.SendPropertyChanged("AreConvertedFilesRequired");
    }
    else
    {
      if (this.ModelTypeValidationRules.Any(x => x.RuleType ==
           (int)ValidationRules.ConvertedFilesRequired)) {
        this.ModelTypeValidationRules.Where(x => x.RuleType ==

     (int)ValidationRules.ConvertedFilesRequired).ToArray()
                     .ForAll(x => x.SpatialModel_Type = null);
        this.SendPropertyChanged("AreConvertedFilesRequired");
      }
    }
  }
}
```

Частина get цієї властивості повертає true, якщо з поточним типом робіт пов’язана хоча б одна сутність ModelTypeValidationRule зі значенням RuleType рівним ValidationRules.ConvertedFilesRequired.

Частина set цієї властивості працює наступним чином:

– якщо нове значення дорівнює true, то перевіряється наявність хоча б однієї пов’язаної з поточним типом робіт сутності ModelTypeValidationRule зі значенням RuleType рівним ValidationRules.ConvertedFilesRequired. Якщо така сутність присутня, то ніяких змін робити не потрібно. В іншому випадку необхідно створити таку сутність;

– якщо нове значення дорівнює false, то для усіх пов’язаних з поточним типом моделей сутностей ModelTypeValidationRule зі значенням RuleType рівним ValidationRules.ConvertedFilesRequired видаляється зв’язок з поточним типом моделі.

Для видалення з бази даних сутностей ModelTypeValidationRule, що не пов’язані з типом робіт перед збереженням змін у View Model викликається наступна функція:

```
protected void BeforeApply() {
  this.DBContext.ModelTypeValidationRules.DeleteAllOnSubmit<ModelTypeValida
  tionRule>(this.DBContext.ActualData<ModelTypeValidationRule>().Where(x =>
  x.SpatialModel_Type == null).ToArray());

  this.DBContext.ModelTypeFileSetValidations.DeleteAllOnSubmit<ModelTypeFil
  eSetValidation>(this.DBContext.ActualData<ModelTypeFileSetValidation>().W
  here(x => x.ModelTypeValidationRule == null).ToArray());
}
```

Також вона видаляє усі сутності ModelTypeFileSetValidation, що не пов’язані з сутностями ModelTypeValidationRule. Це дозволяє уникнути появі у базі даних записів, що є неправильними та не мають сенсу.

Для редагування переліку необхідних у звіті файлів використовується стандартний компонент WPF – ItemsControl(рисунок 4.16(6)). Цей компонент

дозволяє відображати колекцію елементів з використанням шаблону відображення.

Для редагування значень кожного окремого правила для файлу у переліку використовуються стандартні TextBox(для назви типу документу та маски назви файлу) та Telerik RadNumericUpDown(для зміни кількості файлів цього типу). Зміна значень відповідних властивостей об’єктів класу ModelTypeFileSetValidation відбувається завдяки прив’язці даних.

Додавання та видалення необхідних файлів відбувається завдяки командам ViewModel, що викликається при натиску відповідних кнопок у інтерфейсі користувача.

Перевірка правильності введених даних відбувається під час збереження змін та аналогічна перевірці, розглянутій у розділі 4.1.1.

Таким чином було розроблено сторінку редагування типів робіт, що дає можливість додавати, видаляти та модифікувати типи робіт та правила перевірки звітів. Також ця сторінка слідкує за правильністю введених даних, запобігаючи виникненню проблемних ситуацій у процесі користування системою.

4.3 Розробка підсистеми аналізу якості звітності про виконання завдань

Підсистему аналізу якості звітності про виконання завдань було вирішено винести до окремого інструменту «Статистика ошибок загрузки данных(Data Load Errors Statistic)».

Як було зазначено у розділі 4, для створення інструменту у системі Mine Advisor необхідно створити майстер, що реєструється у системі, відповідає за відображення необхідної кнопки на панелі інструментів та керує роботою інструменту; та необхідні сторінки, що є елементами бізнес-логіки.

4.3.1 Розробка майстру інструменту аналізу якості звітності про виконання завдань

Інструмент аналізу якості звітності складається з двох сторінок: стандартної сторінки підключення до бази даних, що вже присутня у системі Mine Advisor, та самої сторінки аналізу якості звітності.

Сторінка підключення до бази даних дає змогу обрати необхідне підключення для роботи майстрів з тих, що вже використовувалися, або ввести данні для нового підключення. Данні для підключення включають до себе повну адресу серверу, на якому розгорнута база даних, базу даних та дані авторизації. Інтерфейс сторінки підключення до бази даних зображено на рисунку 4.17.

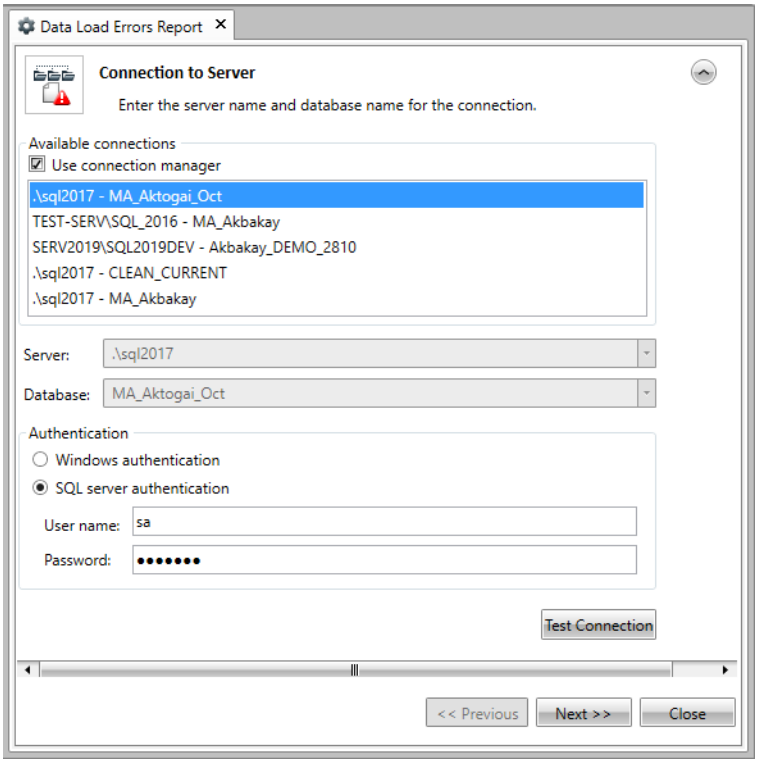


Рисунок 4.17 – Сторінка підключення до бази даних

Сторінку аналізу якості звітності буде розглянуто у розділі 4.3.2.

Для організації сторінок у маршрут та роботи майстру у системі Mine Advisor його клас – ModelStatisticsReportWizard, успадковується від базового класу майстрів Mine Advisor – WizardController, та перевизначає його метод Initialize.

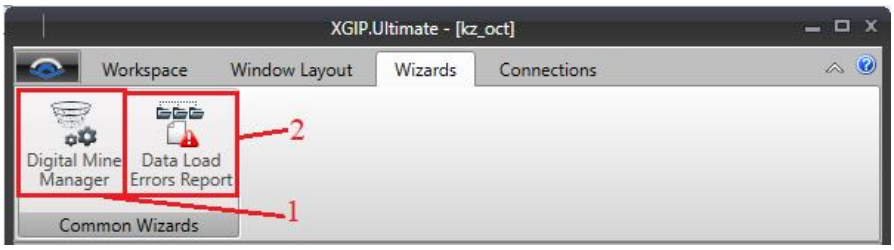
```
protected override void Initialize(){
    this.WizardCaptionKey = CaptionKey;
    this.SwitchNoResultWizardButtonSet();

    this.connPage = new
                DataBaseConnectionElementManagerPage(this.ServiceLocator);
    this.connPage.SkipShowingPage = this.NeedToSkeepConPage();
    this.connPage.ValidatingData += connPage_ValidatingData;
    this.connPage.DataReady += new EventHandler(this.ConnPage_DataReady);

    this.reportPage = new StatisticsReportPage(this.ServiceLocator);

    this.Route.AddPage(this.connPage);
    this.Route.AddPage(this.reportPage);
}
```

Цей метод викликається після створення об’єкту класу майстра під час запуску інструменту через панель інструментів додатку, зображену на рисунку 4.18.



1 – кнопка для запуску інструменту «Менеджер цифрового рудника(Digital Mine Manager)»; 2 – кнопка для запуску інструменту «Статистика ошибок загрузки данных(Data Load Errors Statistic)»

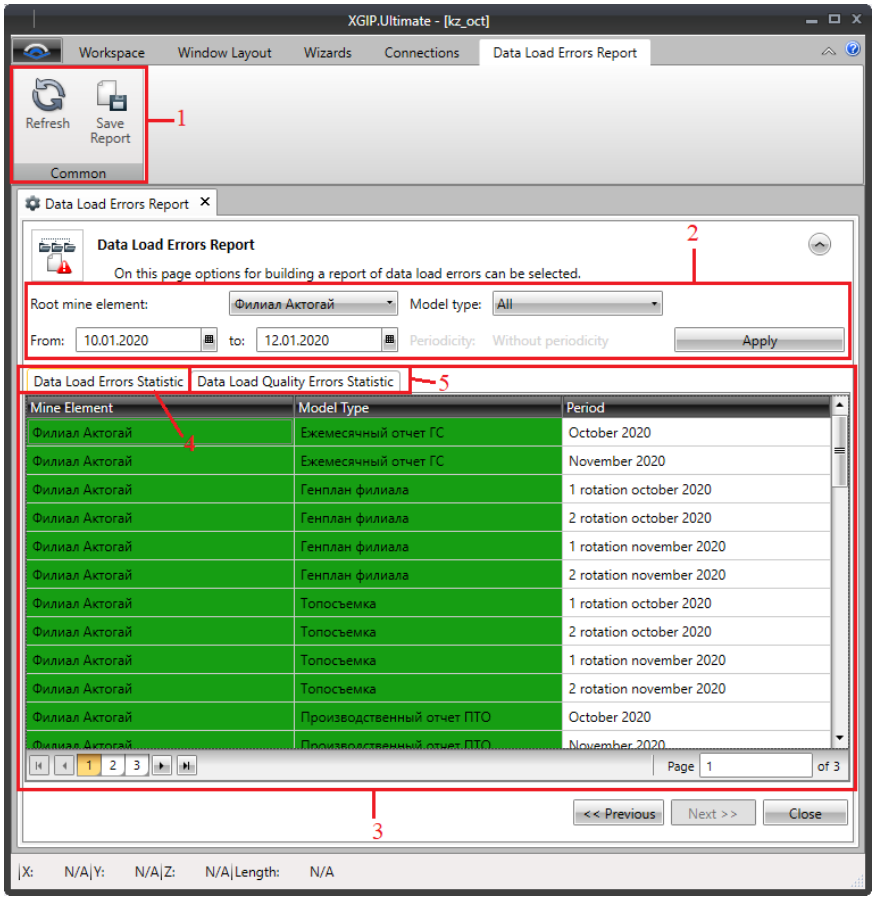
Рисунок 4.18 – Панель інструментів додатку

У цьому методі створюються об’єкт класу сторінки підключення до бази даних (DataBaseConnectionElementManagerPage) та сторінки аналізу якості звітності (StatisticsReportPage). Усі сторінки успадковуються від класу WizardPageWithUI та мають перелік подій, що викликаються впродовж роботи сторінки. На деякі з цих подій сторінки підключення до бази даних (ValidatingData та DataReady) підписується майстер. У обробнику події ValidatingData проводиться перевірка введених користувачем даних, наявності у базі даних необхідних записів тощо. У обробнику події DataReady отримане підключення передається сторінці аналізу якості звітності.

Після створення сторінок та підписки на події, сторінки додаються до маршруту(Route) у порядку їх відображення користувачеві.

4.3.2 Розробка сторінки аналізу якості звітності про виконання завдань

Приклад інтерфейсу сторінки аналізу якості звітності про виконання завдань наведено на рисунку 4.19.



1 – панель інструментів; 2 – інструменти для визначення критеріїв аналізу; 3 – область виведення результатів аналізу; 4 – вкладка з результатами аналізу за періодичністю; 5 – вкладка з результатами аналізу за переліком файлів

Рисунок 4.19 – Приклад інтерфейсу стоірнки аналізу якості звітності про виконання завдань

Сторінка аналізу якості звітності про виконання завдань виробничим персоналом повинна давати наступні можливості:

а) налаштовувати критерії для аналізу, а саме : задавати:

1) задавати дати початку та кінця аналізу;

2) задавати бажаний тип робіт для аналізу;

3) задавати структурний елемент підприємства, що буде прийматися за кореневий під час аналізу;

б) попередній огляд результатів аналізу;

в) зберігати результати аналізу у документі Microsoft Excel з формату .xlsx.

Для налаштування критеріїв аналізу на сторінку було додано наступний перелік компонентів:

– два календаря (DatePicker) для вибору часових меж аналізу;

– випадаючий список (ComboBox) для вибору типу робіт, для якого проводиться аналіз;

– кнопка з випадаючим змістом (DropDownButton), для вибору структурного елементу, що буде кореневим для аналізу. Випадаючим змістом кнопки є дерево структури підприємства;

– кнопку «Применить(Apply)» за натиском якої починається аналіз якості звітності за обраними критеріями.

Данні, що відображаються у цих компонентах та результат взаємодії користувача з ними передаються до View Model за допомогою прив’язки даних.

Перелік типів робіт доступних до вибору фільтрується в залежності від обраного структурного елементу. Також є можливість відмінити вибір та виконати аналіз за усіма типами робіт.

Для попереднього перегляду результатів аналізу на сторінку було додано компонент керування вкладками (TabControl), що містить у собі дві вкладки:

– «Статистика ошибок загрузки данных (Data Load Errors Statistic)» на якій відображаються періоди, під час котрих не було завантажено звіту з виконання конкретного типу робіт до конкретного структурного елементу підприємства;

– «Статистика ошибок качества загрузки данных (Data Load Quality Errors Statistic)» на якій відображаються звіти, що не мають усіх необхідних файлів,

зазначених у правилах перевірки для конкретного типу робіт конкретного структурного елементу підприємства;

Для відображення попереднього огляду використовується компонент RadGridView, що дозволяє у табличному вигляді відобразити колекцію елементів.

Клітинки таблиць у колонках «Структурный элемент (Mine Element)» та «Тип модели(Model Type)» мають колір поточного стану структурного елементу та типу моделі.

На вкладці «Статистика ошибок загрузки данных» у колонці «Период (Period)» відображається назва періоду, у який не було додано звіт. Якщо тип робіт не має періодичності, то перевіряється наявність хоча б однієї моделі для кожного структурного елементу підприємства, на яких цей тип робіт проводиться, за увесь час навіть поза межами обраних дат аналізу. Якщо жодної моделі не має, то відповідна клітинка у колонці «Период» стає синьою та виводиться напис «Всё время(All time)». Така додаткова перевірка дає змогу виявити типи робіт, що не мають періодичності, але звіт з яких повинен бути.

На вкладці «Статистика ошибок качества загрузки данных» у таблиці окрім колонки «Период» є дві колонки (див. рисунок 4.20):

– «Тип ошибки(Error type)» у якій виводиться тип помилки, що був допущений під час завантаження звіту до системи. Наприклад «Пустая модель(Empty model)», якщо до звіту не було додано файлів взагалі, чи «Нарушено правило валидации (Validation rule failed)», якщо не усі необхідні файли додані до звіту;

– «Модель(Model)» у якій виводиться назва та дата звіту, що має помилки.

Така форма подачі результатів аналізу дає змогу однозначно виявити проблемні структурні елементи чи типи робіт та провести необхідні заходи з покращення якості звітності.

Data Load Errors Statistic | Data Load Quality Errors Statistic

Mine Element	Model Type	Period	Error Type	Model
Филиал Актогай	Test Validation Rule	1 rotation november 2020	Validation rule failed	11.02.2020
Филиал Актогай	Test Validation Rule	2 rotation november 2020	Empty model	11.17.2020

Рисунок 4.20 – Результат аналізуякості звітності за критерієм переліку необхідних файлів

Важливо відзначити, що для типів робіт, які мають періодичність, аналіз проводиться лише за тими періодами, що є завершеними та частково чи повністю попадають у дати аналізу. Так, наприклад при обраній кінцевій даті аналізу 12-те грудня 2020 року місяць грудень буде виключено з аналізу для типів робіт, що мають місячну періодичність.

Для збереження результатів аналізу було розроблено клас ReportDataRowDescriptor:

```
public class ReportDataRowDescriptor
{
    public ReportDataRowDescriptor()
    {
        this.MainColor = Brushes.Transparent.Color.ToUint();
        this.ErrorColor = this.MainColor;
    }

    public bool IsFirstInGroup { get; set; }
    public string MineElement { get; set; }
    public string ModelType { get; set; }
    public string ErrorType { get; set; }
    public string Period { get; set; }
    public string Periodisity { get; set; }
    public string Model { get; set; }
    public long MainColor { get; set; }
    public long ErrorColor { get; set; }
    public DateTime DateFrom { get; set; }
    public DateTime DateTo { get; set; }
    public int MineElementID { get; set; }
    public string AdditionalData { get; set; }
}
```

Цей клас є простим DTO, що описує тип помилки та має властивості для визначення структурного елементу, типу робіт та звіту у яких було виявлено

помилку. Також цей клас зберігає додаткову інформацію про колір клітинок, та властивість IsFirstInGroup, що необхідна для групування клітинок у документі Microsoft Excel.

Для спрощення процесу перевірки звітів за критерієм наявності необхідних файлів у класі SpatialModel_Type було розроблено функцію GetFileSetValidationRules, що повертає правила типу робіт у вигляді словнику(Dictionary), ключем у якому є маска для назви файлу, а значенням виступає кількість файлів що мають відповідати цій масці.

Для перевірки переліку файлів звіту у класі SpatialModel було розроблено функцію AreAllRequiredFilesPresent:

```
public bool AreAllRequiredFilesPresent(){
  var fileRules = this.SpatialModel_Type.GetFileSetValidationRules();
  if(fileRules.Any() && !this.SpatialModel_OriginalItems.Any() ||
      (!this.SpatialModel_OriginalItems.Any() &&
      !this.SpatialModel_ConvertedItems.Any())){
       return false;
  }

  string[] originalFileNames = this.SpatialModel_OriginalItems.Select(x =>
           Regex.Replace(x.FileName, @"\sv\d+$", string.Empty)).ToArray();

  foreach(KeyValuePair<string,int> rule in fileRules){
    if(rule.Value - originalFileNames.Count(x => Regex.IsMatch(x, rule.Key,
                                              RegexOptions.IgnoreCase)) >0){
      return false;
    }
  }

  return true;
}
```

Ця функція отримує результат функції SpatialModel_Type. GetFileSetValidationRules та перевіряє перелік доданих оригінальних файлів на відповідність правилам перевірки. Під час завантаження файлів до системи Mine Advisor до назв файлів може додаватися версія, тому необхідно видалити версію з назви оригінального файлу для правильності перевірки.

Перевірка назв оригінальних файлів відбувається завдяки стандартним методам роботи з регулярними виразами у .Net[31].

Процес аналізу якості звітів відбувається за алгоритмом, розглянутим у розділі 2.3 (див. рисунок 2.3). Функції аналізу якості звітності наведено у додатку А.

Для збереження результатів аналізу у файлі Microsoft Excel використовується бібліотека Microsoft.Office.Interop.Excel[32]. Суть її роботи полягає у запуску екземпляру додатку Microsoft Excel та взаємодії з ним через спеціальні інтерфейси.

4.4 Оцінка повноти запропонованого рішення

Розроблена система виконує усі задачі та вирішує усі проблеми, що були виявлені під час аналізу предметної області (див. розділ 1.6).

Для цього було розроблено структуру бази даних, що дає змогу зберігати ієрархічну структуру гірничодобувного підприємства, типи робіт, що проводяться на структурних елементах однієї категорії, критерії оцінки якості звітності та звіти з переліком необхідних файлів.

Завдяки відсутності додаткових умов для створення структурних елементів підприємства та додавання їх до ієрархії, є можливість створювати дуже різноманітні структури, що мають декілька кореневих елементів та не мають обмежень розгалужування.

Наявність категорії у структурних елементів дає змогу визначити перелік робіт, що проводяться на однотипних структурних елементах підприємства.

Правила перевірки якості звітності налаштовуються для кожного типу робіт окремо, що дає змогу для більш точного аналізу якості. Визначений перелік періодичностей виконання робіт (див. розділ 2.2) є достатнім для вимог майже будь-якого підприємства. Оскільки правила перевірки переліку доданих файлів створюються з використанням регулярних виразів для назв файлів, є можливість

контролювати правильність найменування файлів звітності відповідно до внутрішніх правил підприємства.

Також до розробленої системи було додано можливість зберігати, окрім оригінальних файлів звітності, конвертовані файли, що є двомірним та трьохмірним зображенням структурних елементів підприємств або їх частин. Це дає змогу не тільки документально оцінити виконані роботи, а й побачити їх результат.

Можливість додавати конвертовані файли є результатом рішення розробки системи на базі додатку Mine Advisor. Це дало змогу конвертувати великий перелік форматів файлів.

Для збереження файлів, що додаються до звітів, використовується сховище, що є частиною системи Mine Advisor. Це дозволяє зберігати файли на власному сервері, завдяки чому є повний контроль над файлами, їх збереженням та передачею.

Розроблений алгоритм аналізу якості звітності про виконання завдань(див. розділ 2.3) є універсальним та може бути використаним у будь-якій сфері промисловості, де головними критеріями до звітності є своєчасність та повнота. За результатами роботи цього алгоритму можливо однозначно визначити структурні елементи та типи робіт, звітність з яких є неякісною та прийняти необхідні заходи з покращення якості праці виробничого персоналу.

Отже, запропоноване рішення є повним та може бути використано на підприємствах різних сфер, не тільки гірничодобувної.

ВИСНОВКИ

У процесі виконання атестаційної роботи було проаналізовано сучасний стан проблеми зберігання та аналізу якості звітності про виконання завдань виробничим персоналом. Як результат, було виявлено відсутність універсальних методів вирішення цієї проблеми, окрім декількох приватних рішень. Особлива увага у аналізі проблеми була приділена гірничодобувній промисловості, що є консервативною, а тому рідко запроваджує нові підходи вирішення проблем.

У межах атестаційної роботи було проаналізовано особливості структури гірничодобувного підприємства та критерії якості виконання завдань на гірничодобувному підприємства; сформульовано вимоги до розроблюваної системи, а саме до структури бази даних та методу аналізу якості звітності; розроблено структуру бази даних та метод аналізу якості звітності про виконання завдань за критерієм переліку необхідних файлів та терміном складання звіту. Створено програмну реалізацію запропонованого методу на базі системи Mine Advisor, що дозволило автоматизувати процес завантаження звітів до системи та аналіз якості звітності. Обґрунтовано вибір технологій для реалізації системи. Додано перевірки правильності введених користувачем даних для запобігання виникненню аварійних ситуацій та ситуацій з втратою даних.

Розроблене програмне забезпечення повністю вирішує проблеми та виконує задачі, що були сформовані під час аналізу предметної області.

Подальше розширення системи полягає у більшій автоматизації процесу аналізу якості звітності, бажано без участі людини, та створення можливості зазначення посади, до повноважень якої входить виконання конкретного виду робіт. Це дасть змогу більш точно визначати проблемні ситуації на підприємстві та зробить процес аналізу якості звітності регулярним.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Петенко И. В., Майдуков Г. Л., Петенко А. В. Проблемы стратегического выбора механизма инновационных преобразований в угольной отрасли Донбасса // [Електронний ресурс] – Режим доступу: https://cyberleninka.ru/article/n/problemy-strategicheskogo-vybora-mehanizma-innovatsionnyh-preobrazovaniy-v-ugolnoy-otrasli-donbassa.

2. Майдуков Г. Л. К вопросу о выборе стратегии инновационных преобразований в угольной отрасли // [Електронний ресурс] – Режим доступу: https://cyberleninka.ru/article/n/k-voprosu-o-vybore-strategii-innovatsionnyh-preobrazovaniy-v-ugolnoy-otrasli.

3. Заернюк В.М., Борисович В.Т. Инвестиционный потенциал драгоценных металлов: современное состояние и перспективы // [Електронний ресурс] – Режим доступу: https://doi.Org/10.24891/fa.11.4.454.

4. Рубцов Н.Н. Потребление золота в мире // [Електронний ресурс] – Режим доступу: https://cyberleninka.ru/article/n/potreblenie-zolota-v-mire/viewer.

5. Череватский Д.Ю. Промышленная политика для угольной промышленности // [Електронний ресурс] – Режим доступу: https://cyberleninka.ru/article/n/promyshlennaya-politika-dlya-ugolnoy-promyshlennosti.

6. В.Б. Кондратьев. Роль горной промышленности в экономике // [Електронний ресурс] – Режим доступу: https://cyberleninka.ru/article/n/rol-gornoy-promyshlennosti-v-ekonomike/viewer.

7. Кочура И.В. Анализ развития экономического потенциала угольной промышленности донбасса в современных условиях хозяйствования // [Електронний ресурс] – Режим доступу: https://cyberleninka.ru/article/n/analiz-razvitiya-ekonomicheskogo-potentsiala-ugolnoy-promyshlennosti-donbassa-v-sovremennyh-usloviyah-hozyaystvovaniya.

8. Михайло Кухар. Добувна промисловість може принести Україні додатково $9-22 млрд // [Електронний ресурс] – Режим доступу:

https://gmk.center/ua/opinion/dobuvna-promislovist-mozhe-prinesti-ukraini-dodatkovo-9-22-mlrd/.

9. Александр Казмиришин. Горнодобывающая промышленность – ключ к развитию украины // [Електронний ресурс] – Режим доступу: https://site.ua/kazma/14523/.

10. Соловьёва О.В., Кузьминов В.В., Ганцева Л.В. Анализ факторов, влияющих на эффективность работы персонала // [Електронний ресурс] – Режим доступу: https://cyberleninka.ru/article/n/analiz-faktorov-vliyayuschih-na-effektivnost-raboty-personala/viewer.

11. А. Ю. Месяц. Повышение производительности труда: системные поход // [Електронний ресурс] – Режим доступу: https://cyberleninka.ru/article/n/povyshenie-proizvoditelnosti-truda-sistemnyy-podhod/viewer.

12. Основы экономики и управления горным предприятием. учебное пособие / Должніков П.Н., Величко Н.М., Должникова А.П. – Донецьк: «Норд-пресс». – 2009. – 200 с.

13. Технология проведения горно-разведочных выработок: учебник / Лук'янов В.Г., Панкратов А.В., Шмуригін В.А. – Томск: Вид-во Томського політехнічного університету, 2011. – 550с. – ISBN 978-5-98298-574-3.

14. Горное дело: учебник для техникумов / Шехурдін В.К., Несмотряєв В.І., Федоренко П.І. – Москва: «Надра», 1987. – 440с.

15. Сервер или облачное решение // Xelent: веб-сайт. URL: https://www.xelent.ru/blog/server-ili-oblachnoe-reshenie/ (дата звернення 30.10.2020).

16. Скан-Архив: Электронный архив документов в 1С // Скан-Архив: веб-сайт. URL: http://scan-archive.ru/ (дата звернення 30.10.2020).

17. Jira // Вікіпедія: веб-сайт. URL: https://uk.wikipedia.org/wiki/Atlassian_JIRA (дата звернення 30.10.2020).

18. 10 современных планировщиков задач// ITC.ua: веб-сайт. URL: https://itc.ua/articles/10-sovremennyh-planirovshhikov-zadach/ (дата звернення 30.10.2020).

19. Введение в системы баз данных, 8-е издание / Дейт, К. Дж. – Москва: «Вильямс», 2005. – 1328с. – ISBN 5-8459-0788-8 .

20. Довідка канадського патентного центру про реєстрацію торгової марки Mine Advisor // [Електронний ресурс] – Режим доступу: file:///C:/Users/Lvbnh/Google Drive/Диплом магистра/литература/Mine Advisor TM certificate.pdf.

21. Mine Advisor – Руководство пользователя // [Електронний ресурс] – Режим доступу: file:///C:/Users/Lvbnh/Google Drive/Диплом магистра/литература/Mine Advisor - Руководство пользователя.pdf

22. Проектирование, разработка и сопровождение баз даных с использованием CASE-средств / Пілецький І.І. – Мінськ: БДУІР, 2009. – 116с. – ISNB 978-985-488-324-3 .

23. Программирование баз данных Microsoft SQL Server 2008. Базовый курс / Виейра Р. – Київ: Діалектика, 2010. – 816с. – ISBN 978-5-8459-1612-9, 978-0-470-25701-2.

24. WPF 4: Windows Presentation Foundation в .NET 4.0 с примерами на C# 2010 для профессионалов / Метью Мак-Дональд. – Москва: ООО «И.Д. Вильямс». 2011 – 1024с. – ISBN 978-5-8459-1657-0.

25. C# 7.0. Карманный справочник / Албахарі Дж., Албахарі Б. – Санкт-Петербург: ООО «Альфа-книга». 2018. – 224с. – ISBN 978-5-9909446-1-9.

26. LINQ to SQL // Microsoft Docs: веб-сайт. URL: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/linq/ (дата звернення 13.11.2020).

27. Entity Framework 6 // Microsoft Docs: веб-сайт. URL: https://docs.microsoft.com/en-us/ef/ef6/ (дата звернення 13.11.2020).

28. Welcome to Telerik UI for WPF // Telerik Docs: веб-сайт. URL: https://docs.telerik.com/devtools/wpf/introduction (дата звернення 13.11.2020).

29. Create Data Transfer Objects (DTOs) // Microsoft Docs: веб-сайт. URL: https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5 (дата звернення 22.11.2020).

30. “await anything;” by Stephen Toub // Microsoft Developer Blogs: веб-сайт. URL: https://devblogs.microsoft.com/pfxteam/await-anything/ (дата звернення 23.11.2020).

31. .NET regular expressions // Microsoft Docs: веб-сайт. URL: https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expressions (дата звернення 01.12.2020).

32. Microsoft.Office.Interop.Excel Namespace // Microsoft Docs: веб-сайт. URL: https://docs.microsoft.com/ru-ru/dotnet/api/microsoft.office.interop.excel?view=excel-pia (дата звернення 01.12.2020).