РЕФЕРАТ

Темою дипломного проекту є «Програма «Електронний освітній ресурс». Програмний модуль інформаційного забезпечення навчально-виховного процесу».

Дипломний проект: 53 рисунки, 2 таблиці, 8 лістингів, 11 джерел та 5 додатків.

Об’єкт дослідження: Програмний модуль інформаційного забезпечення навчально-виховного процесу.

Мета роботи — об’єднання, зберігання та автоматизація взаємодії з даними, які будуть користі та якими зможуть користуватися викладачі та студенти.

Постановка задачі 8 сторінок. Інформація про вимоги до використовуваних технічних засобів, опис інструментальних засобів, що використовувались при розробці програми.

Опис етапів реалізації 19 сторінок. Докладна інформація про розроблення та експлуатацію програми.

Опис програмного продукту 15 сторінок. Докладна інструкція користувача для роботи з програмним модулем та опис деяких ситуацій, які перешкоджають логічним помилкам, які можуть призвести до порушення нормальної роботи програми.

Економічна частина 3 сторінок. Економічний розрахунок вартості програмного продукту.

Охорона праці 7 сторінок. Аналіз небезпечних та шкідливих чинників, які впливають на здоров’я людини на робочому місці за комп’ютером; розрахунок захисту від шкідливих виробничих факторів; заходи по пожежній безпеці та заходи з ергономіки на робочому місці.

В даному дипломному проекті засобами середовища Microsoft Visual Studio 2017, розроблено програмний модуль, який призначений для полегшення роботи викладачів та підвищення освітнього рівня у студентів.

Ключові слова: *програмний продукт, модуль, форма, компонент, інформація, параметричний пошук, дерево даних, вузол, файл, документ, розширення, налаштування, запит, база даних, MySQL, Visual Studio*.

ЗМІСТ

[ВСТУП 3](#_Toc11428873)

[1 ПОСТАНОВКА ЗАВДАННЯ 4](#_Toc11428874)

[1.1 Технічне завдання на розробку програмного продукту 4](#_Toc11428875)

[1.2 Огляд існуючих рішень 8](#_Toc11428883)

[1.3 Обґрунтування вибору мови програмування 9](#_Toc11428884)

[2 ОПИС ЕТАПІВ РЕАЛІЗАЦІЇ 12](#_Toc11428888)

[2.1 Проектування бази даних 12](#_Toc11428889)

[2.2 Діаграма сценаріїв 14](#_Toc11428890)

[2.3 Проектування інтерфейсу та опис складових частин програми 15](#_Toc11428891)

[2.4 Опис основних алгоритмів програми 23](#_Toc11428952)

[3 ОПИС ПРОГРАМНОГО ПРОДУКТУ 31](#_Toc11428972)

[4 АНАЛІЗ ДОСЛІДНОЇ ЕКСПЛУАТАЦІЇ 46](#_Toc11428976)

[5 ОХОРОНА ПРАЦІ 48](#_Toc11428977)

[5.1 Аналіз небезпечних і шкідливих виробничих чинників проектованого технологічного процесу, об’єкту, системи або пристрою 48](#_Toc11428978)

[5.2 Інженерно-технічні заходи з охорони праці 49](#_Toc11428979)

[5.3 Пожежна профілактика 52](#_Toc11428980)

[5.4 Заходи з ергономіки 54](#_Toc11428981)

[6 ЕКОНОМІЧНА ЧАСТИНА 56](#_Toc11428982)

[ВИСНОВКИ 59](#_Toc11428983)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60](#_Toc11428984)

[Додаток А 61](#_Toc11428987)

[Додаток Б 63](#_Toc11428988)

[Додаток В 64](#_Toc11428994)

[Додаток Г 66](#_Toc11428999)

[Додаток Ґ 67](#_Toc11429002)

ВСТУП

*3*

Мета дипломної роботи полягає в тому, щоб поліпшити та закріпити взаємодію між викладачами та студентами, а саме: доволі часто у студентів з’являлося питання до викладачів стосовно лабораторних робіт, літератури, графіку екзаменів, розподілу балів тощо. Дана програма має допомогти оптимізувати та полегшити цю або подобні ситуації.

Також, таким чином має підвищитися загальний рівень успішності у студентів, бо за допомогою даного програмного продукту студенти зможуть ознайомлюватися з майбутніми темами та дисциплінами, правильно розподіляти час та вчасно готуватися до екзаменів, перевіряти поточні оцінки або переглядати поточні лабораторні роботи.

Програмний продукт включатиме в себе наступні рівні доступу:

* адміністратор (повний доступ до даних);
* викладач (може додавати, редагувати, видаляти власні документи та переглядати усі документи);
* студент (може переглядати усі документи).

Функціонально програмний продукт повинен:

* включати в собі базу даних предметної області;
* можливість переглядати всі документи;
* можливість налаштовувати відображення інформації;
* виконувати різний пошук документів: з параметрами, по ключовим словам;
* можливість налаштовувати пошук даних;
* можливість переглядати зміст документів;
* можливість редагувати інформацію про документів;
* можливість зберігати документи.

1. ПОСТАНОВКА ЗАВДАННЯ
   1. Технічне завдання на розробку програмного продукту
      1. Найменування розробки

Темою дипломного проекту є «Програма «Електронний освітній ресурс». Програмний модуль інформаційного забезпечення навчально-виховного процесу».

* + 1. Підстава для розробки

Підставою для розробки програми дипломного проекту є виникнення технологічної необхідності автоматизації певних процесів у взаємодії між викладачами та студентами.

* + 1. Призначення розробки

Розробка програми дипломного проекту призначена для об’єднання, зберігання та автоматизації ведення подібних документів, які будуть користі та якими зможуть користуватися викладачі та студенти. Таким чином має підвищитися загальний рівень успішності у студентів та полегшити певним чином роботу викладачам.

* + 1. Вимоги до програмного продукту, що розробляється

В рамках даної теми було створено базу даних для збереження:

* інформації про типи матеріалів;
* інформації про розширення документів;
* даних про документи;
* інформації для пошуку по ключовим словам.

Для реалізації програмного продукту було передбачено збереження наступних даних:

1. Для детальної інформації про документи, необхідно вести інформацію про типи матеріалів, таким чином про типи матеріалів зберігається наступна інформація: найменування типу матеріалу.
2. Кожний документ має розширення, тому для автоматизації необхідно зберігати інформацію про розширення документів.
3. Дані про документи:

* викладач;
* тип дисципліни;
* типу матеріалу;
* найменування документу;
* ключові слова.

Програмний додаток має містити перегляд усіх документів у зручній формі, а також певне меню налагодження, у якому можна налаштувати певні параметри виведення інформації та пошуку.

Виведення інформації передбачається у вигляді дерева (аналог провіднику в операційній системі Windows). Дане дерево має складатися з чотирьох рівнів, кожний з яких має включати в себе однотипну інформацію:

* перший рівень — дисципліни;
* другий рівень — викладачі;
* третій рівень — матеріали (типи документів);
* четвертий рівень — документи.

Передбачається декілька типів пошуку документів:

* пошук за ключовими словами;
* параметричний пошук.

Для пошуку за ключовими словами необхідно вести інформацію про кожне ключове слово.

1. Про ключові слова зберігається наступна інформація: перелік ключових слів.

Параметричний пошук, має включати в себе декілька параметрів, які можна комбінувати між собою, таким чином щоб користувач знайшов потрібну інформацію як найшвидше.

* + 1. Вимоги до програмного та апаратного забезпечення на етапі експлуатації

Рекомендовані вимоги до програмного та апаратного забезпечення сервера:

* можливість підтримки бази даних MySQL;
* операційна система: Microsoft Windows 7 або вища;
* об'єм жорсткого диску від 1 Гб;
* система керування базами даних MySQL 5.6;
* мережева папка зі спільним доступом;
* процесор з тактовою частотою 1 ГГц або вище;
* оперативний запам’ятовувальний пристрій об’ємом 2 Гб або більше.

Рекомендованими вимогами до програмного забезпечення клієнта програми є:

* наявність пристрою з операційною системою MS Windows 7 або вище;
* наявність стеку програм Microsoft Office 2007 або вище;
* процесор з тактовою частотою 1 ГГц або вище;
* оперативний запам’ятовувальний пристрій об’ємом 2 Гб або більше;
* 3 Гб вільного місця на жорсткому диску.
  + 1. Вимоги до програмної документації

Основними документами, що регламентують розробку програм, повинні бути документи Єдиної системи програмної документації (ЄСПД): керівництво користувача, керівництво програміста, опис застосування.

* + 1. Календарний план робіт

Календарний план робіт було створено за допомогою програми — Microsoft Office Project.

Календарний план робіт представлений у вигляді діаграми Ганта. Також до діаграми включається перелік завдань, терміни виконання завдань та послідовність виконання, таким чином максимально детально описується весь процес реалізації програмного модуля.

Для зручності користування та покращення зовнішнього вигляду всі етапи проектування та реалізації дипломної роботи було розбито на декілька груп.

Також слід зазначити, що у діаграмі Ганта є можливість зв’язувати завдання між собою за допомогою наступних з’єднань:

* закінчення-початок;
* початок-початок;
* початок-закінчення;
* закінчення-закінчення.

Таким чином усе перераховане дає змогу ретельно контролювати увесь процес виробництва проекту та допомагає правильно організувати та розподілити час.

Переглянути діаграму Ганта можна у додатку А.

* 1. Огляд існуючих рішень

Аналогом програми дипломного проекту є UniproRetail . За допомогою цього додатку забезпечується:

* гарантована наявність всіх необхідних документів, звітів, довідників для ведення оперативного обліку;
* коробковий продукт «все в одному», всі модулі з максимальною функціональністю включені у вартість;
* система паролів і права рівня доступу користувачів до об'єктів програми;
* системний журнал всіх дій користувачів ;
* робота програми в мережевому режимі;
* імпорт довідників товарів, а також документів з Excel;
* робота програми UniproRetail без наявності інтернету;
* швидка і легка інсталяція, мінімальні вимоги до системи;
* безкоштовні необмежені в часі оновлення програми.

Професійна програма UniproRetail для обліку документів, у сфері HoReCa, на складських об'єктах. Крім цього є можливість завантажити повнофункціональну демо-версію програми UniproRetail доступ на 14 днів.

Ще один аналог дипломної роботи від компанії — AraX Group. За допомогою цього додатку забезпечується:

* облік вхідної, вихідної кореспонденції і внутрішніх документів (наказів, розпоряджень і т.д.) організації;
* можливість роботи декількох організацій;
* можливість автоматичного формування номеру документа;
* сповіщення по терміну виконання для відповідальних виконавців;
* друк журналів, публікація в MS Word (OO Writer);
* картка документа, експорт картки в MS Word і OO Writer, можливість формування картки по шаблону;
* можливість вставки зображення документа, можливість одержання зображень документів безпосередньо зі сканера;
* пошук по журналах;
* можливість публікації в MS Excel і OO Calc обраних полів результату пошуку;
* довідники. Звіти;
* можливість налаштовування прав доступу кожного користувача до журналів документів;
* користувачеві можна призначити режим «Бачити тільки свої документи»;
* програма підтримує режим роботи в локальній мережі.
  1. Обґрунтування вибору мови програмування

Для реалізації програмного продукту було використано середовище Visual Studio, програмний продукт написаний на мові програмування C#, СУБД було використано MySQL.

* + 1. Опис середовища розробки Visual Studio

Для реалізації даного програмного продукту було обрано інтегроване середовище розробки програмного забезпечення від фірми Microsoft — Visual Studio.

Дане середовище дозволяє створювати різноманітні програмні продукти: консольні програми, програми з графічним інтерфейсом, наприклад віконні додатки Windows Forms, а також Web-додатки тощо.

Середовище розробки Visual Studio має багатовіконний інтерфейс, який можна налаштовувати під потреби конкретного користувача.

Якщо потрібна панель не відображається у вікні Visual Studio, її з легкістю можна викликати, скориставшись відповідним пунктом головного меню «Вид».

Для переміщення панелі в іншу область екрану потрібно скористатися схемою розташування, яка з’являється одразу після початку переміщення.

В панелі «Свойства» властивості компонентів можуть бути згруповані по категоріям або відображатися єдиним списком, впорядкованим за алфавітом.

Visual Studio включає один або декілька з наступних компонентів:

* Visual Basic .NET, а до його появи — Visual Basic;
* Visual C++;
* Visual C#;
* Visual F# (входить до складу Visual Studio 2010);
* Visual Studio Debugger.

Visual Studio допомагає при написанні коду, незалежно від мови, від C #, VB і C ++ до JavaScript і Python, надаючи допомогу в реальному часі.

Список помилок — це все що потрібно для переходу до помилок коду в рішенні і їх виправлення, незалежно від джерела: від компіляції і збірки до проблем аналізу коду. У деяких мовах підтримуються динамічні призначені для користувача аналізатори, що дозволяють виявляти проблеми домену під час введення.

Visual Studio надає прекрасні можливості налагодження незалежно від мови, яку було обрано: від C #, VB і C ++ до JavaScript і Python, XAML і HTML.

За необхідністю можна зупинити виконання програми у будь-якому місці коду, Visual Studio запропонує безліч способів перевірити значення змінних для формулювання або перевірки гіпотези.

Також середовище Visual Studio підтримує покрокове виконання коду, швидко переглядати локальні змінні і оцінювати складні вирази — і все це не виходячи з відладчика.

Також є змога налаштувати оформлення додатка на будь-який смак, поки програма виконується. Більше не потрібно покладатися на дані часу розробки, щоб перевірити макет.

Для додатків з призначеним для користувача інтерфейсом на HTML або XAML проблеми на рівні інтерфейсу можна виявляти, використовуючи функції налагодження коду. Крім того, Visual Studio дозволяє переглядати HTML DOM і візуалізувати структуру коду XAML за допомогою візуального дерева XAML прямо під час виконання програми [1].

* + 1. Опис мови програмування C#

Для реалізації даного програмного комплексу було обрано мову програмування — C#. Тому що на сьогоднішній день мова програмування C# — одна з найпотужніших, яка швидко розвиваються і дуже затребувана в ІТ-галузі. На даний момент на мові C# пишуться найрізноманітніші програми: від невеликих десктопних програмок до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

C# є мовою з Сі-подібним синтаксисом і близький в цьому відношенні до C++ і Java. Тому, якщо користувач знайомий із однією з цих мов, то опанувати C # буде легше.

C# є об'єктно-орієнтованим і в цьому плані багато було перейнято з Java і С++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. Слід зазначити, що C# продовжує активно розвиватися, і з кожною новою версією з'являється все більше цікавих функціональних рішень [2].

* + 1. Опис СУБД MySQL

Для збереження інформації було обрано СУБД MySQL. Було обрано саме цю СУБД за її зручність підключення до проектів, виконання запитів та використання команд MySQL, які виконують дуже велику роль в оптимізуванні програми.

Функціонал даного СУБД чудово підходить для реалізації даного проекту, бо має все необхідне для запису інформації, обробки інформації та отримання інформації.

СУБД MySQL підтримує реляційні бази даних, це означає що данні зберігаються не всі разом, а в окремих таблицях, завдяки чому добиратися до даних стає швидше і зручніше. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів.

Для більш приємного користування базою даних було використано MySQL Workbench. MySQL Workbench — являється візуальною оболонкою СУБД MySQL, за допомогою якого користуватися базою даних стає в рази приємніше. MySQL Workbench включає в себе всі можливості, які надає MySQL наприклад: створення баз даних, створення таблиць, редагування таблиць, об’єднання таблиць, виконання запитів, додання нової інформації до таблиць тощо.

1. ОПИС ЕТАПІВ РЕАЛІЗАЦІЇ

* 1. Проектування бази даних

Для реалізації дипломного проекту була обрана СУБД MySQL 5.5, у якій створена база даних, що містить дев'ять таблиць. Серед загальної кількості таблиць, для реалізації індивідуального завдання, було створено п’ять таблиць, серед яких знаходяться основні та допоміжні таблиці для нормалізації бази даних, а саме:

Таблиця «Materials» зв’язує між собою більшість таблиць у базі даних, в даній таблиці зберігається інформація про документи та все що з ними пов’язано. Дана таблиця зв’язана з наступними таблицями: «Studyload», «Document\_specification», «Material\_tag», «Type». Структура таблиці «Materials» представлена на рисунку 2.1.

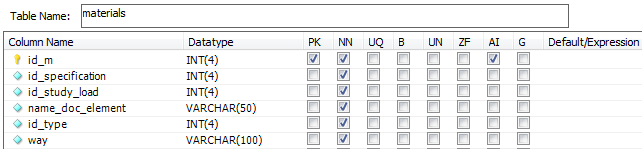


Рисунок 2.1 – Структура таблиці «materials»

Таблиця «Document\_specification» розбиває обліковий атрибут у таблиці «Materials», в даній таблиці зберігається інформація про типи матеріалів, іншими словами типи документів, необхідних для детальнішого представлення інформації про документи. Структура таблиці «Document\_specification» представлена на рисунку 2.2.

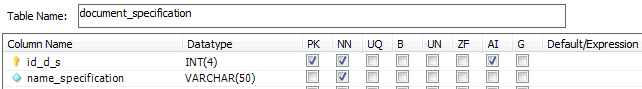


Рисунок 2.2 – Структура таблиці «document\_specification»

Таблиця «Type» розбиває обліковий атрибут у таблиці «Materials», в даній таблиці зберігається інформація про розширення матеріалів, тобто тип документу який, як правило, йде після назви документу, необхідних для детальнішого представлення інформації про документ. Структура таблиці «Type» представлена на рисунку 2.3.

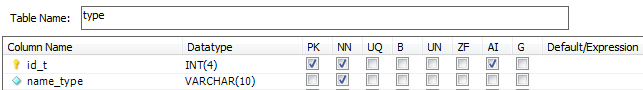


Рисунок 2.3 – Структура таблиці «type»

Таблиця «Material\_tag» створена для розбиття відношення — багато до багатьох між таблицями «Materials» та «Tag». Структура таблиці «Material\_tag» представлена на рисунку 2.4.

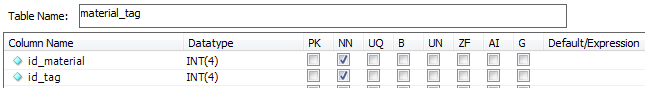


Рисунок Б.4 – Структура таблиці «material\_tag»

Таблиця «Tag» створена для збереження ключових слів, за допомогою яких реалізовується пошук у програмі. Дана таблиця пов’язана з таблицею «Material\_tag». Структура таблиці «Tag» представлена на рисунку 2.5.

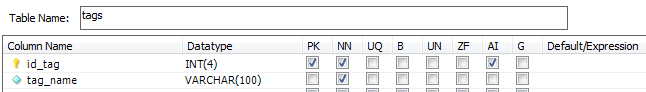


Рисунок 2.5 – Структура таблиці «tags»

Загальна схема таблиць та зв’язки між ними продемонстровано на рисунку 2.6. Заповнені таблиці даними можна побачити у додатку Б.

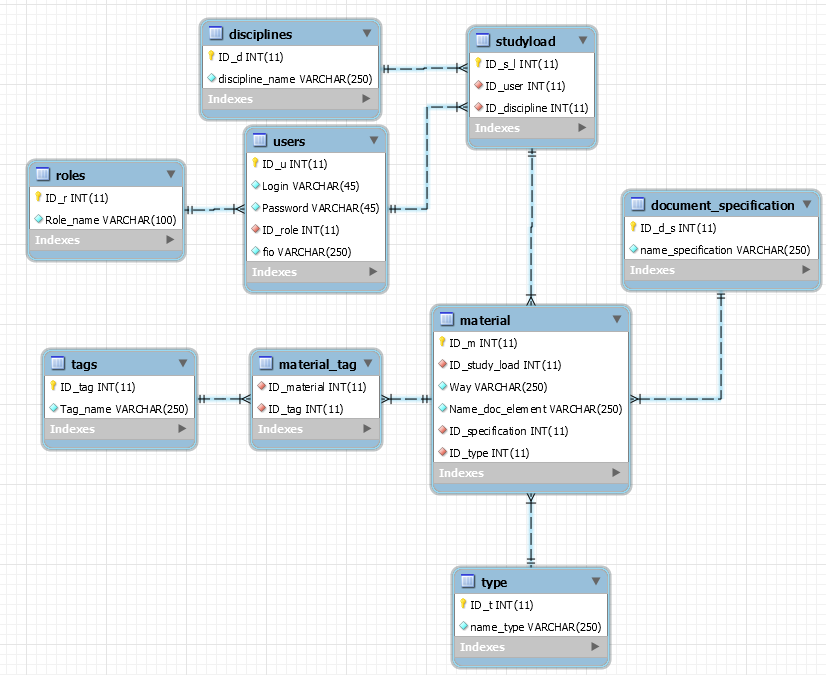


Рисунок 2.6 – Загальна схема бази даних

* 1. Діаграма сценаріїв

Діаграма сценаріїв — демонструє розподіл функціоналу, який припадає на кожного користувача програми, тим самим чудово демонструє рівні доступу до програми. У дипломній роботі передбачено три типи авторів, на яких припадає різний обсяг функціоналу:

* адміністратор;
* викладач;
* студент.

Автор «Адміністратор» — має найбільший рівень доступу до програми, відповідно на нього припадає весь обсяг функціоналу, має повний доступ до даних про викладачів, може контролювати всі дані про документи, не зважаючи яким викладачем вони були додані.

Автор «Викладач» — має середній рівень доступу до програми, має повний доступ до документів, які додав саме він, тобто не може змінювати або видаляти документи, які додав інший викладач.

Автор «Студент» — має найменший рівень доступу до програми, відповідно на нього припадає найменше функціонала, може переглядати документи.

Якщо актори визначають зовнішню сутність системи, то внутрішню відображають сценарії. Розглядаючи окремі сценарії, можна розділити функціональність системи на окремі складові, з якими можливо працювати паралельно.

Для сценаріїв визначають два типи відношень:

* відношення «розширює» означає те, що функції одного сценарію є доповненнями до функцій іншого сценарію;
* відношення «використовує» означає те, що конкретний сценарій може бути використаний для розширення кількома іншими сценаріями.

Діаграму сценаріїв програмної системи дипломного проекту можна побачити у додатку В.

* 1. Проектування інтерфейсу та опис складових частин програми

Графічний інтерфейс забезпечує взаємодію користувача з програмним додатком. Користувацький інтерфейс модуля «Фільтри» складається з декількох форм. Схема зв’язку модулів наведена у додатку Г.

Перша форма, яку користувач може бачити є головною, бо через дану форму користувач може потрапити до інших форм модуля, також дана форма включає в себе найбільшу кількість компонентів та взагалі функціоналу, а саме дана форма забезпечує:

* виведення інформації про всі документи, що знаходяться у базі даних;
* зображені документи мають бути представлені у вигляді дерева файлів або документів;
* параметричний пошук документів;
* пошук документів за допомогою ключових слів;
* меню налаштування параметричного пошуку;
* меню налаштування зовнішнього вигляду дерева файлів;
* виведення загальної інформації про кількість файлів та документів.

На формі знаходяться компоненти, до кожного компонента форми було накладено певний, вище перерахований, функціонал. Переглянути зовнішній вигляд форми можна на рисунках 2.7–2.9.

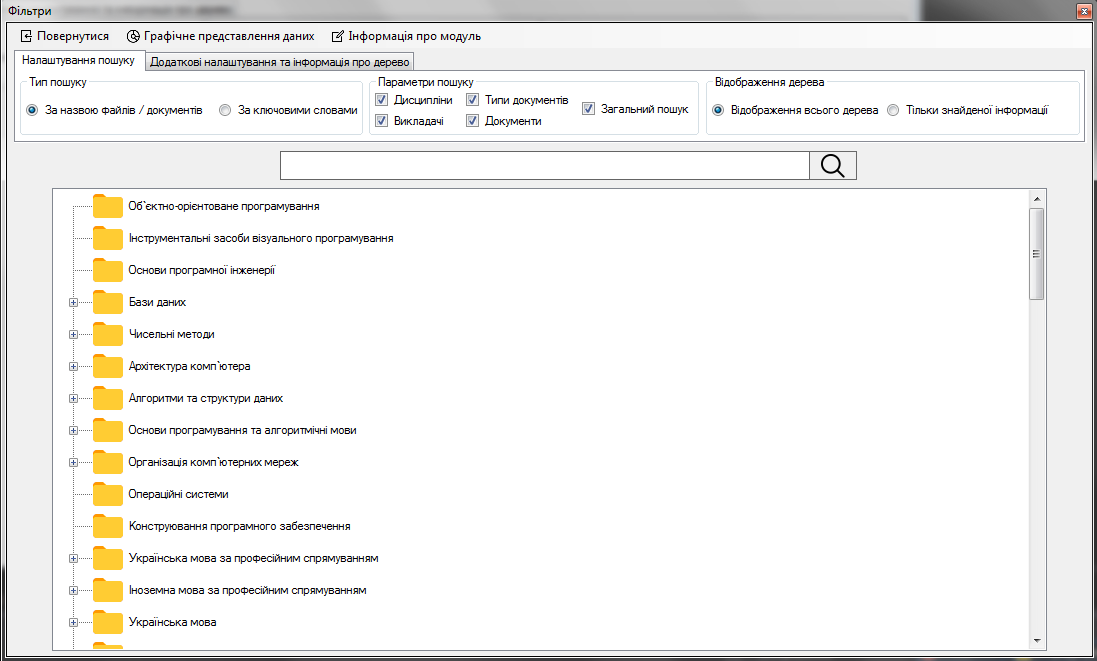


Рисунок 2.7 – Зовнішній вигляд форми

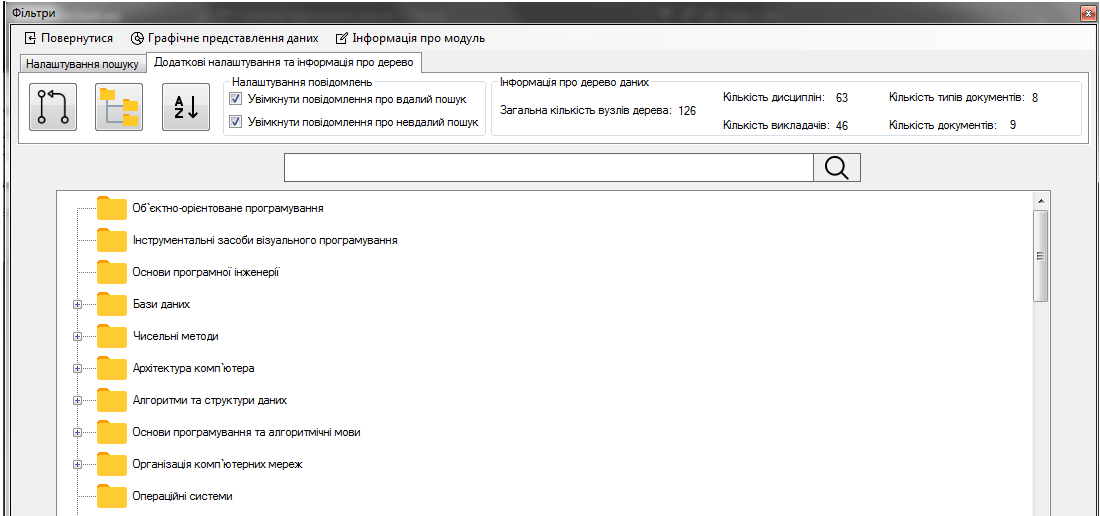


Рисунок 2.8 – Зовнішній вигляд другої вкладники компоненту TabControl

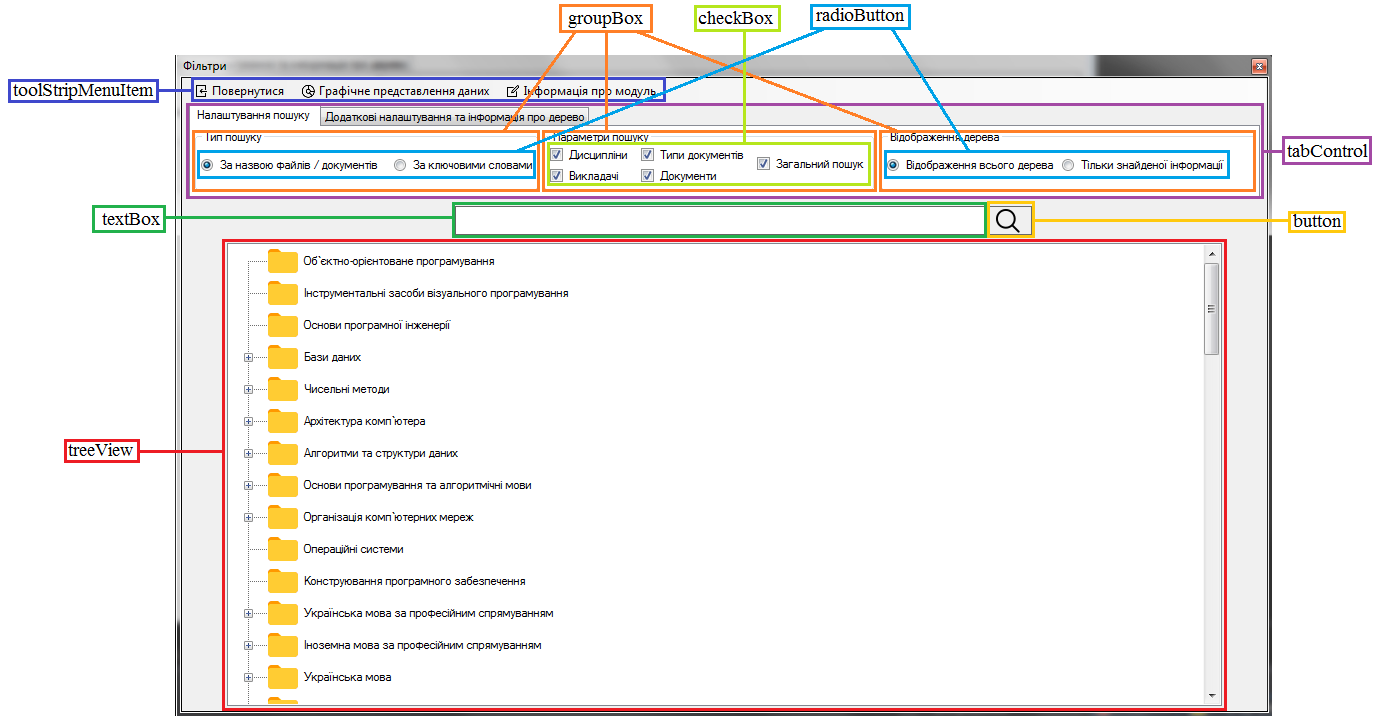


Рисунок 2.9 – Компоненти на формі

Перелік та характеристика компонентів на формі:

* treeView — головний компонент на формі бо всі інші форми тим, чи іншим чином зв’язані з даним компонентом. Даний компонент виводить інформацію, що знаходиться у базі даних у формі дерева даних. Дерево даних може складатися з чотирьох рівнів (дисципліни, викладачі, типи документів, документи);
* textbox — за допомогою даного компоненту виконується пошук. До даного компоненту вводиться текст, по якому буде виконуватися пошук;
* button — при натисканні даного компоненту виконується пошук даних, по тексту який було введено до компоненту textbox;
* tabControl — компонент-контейнер, у якому можуть знаходитися інші компоненти. У компонента є дві вкладники, на яких розміщується компоненти з наступний функціонал: меню налаштування зовнішнього вигляду дерева інформації, меню налаштування пошуку за параметрами, меню зніти типу пошуку, перегляд інформації про дерево даних;
* groupBox — компонент-контейнер, який групує однотипні компоненти між собою;
* radioButton — компонент-перемикач, призначений для встановлення парламентів пошуку, відображення зовнішнього вигляду дерева даних;
* comboBox — компонент-перемикач, призначений для налаштування пошуку з параметрами;
* toolStripMenuItem — компонент-меню, за допомогою даного компоненту користувач може перейти до інших форм модулю;
* imageList — даний компонент створює список зображень, таким чином прив’язувати зображення до компонентів стає легше.

Слід зазначити що компонент treeView включає в себе не притаманні іншим компонентам методи та параметри, за допомогою яких було реалізовано виведення даних у зручному форматі (див. рис. 2.4). Зображення даних було реалізовано за допомогою наступних методів та параметрів:

* GetNodeCount(передається значення логічного типу) — даний метод повертає ціле значення про кількість дочірніх вузлів даного вузла;
* IsExpanded — даний параметр перевіряє стан вузла, стан вузла може бути або відкритим, або закритим. Даний метод повертає значення типу bool, а саме: true, якщо даний вузол відкритий, false — якщо вузол закритий;
* Toggle() — даний метод відкриває або закриває вузли дерева, тобто робить протилежний стан вузла.
* Remove() — видаляє вузол дерева;
* Add(передається значення рядкового типу) — створює вузол дерева;
* treeView1.SelectedNode — параметр повертає шлях до обраного вузла дерева;
* CollapseAll() — метод закриває усі вузли дерева;
* BeginUpdate() — вмикає редагування дерева, тобто якщо необхідно додати, відредагувати або видалити вузли дерева, спочатку необхідно використати даний метод, після чого можна вносити зміни до дерева даних, після зміни дерева необхідно використати метод для вимикання даного режиму;
* EndUpdate() — вимикає редагування дерева;
* IsSelected — даний параметр перевіряє чи було обраний вузол дерева, повертає значення логічного типу, а саме: true — якщо даний вузол було обрано, «false» — якщо вузол не обраний.

За допомогою наступної форми, користувач може ознайомитися з інформацією про викладачів, а саме:

* кількість дисциплін, які веде кожен викладач;
* кількість дисциплін, які було викладено кожним викладачем;
* графічне зображення, яке демонструє кількість дисциплін у кожного викладача;
* графічне зображення, яке демонструє кількість документів, які було викладено кожним викладачем;
* можливість обирати тип графіків.

Зовнішній вигляд форми можна переглянути на рисунку 2.10. Компоненти, які розташовані на формі можна переглянути на рисунку 2.11.

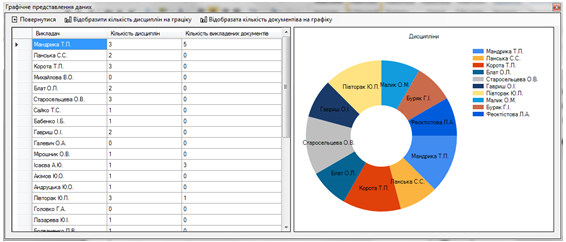


Рисунок 2.10 – Зовнішній вигляд форми

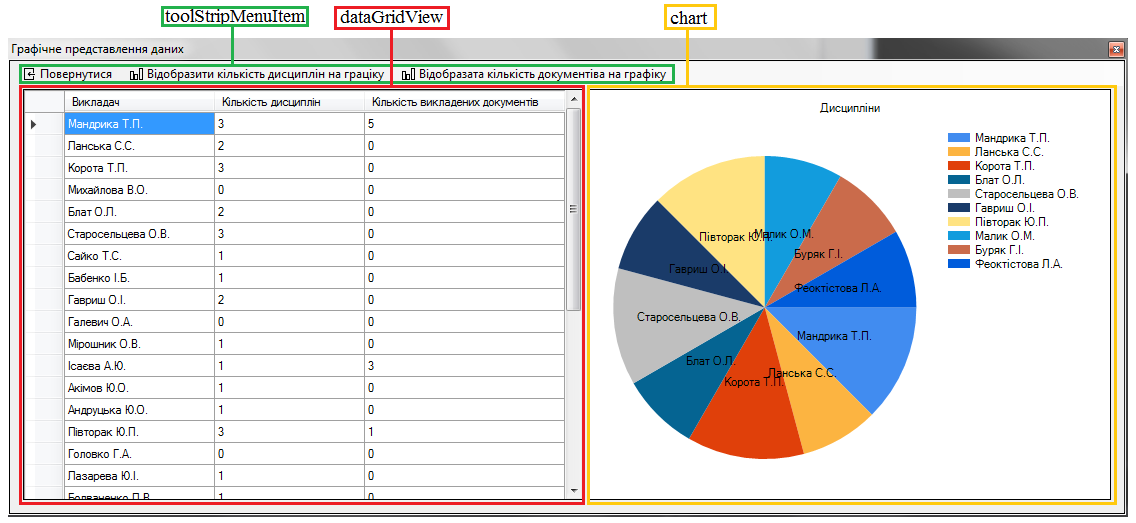


Рисунок 2.11 – Компоненти на формі

Перелік та характеристика компонентів на формі:

* toolStripMenuItem — компонент-меню, за допомогою даного компоненту користувач може перейти до головної форм модулю, також за допомогою даного компоненту користувач може змінювати типи та параметри графіку;
* dataGridView — компонент-таблиця, до якого заноситься інформація про викладачів, дисципліни та документи;
* chart — компонент виводить інформацію у вигляді графіків, які можна за бажанням змінювати.

Використанні методи та параметри компонентів:

* ChartType — параметр відповідає за тип графіку, використовується з компонентом chart;
* Rows.Add(приймає список параметрів рядкового типу) — за допомогою даного методу додається запис до таблиці, використовується з компонентом «dataGridView»;
* Points.AddXY(приймає перший аргумент — рядковий тип, другий — цілий) — додає запис до колеції, використовується з компонентом chart.

Третя форма відповідає за кратку але потрібну інформацію про програмний модуль, включає в себе лише декілька компонентів, переглянути зовнішній вигляд можна на рисунку 2.12, перелік компонентів на рисунку 2.13.

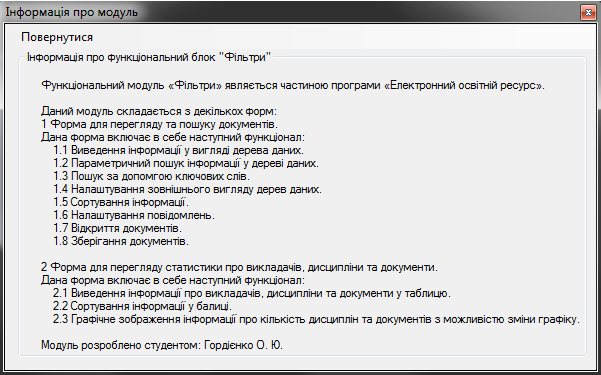


Рисунок 2.12 – Зовнішній вигляд форми

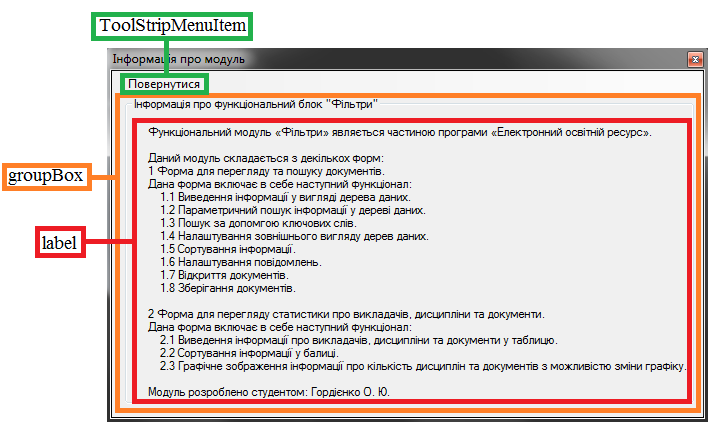


Рисунок 2.13 – Компоненти на формі

Перелік та характеристика компонентів на формі:

* toolStripMenuItem — компонент-меню, за допомогою даного компоненту користувач може повернутися до головної форм модулю;
* groupBox — компонент-контейнер, необхідний для групування компонентів, в даному випадку призначений для додаткового повідомлення та поліпшення зовнішнього вигляду форми;
* label — компонент у якому знаходиться кратка інформація про функціонал інших форм та інформація про розробника модуля.

У програмному модулі присутня ще одна форма «Редагування». Дана форма активується після обрання викладачем або адміністратором пункту редагування у контекстному меню документа. Зовнішній вигляд форми можна побачити на рисунку 2.14, компоненти на формі можна переглянути на рисунку 2.15.

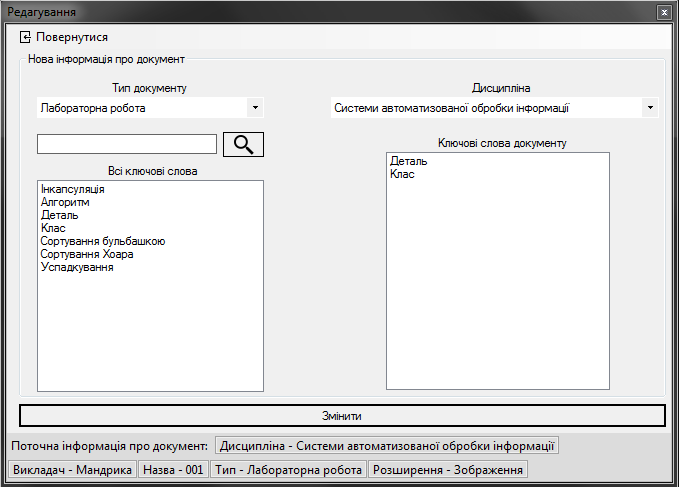


Рисунок 2.14 – Зовнішній вигляд форми

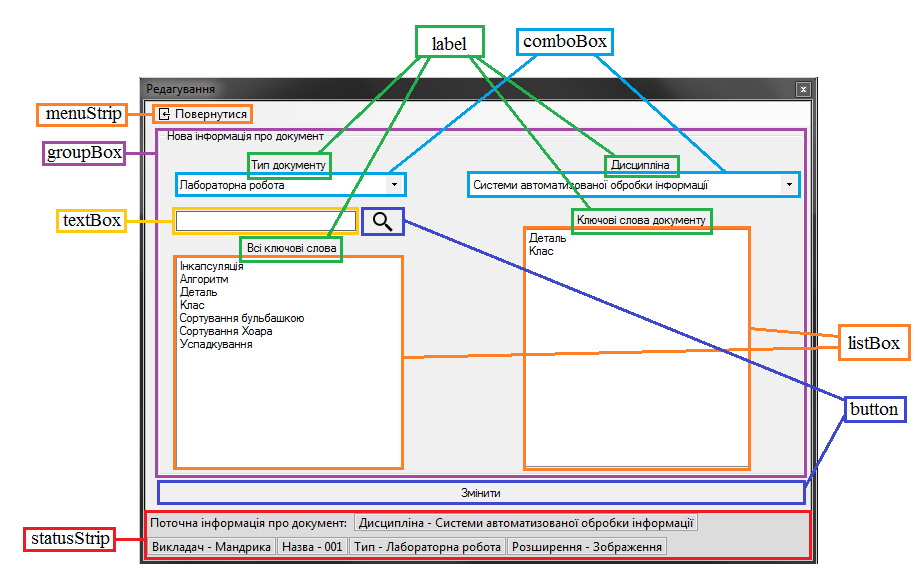


Рисунок 2.15 – Компоненти на формі

Перелік характеристик компонентів, параметрів та методів на формі:

* label — у дані компоненти виводяться допоміжні повідомлення для користувача, які свідчать про функціонал компонентів;
* comboBox — компонент представляє собою випадаючий список, у якому користувач може обрати дані про тип та дисципліну документу;
* menuStrip — даний компонент відповідає за повернення до меню;
* groupBox — компонент групує інші компоненти та додатково інформує користувача;
* textBox — до даного компоненту вводиться текст по якому буде реалізовано пошук ключових слів що знаходяться у компоненті listBox;
* statusStrip — у даний компоненту виводиться стара інформація про документ, а саме: дисципліна, викладач, назва, тип та розширення;
* listBox — у дані компоненти виводиться інформація про всі ключові слова та ключові слова, які належать документу;
* button — при натисканні компонентів реалізується або пошук ключових слів за введеним текстом, або редагування інформація про документу.
* RemoveAt(ціле значення) — даний метод використовується у компоненті listBox, даний метод видаляє елемент списку за встановленою позицією;
* SelectedItem() — метод повертає назву елемента, який було виділено, використовується у компоненті listBox;
* Clear() — метод видаляє всі елементи, які знаходилися у списку, використовується у компоненті listBox;
* Add(рядкове значення) — додає до списку атрибуту елемент, використовується у компонентах listBox та comboBox;
* Items[ціле значення] — даний параметр дозволяє дістатися до елементу списку, використовується у компонентах listBox та comboBox.

2.4 Опис основних алгоритмів програми

Один з основних алгоритмів програмного продукту — побудова дерева даних. Алгоритм полягає в тому, що дерево будується по кожному рівню окремо, кожний наступний рівень будується після того як побудовано попередній. Алгоритми побудови кожного рівня відрізняються між собою, кожний наступний рівень складніше будується, так як в одній гілці дерева може бути декілька елементів, код алгоритмів побудови дерева даних представлений у лістингах 2.1–2.3.

Перший рівень дерева даних будується за допомогою інформації у базі даних, за допомогою змінної — sql, передається запит до бази даних, після чого виконується запит та повертає все інформацію. Далі за допомогою метода — treeView1.Nodes.Add(), створюється вузол дерева, який відображається у компоненті «treeView», код побудови першого рівня дерева можна переглянути у лістингу 2.1.

Лістинг 2.1 – Побудова першого рівня дерева даних

string sql;

if(obTypeSeek.sortFlag == false) sql = "SELECT \* FROM thesis.disciplines;";

else sql = "SELECT \* FROM thesis.disciplines order by disciplines.discipline\_name;";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

treeView1.BeginUpdate();

treeView1.Nodes.Add(reader[1].ToString());

countElements.countSubject++;

treeView1.EndUpdate();

}

Другий рівень будується, в цілому, також як попередній за винятком, що інформація у дереві даних зрівнюються з інформацією у базі даних, при виявленні збіжності, запам’ятовується позиція, при якій було знайдено збіжність, та за цією позицією створюється новий вузол дерева, переглянути код побудови другого рівня можна у лістингу 2.2.

Лістинг 2.2 – Побудова другого рівня дерева даних

sql = "SELECT users.fio, studyload.id\_user, studyload.id\_discipline, disciplines.discipline\_name FROM users, studyload, disciplines WHERE users.id\_u = studyload.id\_user and studyload.id\_discipline = disciplines.id\_d;";

command = new MySqlCommand(sql, conn);

reader = command.ExecuteReader();

while (reader.Read())

{

bool countFirstNodesFlag = false;

int countFirstNodes = Convert.ToInt32(treeView1.GetNodeCount(countFirstNodesFlag));

int positionNodes = 0;

treeView1.BeginUpdate();

for (int i = 0; i < countFirstNodes; i++)

{

if (treeView1.Nodes[i].Text == reader[3].ToString())

{

positionNodes = i;

break;

}

}

treeView1.Nodes[positionNodes].Nodes.Add(reader[0].ToString());

arrayMaterials[countArrayMaterials] = new ClassMaterials();

arrayMaterials[countArrayMaterials].name = reader[0].ToString();

arrayMaterials[countArrayMaterials++].discipline = reader[3].ToString();

countElements.countName++;

treeView1.EndUpdate();

}

Даний алгоритм включає побудову зразу двох рівнів: третій та четвертий. Спочатку будується вузол третього рівня, після чого будуються всі дочірні вузли. Алгоритм включає в себе теж саме що й попередні, за винятком більшої кількості порівнянь інформації, почергово зрівнюється інформація від першого рівня до третього, код побудови двох рівнів можна побачити у лістингу 2.3.

Лістинг 2.3 – Побудова другого рівня дерева даних

sql = "SELECT users.fio, studyload.id\_s\_l, disciplines.discipline\_name, document\_specification.name\_specification, materials.name\_doc\_element, type.name\_type FROM materials, document\_specification, studyload, users, disciplines, type WHERE materials.id\_specification = document\_specification.id\_d\_s and studyload.id\_s\_l = materials.id\_study\_load and users.id\_u = studyload.id\_user and disciplines.id\_d = studyload.id\_discipline and materials.id\_type = type.id\_t;";

command = new MySqlCommand(sql, conn);

reader = command.ExecuteReader();

while (reader.Read())

{

bool writeNode = false;

bool countFirstNodesFlag = false;

int countFirstNodes = Convert.ToInt32(treeView1.GetNodeCount(countFirstNodesFlag));

bool tmp = false;

int tmpCount = 0;

treeView1.BeginUpdate();

for (int i = 0; i < countFirstNodes; i++)

{

tmpCount = Convert.ToInt32(treeView1.Nodes[i].GetNodeCount(tmp));

for (int j = 0; j < tmpCount; j++)

{

if (treeView1.Nodes[i].Text == reader[2].ToString() && treeView1.Nodes[i].Nodes[j].Text == reader[0].ToString())

{

for (int countArray = 0; countArray < countArrayMaterials; countArray++)

{

if (arrayMaterials[countArray].name == reader[0].ToString() && arrayMaterials[countArray].discipline == reader[2].ToString())

{

if (arrayMaterials[countArray].checkPrint(reader[3].ToString()) == false)

{

arrayMaterials[countArray].printToMaterials(reader[3].ToString());

treeView1.Nodes[i].Nodes[j].Nodes.Add(reader[3].ToString());

countElements.countTypeDocuments++;

}

bool secondTmp = false;

int t = Convert.ToInt32(treeView1.Nodes[i].Nodes[j].GetNodeCount(secondTmp));

for (int oneElement = 0; oneElement < Convert.ToInt32(treeView1.Nodes[i].Nodes[j].GetNodeCount(secondTmp)); oneElement++)

{

if (treeView1.Nodes[i].Nodes[j].Nodes[oneElement].Text == reader[3].ToString())

{

treeView1.Nodes[i].Nodes[j].Nodes[oneElement].Nodes.Add(reader[4].ToString() + "." + reader[5].ToString());

countElements.countDocuments++;

writeNode = true;

break;

}

}

}

}

}

}

if (writeNode == true) break;

}

treeView1.EndUpdate();

}

Алгоритм пошуку інформації у дереві даних, один з основних алгоритмів програми. Алгоритм полягає в тому, що, в першу чергу, формується запит який складається з пошуку інформації у базі даних. Далі запит передається та виконується на сервері, після чого повертається результат. Отриманий результат порівнюється з даними у дереві та якщо було знайдено співпадіння — відкриває вузол дерева за допомогою методу — treeView1.Nodes[позиція гілки].Toggle().

Порівняння даних здійснюється в декілька етапів, кожний наступний етап подібний до попереднього та включають в себе більше порівнювань на відповідність.

Всі етапи базуються на пошуку по рівнях, якщо було знайдено відповідний вузол або вузли, за допомогою метода — treeView1.Nodes[позиція гілки].Toggle(), то вузли будуть розкриті.

Окрім цього за допомогою даного метода, для відкриття вузлів, якщо даний метод використати для вже відкритого вузла, то даний вузол закриється.

Для виправлення даної проблеми було розроблено певний алгоритм, який полягає в тому, що про всі відкриті вузли ведеться інформація, яка записується по масиву, таким чином виконується ще одне порівняння з елементами даного масиву, якщо було знайдено відповідні елементи — вузол не змінюється, у іншому випадку вузол відкриється, а інформація про нього запишеться до масиву, переглянути код останнього етапу можна у лістингу 2.4, бо даний етап максимально демонструє усі попередні.

Лістинг 2.4 – Комбінований пошуку даних

int FlagFindElement = 0;

bool FlagFindFirstElement = false;

int []masNumberLastElement = new int[3] { 0, 0, 0};

string connStr = "server=localhost;user=root;database=thesis;password=s1233212;";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT materials.name\_doc\_element, type.name\_type FROM materials, type WHERE materials.id\_type = type.id\_t and name\_doc\_element LIKE ('%" + relatedWords(tmp) + "%');";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

if (FlagFindFirstElement == false) FlagFindFirstElement = true;

else break;

bool tmpForSeek = false;

for (int i = 0; i < treeView1.GetNodeCount(tmpForSeek); i++)

{

for (int j = 0; j < treeView1.Nodes[i].GetNodeCount(tmpForSeek); j++)

{

for (int m = 0; m < treeView1.Nodes[i].Nodes[j].GetNodeCount(tmpForSeek); m++)

{

for (int n = 0; n < treeView1.Nodes[i].Nodes[j].Nodes[m].GetNodeCount(tmpForSeek); n++)

{

if (relatedWordsFor\_materials(treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Text, tmp) == true)

{

FlagFindElement++;

int resArraySeekByKeyworks = 0;

for (int arrayI = 0; arrayI < countArraySeekByKeyworks; arrayI++)

{

if (arraySeekByKeyworks[arrayI].checkAllNode(i, j, m) == 3)

{

resArraySeekByKeyworks = 3;

break;

}

if (arraySeekByKeyworks[arrayI].checkAllNode(i, j, m) == 2)

{

resArraySeekByKeyworks = 2;

}

}

if (resArraySeekByKeyworks == 0)

{

arraySeekByKeyworks[countArraySeekByKeyworks] = new ClassSeekByKeyworks();

arraySeekByKeyworks[countArraySeekByKeyworks++].fillMasNodeNumber(i, j, m);

treeView1.BeginUpdate();

treeView1.Nodes[i].Toggle();

treeView1.Nodes[i].Nodes[j].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Toggle();

treeView1.EndUpdate();

}

else

{

if (resArraySeekByKeyworks == 2)

{

treeView1.BeginUpdate();

arraySeekByKeyworks[countArraySeekByKeyworks] = new ClassSeekByKeyworks();

arraySeekByKeyworks[countArraySeekByKeyworks++].fillMasNodeNumber(i, j, m);

treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Toggle();

treeView1.EndUpdate();

}

}

}

}

}

}

reader.Close();

conn.Close();

Алгоритм пошуку інформації у дереві даних за допомогою ключових слів. Алгоритм полягає в тому, що, в першу чергу, формується запит який складається з ключового слова, яке прив’язуються до документа у базі даних. Далі запит передається та виконується на сервері, де перевіряє введене слово з усіма існуючими та прив’язаними до документів, після чого повертається результат. Отриманий результат порівнюється з даними у дереві та якщо було знайдено співпадіння — відкриває вузол дерева за допомогою методу — treeView1.Nodes[позиція гілки].Toggle(), переглянути код пошуку даних за допомогою ключових слів можна у лістингу 2.5.

Лістинг 2.5 – Пошук за допомогою ключових слів

int FlagFindElement = 0;

string connStr = "server=localhost;user=root;database=thesis;password=s1233212;";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT users.fio, disciplines.discipline\_name, document\_specification.name\_specification, materials.name\_doc\_element, type.name\_type, tags.tag\_name FROM materials, document\_specification, studyload, users, disciplines, material\_tag, tags, type WHERE materials.id\_specification = document\_specification.id\_d\_s and studyload.id\_s\_l = materials.id\_study\_load and users.id\_u = studyload.id\_user and disciplines.id\_d = studyload.id\_discipline and materials.id\_m = material\_tag.id\_material and material\_tag.id\_tag = tags.id\_tag and materials.id\_type = type.id\_t and tags.tag\_name LIKE('%" + textBox1.Text + "%'); ";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

bool tmpForSeek = false;

for (int i = 0; i < treeView1.GetNodeCount(tmpForSeek); i++)

{

if (treeView1.Nodes[i].Text == reader[1].ToString())

{

for (int j = 0; j < treeView1.Nodes[i].GetNodeCount(tmpForSeek); j++)

{

if (treeView1.Nodes[i].Nodes[j].Text == reader[0].ToString())

{

for (int m = 0; m < treeView1.Nodes[i].Nodes[j].GetNodeCount(tmpForSeek); m++)

{

if (treeView1.Nodes[i].Nodes[j].Nodes[m].Text == reader[2].ToString())

{

for (int n = 0; n < treeView1.Nodes[i].Nodes[j].Nodes[m].GetNodeCount(tmpForSeek); n++)

{

string tmpForReader = reader[3].ToString() + "." + reader[4].ToString();

if (treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Text == tmpForReader)

{

FlagFindElement++;

int resArraySeekByKeyworks = 0;

for (int arrayI = 0; arrayI < countArraySeekByKeyworks; arrayI++)

{

if (arraySeekByKeyworks[arrayI].checkAllNode(i, j, m) == 3)

{

resArraySeekByKeyworks = 3;

break;

}

if (arraySeekByKeyworks[arrayI].checkAllNode(i, j, m) == 2)

{

resArraySeekByKeyworks = 2;

}

}

if (resArraySeekByKeyworks == 0)

{ arraySeekByKeyworks[countArraySeekByKeyworks] = new ClassSeekByKeyworks(); arraySeekByKeyworks[countArraySeekByKeyworks++].fillMasNodeNumber(i, j, m);

treeView1.BeginUpdate(); treeView1.Nodes[i].Toggle(); treeView1.Nodes[i].Nodes[j].Toggle(); treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle(); treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Toggle();

treeView1.EndUpdate();

}

else

{

if (resArraySeekByKeyworks == 2)

{

treeView1.BeginUpdate();

arraySeekByKeyworks[countArraySeekByKeyworks] = new ClassSeekByKeyworks(); arraySeekByKeyworks[countArraySeekByKeyworks++].fillMasNodeNumber(i, j, m);

treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle(); treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Toggle(); treeView1.EndUpdate();

}

}

}

}

}

}

}

}

reader.Close();

conn.Close();

1. ОПИС ПРОГРАМНОГО ПРОДУКТУ

Функціонал форми базується на компоненті treeView, бо з даним компонентом зв’язуються усі інші компоненти даної форми. На компоненті treeView зображені файли та документи, за замовчуванням всі вузли закриті.

Для пошуку інформації у дереві даних, перше що треба зробити — ввести текст до компоненту textbox, далі необхідно натиснути по компоненту button, після чого буде виконано пошук за введеним текстом, якщо пошук був вдалий — розкриються певні вузли дерева, у іншому випадку з’явиться повідомлення про невдалий пошук, а всі вузли згорнуться, результати пошуку даних можна переглянути на рисунках 3.1–3.2. Слід зазначити, що за замовчуванням тип пошуку даних — параметричний. Тип пошуку можна змінити у меню «Тип пошуку» на вкладці «Налаштування пошуку».

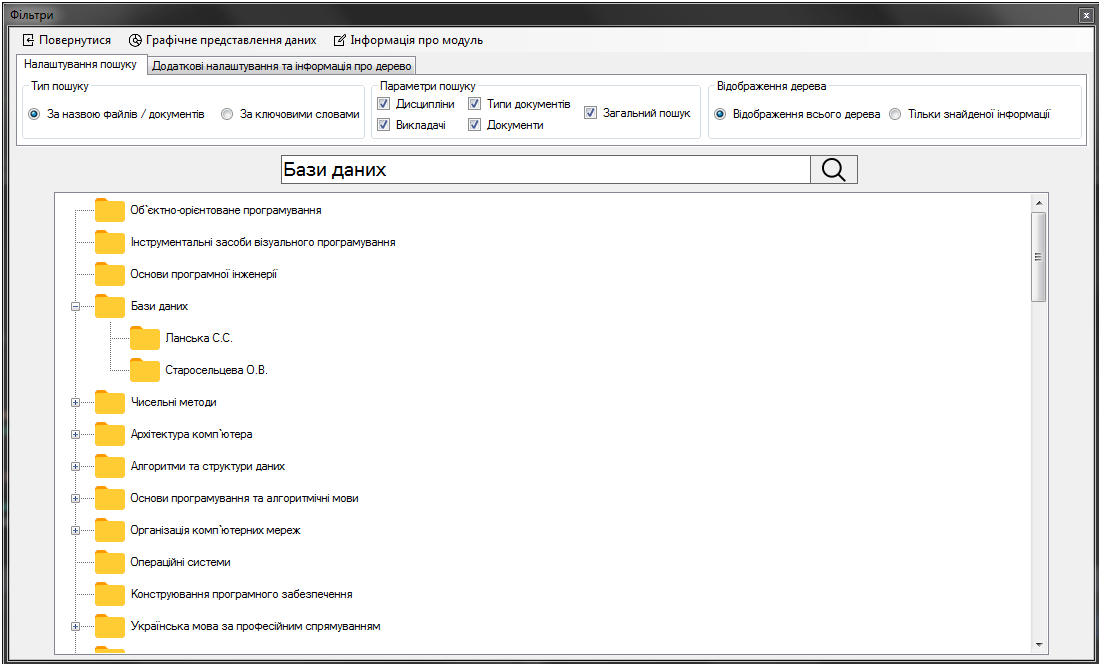


Рисунок 3.1 – Вдалий пошук

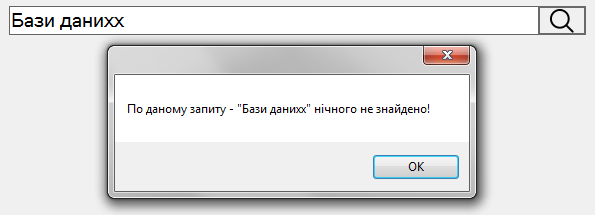


Рисунок 3.2 – Невдалий пошук

Для пошуку з параметрами необхідно вводити назву файлів та документів. Слід зазначити що не обов’язково вводити всю назву файлу або документа, можна ввести декілька символів, для пошуку даних цього вистачить. Також не обов’язково вводити текст у потрібному регістрі, вдалий пошук з даним прикладом можна побачити на рисунку 3.3.

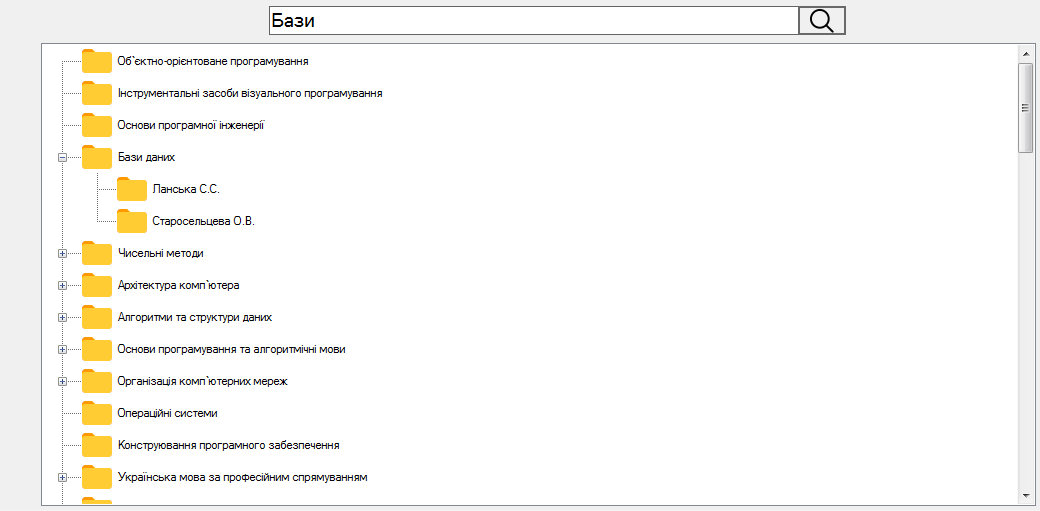


Рисунок 3.3 – Оптимізований вдалий пошук

Також пошук буде вдалим навіть якщо між символами будуть знаходитися проміжки, навіть якщо її буде декілька, Результат даного пошуку можна побачити на рисунку 3.4.

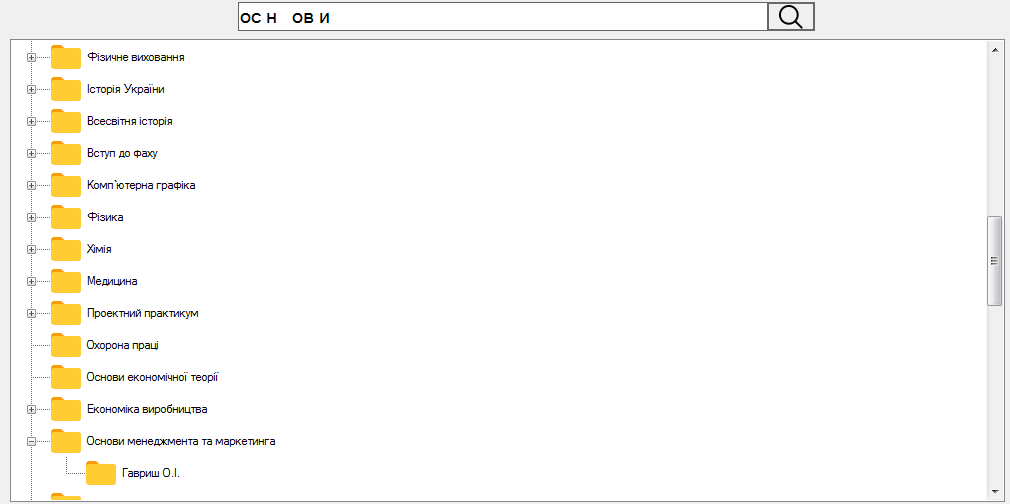


Рисунок 3.4 – Оптимізований вдалий пошук

Меню «Параметри пошуку» відповідають за налаштування саме параметричного пошуку. Таким чином параметричний пошук можна налаштувати за бажанням по різним рівням дерева, таким чином знаходити необхідні дані можна у багато разів швидше, результат пошуку з параметрами можна переглянути на рисунку 3.5.

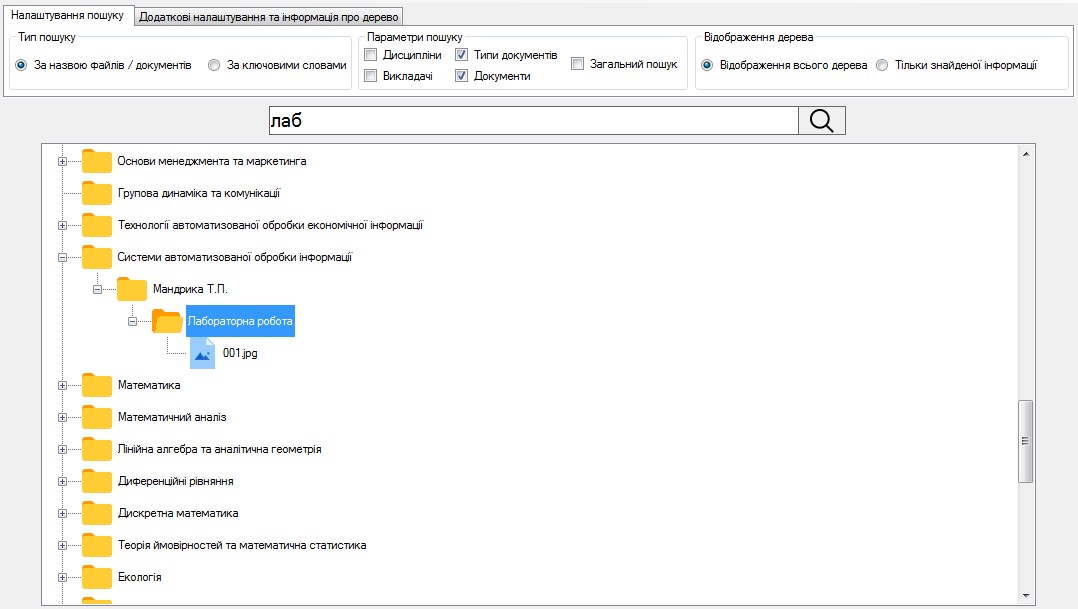


Рисунок 3.5 – Результат параметричного пошуку

Пошук по ключовим словам здійснюється таким же чином як і параметричний, за винятком того що необхідно перемкнути параметр пошуку у меню «Тип пошуку». Даний тип здійснює пошук за ключовими словами які прив’язуються тільки к документам (4 рівень дерева даних), пошук по ключовим словам зображено на рисунку 3.6.

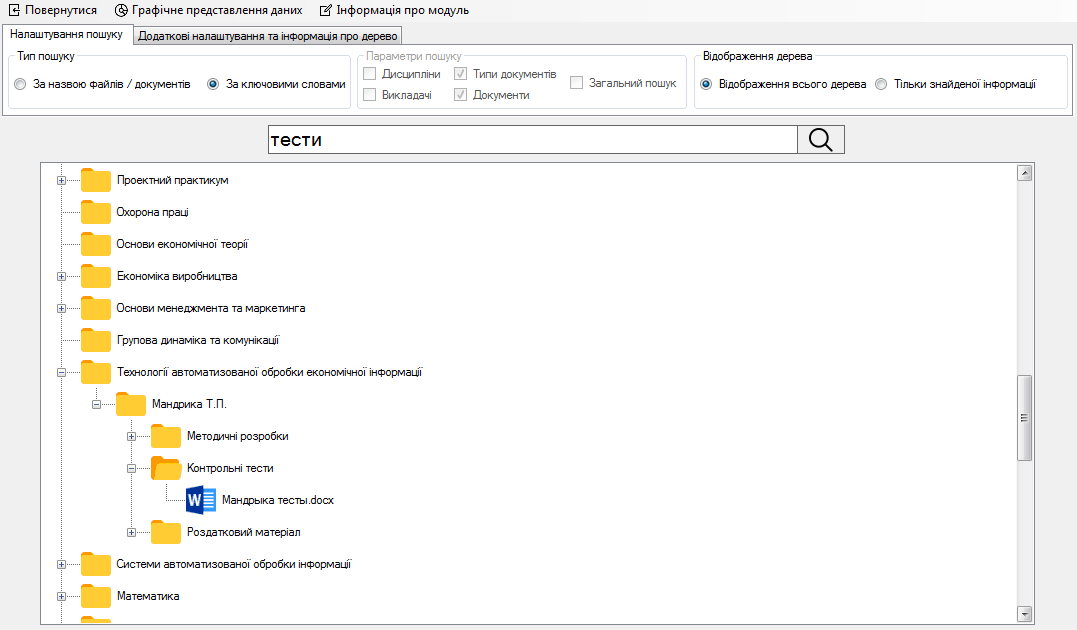


Рисунок 3.6 – Результат пошуку за допомогою ключових слів

Слід зазначити, якщо вдалий пошук знаходить більше одного вузла дерева — з’являється певне повідомлення, яке можна побачити на рисунку 3.7, дане повідомлення, разом з повідомленням про невдалий пошук, можна вимкнути у меню «Повідомлення», що знаходиться на вкладці «Відображення дерева та інформації».

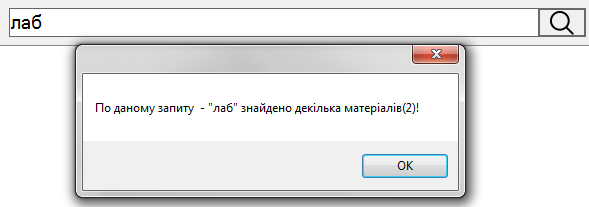


Рисунок 3.7 — Вдалий пошук при декількох знайдених вузлів

Щоб розрізнити файли від документів, було змінено рисунок документів в залежності від типу, зображено на рисунку 3.8.

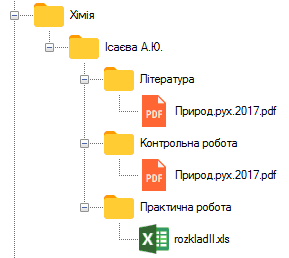


Рисунок 3.8 – Зображення різних типів документів

При великій кількості знайдених та розкритих вузлів дерева перегляд інформації ускладнюється, в такому випадку можна скористатися функцією відображення тільки знайдених даних, для цього необхідно обрати відповідний параметр відображення дерева «Тільки знайдені файли та документи», що знаходить у меню «Параметри відображення дерева при пошуку» на вкладці «Налаштування пошуку». Згідно з назвою дана функція відображує тільки знайдену інформацію у дереві даних, дану функцію можна переглянути на рисунку 3.9.

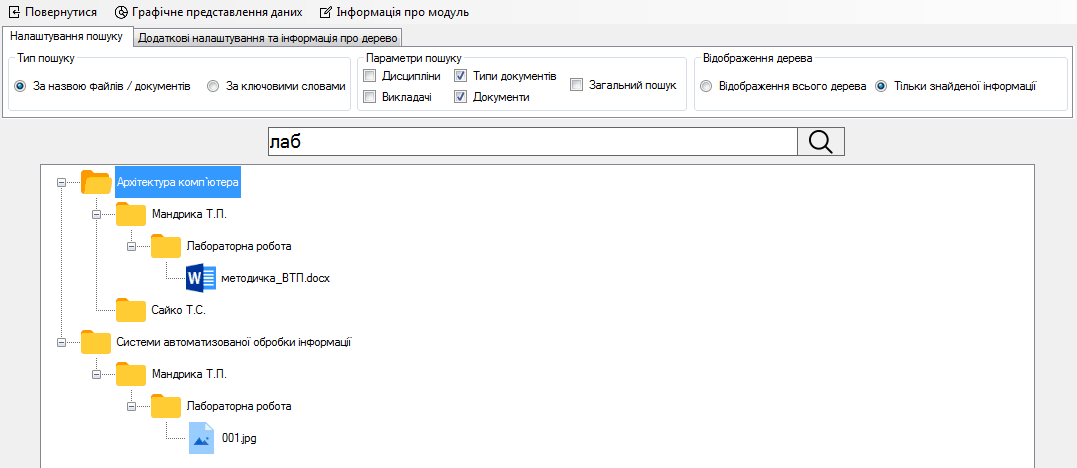


Рисунок 3.9 – Відображення тільки знайдених вузлів дерева даних

Слід зазначити, що окрім відображення знайденої інформації, дана функція також розгортає всі знайдені вузли дерева.

При зміні типу відображення дерева запит залишається та пристосовується до типу відображення дерева, тобто якщо переключити тип відображення дерева, коли дані було знайдено, всі знайдені файли залишаться у тій формі, у якій вони були до того, зміняться усі інші згорнуті файли (з’являться або зникнуть), зображено на рисунка 3.10.

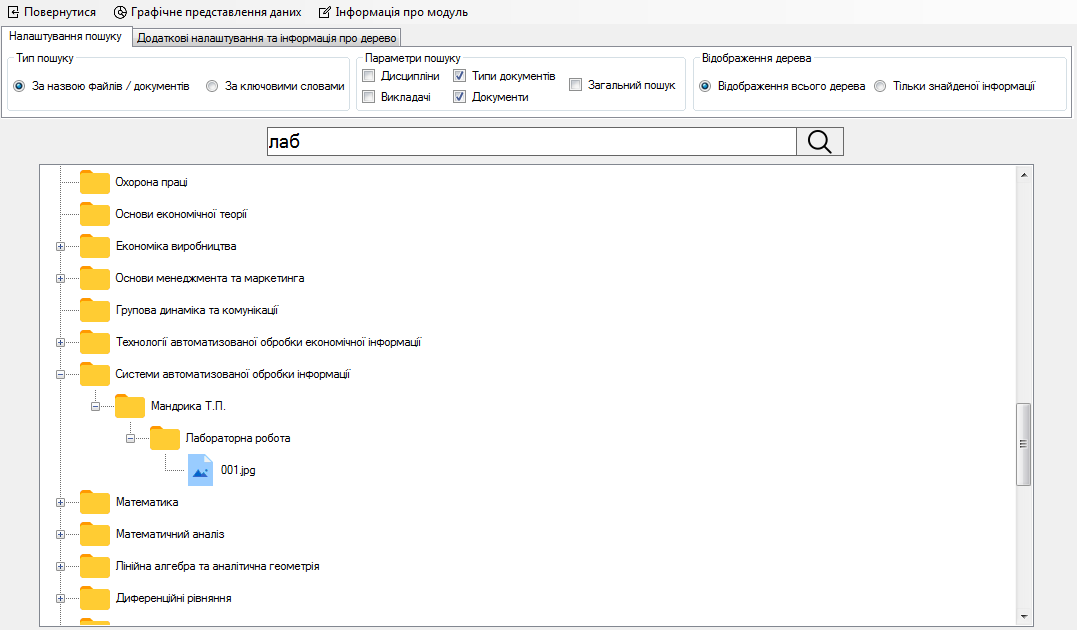


Рисунок 3.10 – Відображення всіх вузлів дерева даних при вдалому пошуку

При знаходженні великої кількості даних є можливість згорнути всі вузли дерева, дана, функція працює для обох типів відображення дерева.

Також дерево можна привести до базового вигляду, тобто вигляду за замовчуванням, для цього необхідно натиснути на компонент button, що знаходиться на вкладці «Відображення дерева та інформації». Даний компонент розташований у лівій частині меню налаштувань, в даній частині меню знаходиться три кнопки: згортання, відновлення, сортування. Усі кнопки розташовані у тому порядку, як їх було перераховано. Отже при натисканні другої кнопки все дерево відновиться та всі вузли згорнуться. Якщо навести та затримати курсор на кнопках з’явиться спливаюче віконце з інформацією про обрану кнопку, зображено на рисунку 3.11.

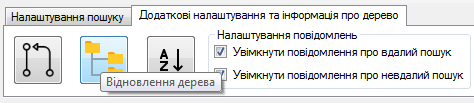


Рисунок 3.11 – Спливаюче віконце з інформаціє про фунціонал компоненту

Третя кнопка відповідає за сортування дерева даних. З відсортованим деревом можна виконувати всі раніше перераховані дії. Щоб відмінити сортування необхідно ще раз натиснути на дану кнопку, результат сортування зображено на рисунку 3.12.

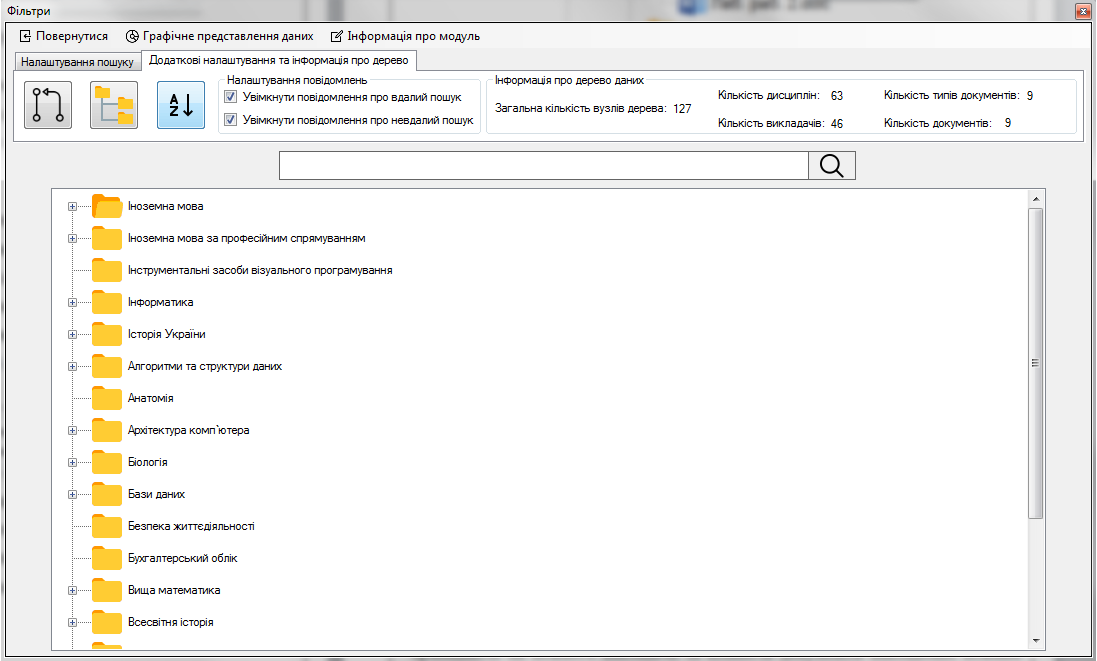


Рисунок 3.12 – Відсортоване дерево даних

В меню налаштування також можна переглянути інформацію про дерево даних, а саме: загальну кількість файлів, кількість файлів на кожному рівні у тому числі загальну кількість документів, дана інформація знаходиться у меню «Інформація про дерево даних», що знаходиться на вкладниці «Відображення дерева та інформації», зображено на рисунку 3.13.

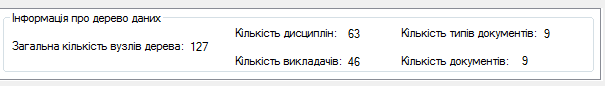


Рисунок 3.13 – Інформація про дерево даних

Щоб переглянути зміст необхідних документів, необхідно натиснути правою клавішею миші на необхідний документ, після чого з’явиться контекстне меню у якому необхідно обрати пункт «Відкрити документ», зображено на рисунку 3.14.

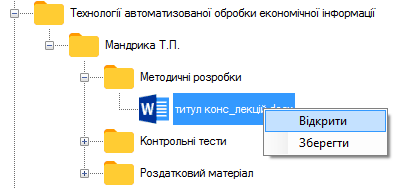


Рисунок 3.14 – Контекстне меню

За допомогою меню користувач може переходити до наступних форм, при натисканні другого параметру у меню користувач потрапить до форми графічного представлення даних.

На даній формі користувач має змогу переглядати статистичні дані, у різному вигляді, які пов’язані з викладачами, дисциплінами та документами, зовнішній вигляд форми можна побачити на рисунку 3.15.

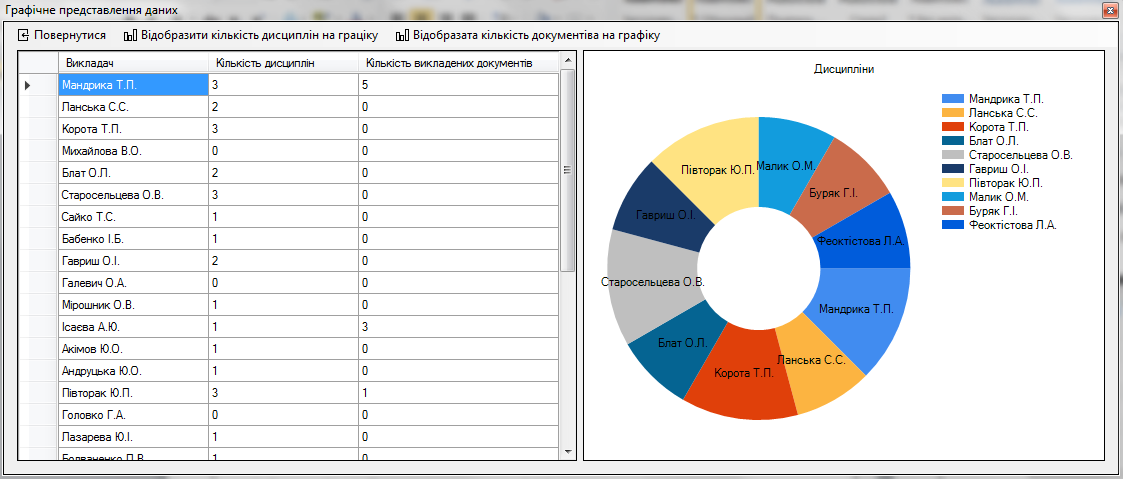


Рисунок 3.15 – Зовнішній вигляд форми

У лівій частині форми знаходиться інформація у вигляді таблиці, а даній таблиці представлена інформація про всіх викладачів, кількість дисциплін, які припадають на кожного викладача та кількість документів викладених кожним викладачем, є змога відсортувати інформацію у таблиці за кожним атрибутом у відповідності з типом атрибута.

У правій частині форми розміщається графік, на якому демонструється інформація, що й в таблиці за винятком того, що кількість викладачів у графіку обмежена, тому обираються тільки ті викладачі у яких більше дисциплін або викладених документів, на різних типів графіку — різна кількість відображеної інформації.

Графік за бажанням можна змінювати у меню, на вибір користувачу представляється два графіки: графік дисциплін, графік документів. Також кожному графіку можна змінювати тип, список типів для обох графіків однаковий, список доступних типів представлено на рисунку 3.16.

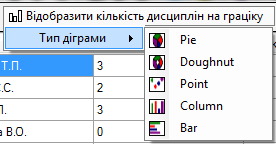


Рисунок 3.16 – Список типів графіків

При наведенні на кожний тип графіку буде з’являтися відповідна інформація про тип графіку, зображено на рисунку 3.17.

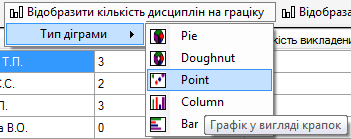


Рисунок 3.17 – Інформація про тип графіку

Для повернення на головну форму необхідно скористатися першим пунктом меню — «Повернутися». Після чого користувач зможе перейти на останню форму «Інформація про модуль». На даній формі користувач зможе ознайомитися з основним функціоналом всього модуля, зовнішній вигляд форми зображено на рисунку 3.18.

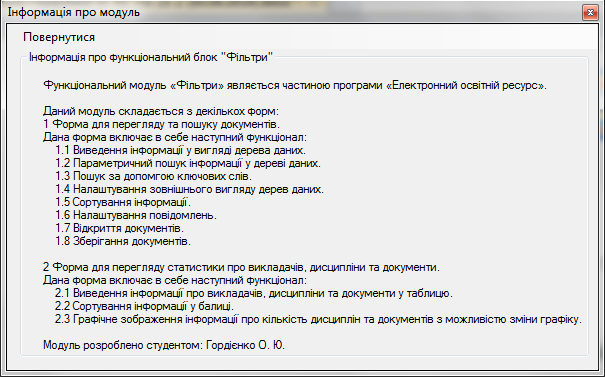


Рисунок 3.18 – Зовнішній вигляд форми

Щоб повернутися до головної форми необхідно скористатися параметром головного меню — «Повернутися».

У програмному модулі присутня ще одна форма «Редагування». Дана форма активується після обрання викладачем або адміністратором пункту редагування у контекстному меню документа. Зовнішній вигляд форми можна побачити на рисунку 3.19.

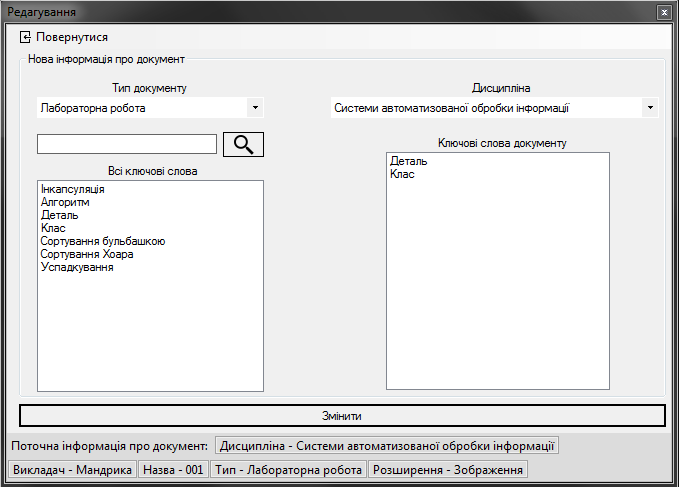


Рисунок 3.19 – Зовнішній вигляд форми «Редагування»

Форма «Редагування» забезпечує зміну інформації про документ, а саме: тип документу, дисципліну та ключові слова.

У нижній частині форми знаходиться інформація про документ у момент до редагування, тобто стара інформація.

Для додавання ключового слова до документу необхідно з лівого списку обрати відповідний елемент, після чого користувачу буде запропоновано додати дане ключове слово до документу, зображено на рисунку 3.20.

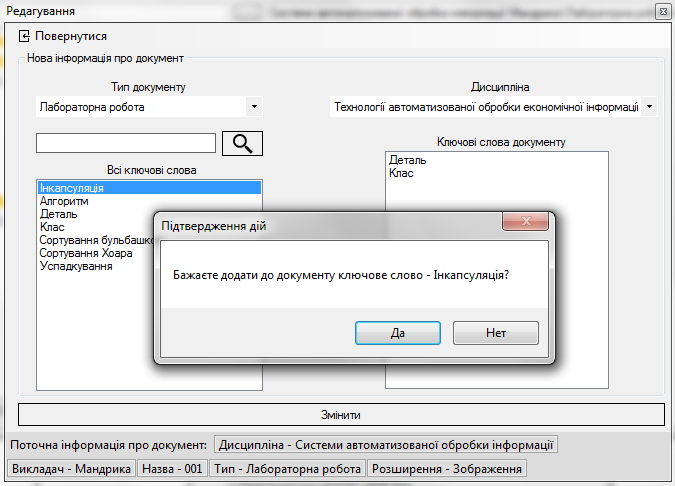


Рисунок 3.20 – Додавання до документу нове ключове слово

Якщо все було зроблено вірно, то з’явиться відповідне повідомлення та дане ключове слово додасться до списку праворуч, зображено рисунку 3.21.

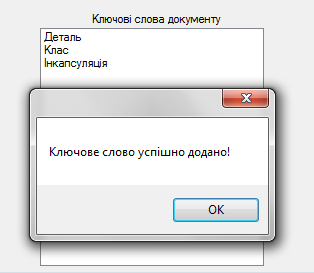


Рисунок 3.21 – Вдале додавання ключового слова до документу

Передбачена ситуація, коли користувач намагається додати ключове слово, яке вже є у списку прив’язаних до документу, у такому разі з’явиться певне повідомлення про помилку, зображено на рисунку 3.22.

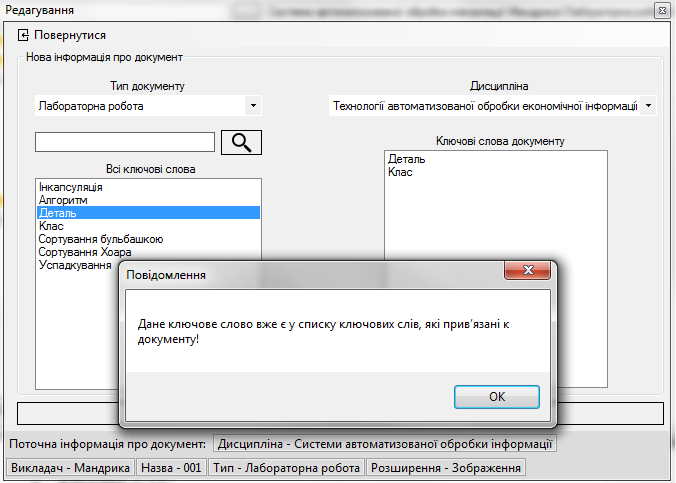


Рисунок 3.22 – Повідомлення про помилку

Для того щоб видалити ключове слово зі списку необхідно натиснути на обране ключове слово, після чого користувачу буде запропоновано видалити його, зображено на рисунку 3.23.

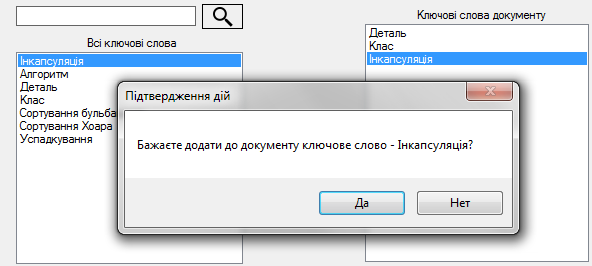


Рисунок 3.23 – Видалення ключового слова

При успішному видаленні з’явиться певне повідомлення та обране ключове слово зникне з правого списку, зображено на рисунку 3.24.

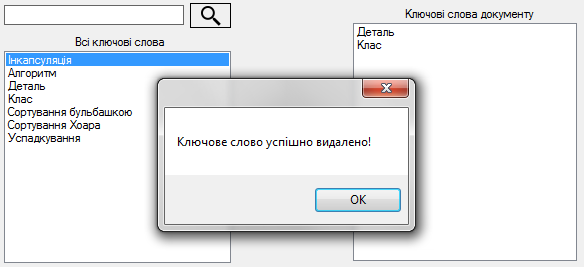


Рисунок 3.24 – Успішне видалення ключового слова

Передбачена ситуація коли користувач намагається видалити єдине ключове слово документу, програма не дозволяє документу залишитися без ключових слів, при спробі видалити останнє ключове слово документу — з’явиться певне повідомлення про помилку, зображено на рисунку 3.25.

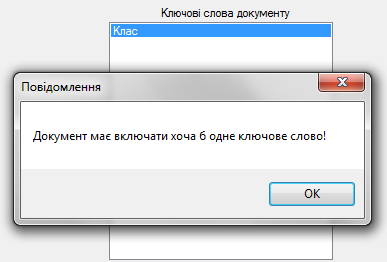


Рисунок 3.25 – Повідомлення про помилку

Після встановлення необхідної інформації про документ потрібно натиснути кнопку «Змінити». Якщо все зроблено вірно, то з’явиться певне повідомлення, зображено на рисунку 3.26.

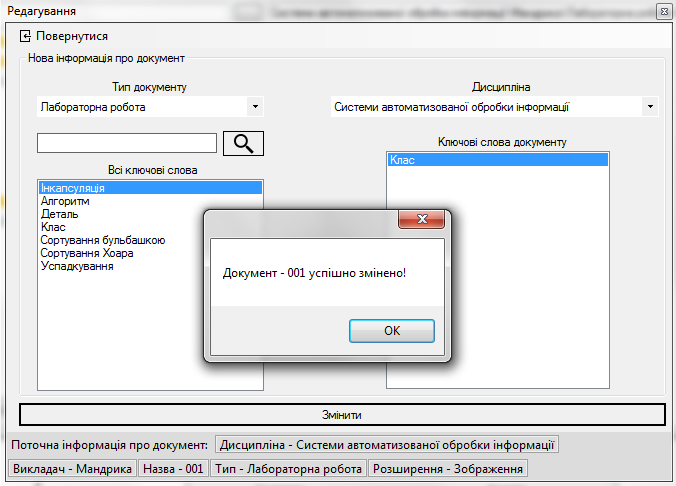


Рисунок 3.26 – Успішне редагування документу

Передбачена ситуація коли два документа з однаковими назвами заносять до одного файлу, у такому разі користувач отримує повідомлення про помилку, можна побачити на рисунку 3.27. Заповнені таблиці інформацією можна переглянути у додатку Д.

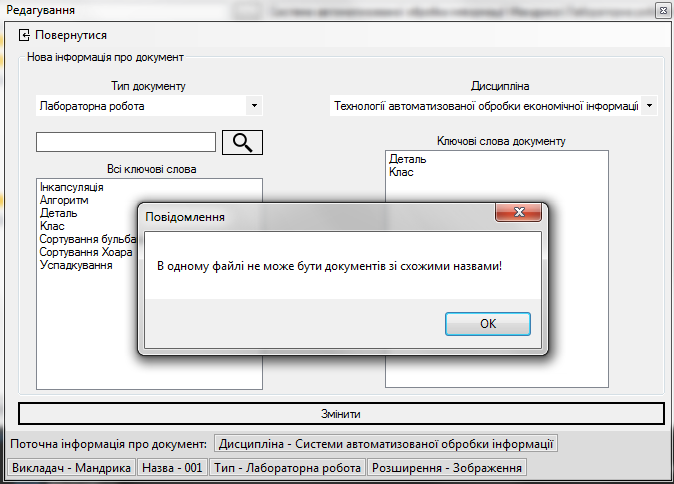


Рисунок 3.27 – Повідомлення про помилку

1. АНАЛІЗ ДОСЛІДНОЇ ЕКСПЛУАТАЦІЇ

Контроль на коректне введення, стабільне та нормалізоване відображення даних, оптимізований пошук даних, стабільна взаємодія компонентів — це все можна описати як надійність програмного продукту, якій необхідно приділяти багато часу, бо від результату залежить комфортність використання даної програми.

У даному розділі описано аналіз можливих ситуацій, що були проаналізовані при проектуванні додатку, тобто ситуації, що можуть призвести до виникнення помилок.

Однією з таких ситуацій була на етапі проектування бази даних. Ситуація виникла при тестуванні бази даних за допомогою запитів. Виникала аномалія надлишку інформації у базі даних, через те що певна частина бази даних була спроектована не вірно, а саме через те що одну дисципліну можуть вести декілька викладачів, таким чином інформація дублювалася, запит зображено на рисунку 4.1. У майбутньому ця проблема зачепила би функціонал пов'язаний з виведення даних.

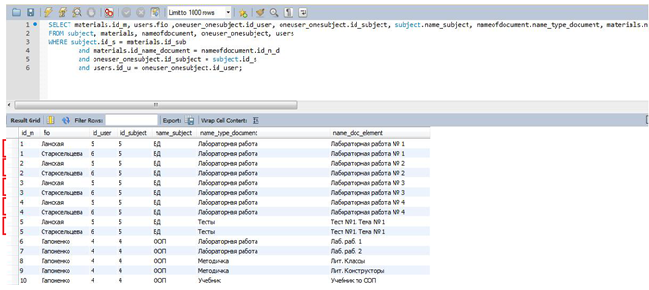


Рисунок 4.1 – Виконання запиту

У результаті було зроблено висновок, що дана аномалія виникла через неправильне з’єднання таблиць. Тому було прийнято рішення переглянути та перебудувати таблиці та зв’язки між ними. В результаті було перебудована таблицю «Studyload» та зв’язано з таблицею «Material», таким чином було вирішено дану проблему, минулі та теперішні зв’язки між таблицями можна переглянути на рисунках 4.2-4.3, вірний результат запиту можна побачити на рисунку 4.4.

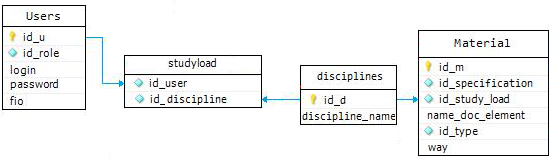


Рисунок 4.2 – Перший варіант з’єднання таблиць

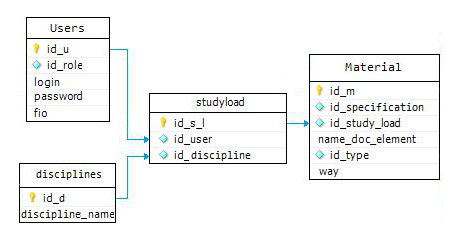


Рисунок 4.3 – Другий варіант з’єднання таблиць

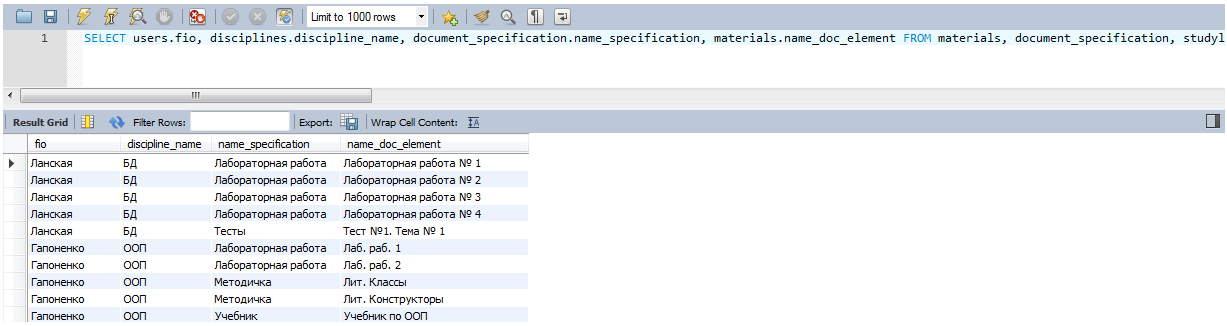


Рисунок 4.4 – Вірне виконання запиту

1. ОХОРОНА ПРАЦІ
   1. Аналіз небезпечних і шкідливих виробничих чинників проектованого технологічного процесу, об’єкту, системи або пристрою

Фактори небезпечних та шкідливих виробничих чинників для студентів та викладачів, задіяних в процесі використовування програмного продукту за допомогою персональних комп’ютерів у приміщенні поділяються на три категорії:

1. Психофізіологічні:

* тривале статичне напруження;
* великий обсяг інформації, що обробляється за одиницю часу;
* монотонність праці;
* нераціональна організація робочого місця;
* напруга зору;
* напруга пам’яті;
* напруга уваги.

2 Хімічні:

* підвищений вміст вуглекислого газу;
* підвищений вміст озону;
* підвищений вміст аміаку.

3 Фізичні:

* підвищений рівень пульсації світового потоку;
* нерівномірний розподіл яскравості в полі зору;
* підвищений вміст важких аероніонів;
* запиленість повітря робочого приміщення;
* підвищений рівень напруги в електричному ланцюзі, замикання якої може пройти через тіло працюючого;
* підвищений рівень випромінювання;
* можливість ураження статичною електрикою;
* підвищений рівень шуму [3].
  1. Інженерно-технічні заходи з охорони праці

Втома очей, підвищення навантаження на кисті верхніх кистівок та хребет, гіподинамія, нервово-емоційне напруження — усе це негативний вплив на організм людини при роботі з комп’ютером.

Для запобігання професійним захворюванням та підтримки працездатності було передбачено внутрішньозмінні регламентовані перезви для відпочинку, які містять додаткові нетривалі перерви в періоди, що передують появі втомлення та зниження працездатності.

Для розробників програм було передбачено регламентовану перерву для відпочинку тривалістю 15 хв. Через кожну годину при восьми годинній робочій зміні. У випадку, коли виробничі обставини не дозволяють застосувати регламентні перерви, тривалість безперервної роботи повинно не перевищувати чотири години.

Для зниження рівня шуму в приміщеннях з комп’ютерною технікою було здійснено наступні поради:

* акустична обробка приміщень(розміщення на внутрішніх поверхнях приміщень облаштувань, що поглинають звук);
* установка звукоізолюючого огородження у вигляді стін, перетину, кабін на шляху розповсюдження шуму;
* зміну напрямку випромінювання шуму в протилежну сторону від робочого місця;
* використання ПК, в яких вентилятор на процесорі встановлено типу (ВОХ-процесор), або кращій;
* переведення жорсткого диска в режим сплячки (Standby), якщо комп’ютер не працює на протязі визначеного часу.

Допустимі мікрокліматичні умови у приміщенні становлять – кратність повітрообміну 3 одиниці на годину, вологість від 40 до 60%, температура від 22 до 24ºС, швидкість руху повітря від 1 до 2 м/с. Для створення оптимальних мікрокліматичних умов було виконано, в першу чергу, зменшено тепловиділення в джерелі, що частково передбачено при конструюванні. Крім цього, забезпечено певну площу і об'єм виробничого приміщення: між кожним робочим місцем було забезпечено прохід шириною не меншою за 1 м, площа на одного працюючого становить 6 м2 і об'єм 19,5 м3. ЕОМ розміщуються так, щоб теплові потоки від них не були спрямовані на оператора або його сусідів, обмежено кількість ЕОМ в кожній аудиторії [4].

Віконні прорізи аудиторій для роботи з персональними комп’ютерами мають обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки. Для внутрішнього оздоблення приміщень з персональними комп’ютерами було використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі від 0,7 до 0,8, для стін від 0,5 до 0,6. Покриття підлоги зроблено матовим з коефіцієнтом відбиття від 0,3 до 0,5. Поверхня підлоги є рівною, неслизькою, з антистатичними властивостями.

Щодня перед початком роботи назначені особи очищають монітори від пилу та інших забруднень. Після закінчення роботи персональні комп’ютери і периферійні пристрої вимикаються від електричної мережі.

Було врегулювано освітленості на поверхні робочого столу, в зоні розміщення документів, становить від 300 до 500 лк. У місцях аудиторі, де дані значення освітленості неможливо забезпечити системою загального освітлення, було врегулювано за допомогою місцевого освітлення. При цьому світильники місцевого освітлення встановлено таким чином, щоб не створювати відблисків на поверхні екранів, а освітленість екранів не перевищує 300 лк.

Джерела світла в штучному освітленні застосовуються переважно люмінесцентні лампи типу ЛБ. У разі влаштування відбитого освітлення в аудиторіях допускається застосування металогалогенних ламп потужністю 250 Вт.

У разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім системи загального освітлення додатково встановлено світильники місцевого освітлення) [5].

Таким чином було вирішено проблеми освітлення та частково навантаження на зір людини, яка або створювала програмний продукт, або його використовує.

В аудиторіях, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлено аварійний резервний вимикач, який може повністю вимкнути електричне живлення аудиторії, крім освітлення.

Підлога була встановлена зі змінної конструкції, щоб забезпечувався вільний доступ до кабельних комунікацій під час обслуговування.

Персональні комп’ютери і периферійні пристрої підключаються до електромережі тільки за допомогою справних штепсельних з’єднань і електророзеток заводського виготовлення. У штепсельних з’єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, є спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція є такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз’єднання при відключенні є зворотним. Не допускається підключати персональні комп’ютери та периферійні пристрої до звичайної двопровідної електромережі, в тому числі з використанням перехідних пристроїв [6].

Периферійні пристрої у центрі приміщення прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не допускається застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, які містять сірку.

Значення напруженості електростатичного поля на робочих місцях (як у зоні екрана дисплея, так і на поверхнях обладнання, клавіатури, друкувального пристрою) мають не перевищувати гранично допустимих за ГОСТ 12.1.045-84 [7]. Значення напруженості електромагнітних полів на робочих місцях з ВДТ мають відповідати нормативним значенням ГОСТ 12.1.006-84 [8]. Інтенсивність потоків інфрачервоного випромінювання має не перевищувати допустимих значень відповідно до ГОСТ 12.1.005-88 [9]. Інтенсивність потоків ультрафіолетового випромінювання має не перевищувати допустимих значень відповідно до СН 245-71[10].

Напруженість електромагнітного поля на відстані 0,5 м навколо ПК ПО електричній складовій на магнітній складовій, потужність дози рентгенівського випромінювання на відстані 5 см від екрану та інших поверхонь ПК не перевищує 100 мкР/год.

Для моніторів, які не відповідають нормам, було встановлено захисний фільтр для екрана, що послабляє змінне електричне й електростатичне поля. Існує дві модифікації фільтрів: скляні і пластикові. Для здійснення колективного захисту, якщо сусідні робочі місця потрапляють у зону впливу поля, встановлено захисне покриття задньої і бічних стінок, змонтовано спеціальні екрануючі панелі на задню і бічні сторони монітора, встановлено перегородки між різними користувачами.

Таким чином були вирішені проблеми: можливість ураження статичною електрикою, підвищений рівень напруги в електричному ланцюзі, замикання якої може пройти через тіло працюючого.

* 1. Пожежна профілактика

Електрична енергія певних умовах легко переходить у теплову і це може викликати пожежі і вибухи. Пожежна небезпека електрообладнання, електронних приладів, радіоелектронної апаратури, апаратури управління, електроприймачів пов’язана з використанням спалимих матеріалів: гуми, пластмас, лаків, олій.

Джерелами займання можуть бути електричні іскри, дуги, коротке замикання, струмові перевантаження, перегріті опірні поверхні, несправність обладнання. Окиснювачем звичайно служить кисень. Але потужність і тривалість дії цих джерел займання порівняно малі, тому горіння, як правило, не розвивається. Виникнення пожежі в електронних пристроях можливо, якщо використовуються спалимі і важко спалимі матеріали і вироби.

Кабельні лінії електроживлення виконані з спалимого ізоляційного матеріалу, тому є найбільш пожежонебезпечними елементами в конструкціях електрообладнання.

Профілактикапередбачає наступні заходи:

* правильний вибір, монтаж і експлуатація електричних мереж, електрообладнання;
* правильний вибір конструкції електрообладнання, способу встановлення і класу ізоляції;
* електричний захист електричних мереж, електрообладнання (швидкодіючі реле, автоматичні вимикачі, запобіжники).

Запобігання пожеж від перевантажень:

* при проектуванні необхідно правильно вибирати переріз провідників мереж і схем за допустимою густиною струму;
* в процесі експлуатації електричних мереж не можна включати додатково електроприймачі, якщо мережа на це не розрахована;
* для захисту електрообладнання від струмів перевантаження найбільш ефективні автоматичні і електронні схеми захисту, вимикачі, теплові реле і плавкі запобіжники.

Запобігання пожеж від перехідних контактних з'єднань:

* для збільшення площі дійсного дотику контактів необхідно використовувати пружні контакти або спеціальні стальні пружини;
* для відводу тепла від точок дотику і розсіювання його необхідно виготовляти контакти певної маси і поверхні охолодження;
* всі контактні з’єднання повинні бути доступні для огляду [11].
  1. Заходи з ергономіки

При розміщенні робочих столів з персональними комп’ютерами слід дотримувати:

* відстань між бічними поверхнями персональних комп’ютерів 1,2 м.;
* відстань від тильної поверхні одного персонального комп’ютера до екрана іншого – 2,5 м.

За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою від 1,5 до 2 м.

Висота робочої поверхні робочого столу має регулюватися в межах від 680 до 800 мм, а ширина і глибина — забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: від 600 до 1400 мм, глибина — від 800 до 1000 мм).

Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600 мм, завширшки не менше ніж 500 мм, завглибшки (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги не менше ніж 650 мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край — заокругленим. Регулювання за кожним із параметрів має здійснюватися незалежно, легко і надійно фіксуватися. Шаг регулювання елементів стільця має становити: для лінійних розмірів — від 15 до 20 мм, для кутових — від 2 до 5 градусів.

Кут нахилу сидіння — до 15 градусів вперед і до 5 градусів назад. Висота спинки стільця має становити 300 мм, ширина – не менше ніж 380 мм, радіус кривизни горизонтальної площини — 400 мм. Поверхня сидіння і спинки стільця має бути напівм’якою з нековзним, повітронепроникним покриттям, що легко чиститься і не електризується. Робоче місце має бути обладнане підставкою для ніг завширшки не менше ніж 300 мм, завглибшки не менше ніж 400мм, що регулюється за висотою в межах до 150 мм і за кутом нахилу опорної поверхні підставки до 20 градусів.

Робочі місця слід розташовувати відносно світових прорізів так, щоб природне світло падало переважно з лівого боку. Монітор має розташовуватися на оптимальній відстані від очей користувача, що становить 600–700мм, але не ближче ніж за 600мм з урахуванням розміру літерно-цифрових знаків і символів.

Розташування пристрою введення — виведення інформації має забезпечувати добру видимість монітору, зручність ручного керування в зоні досяжності моторного поля і за висотою — від 900 до 1300 мм, за шириною від 400 до 500 мм. Під матричні принтери потрібно підкладати вібраційні килимки для гасіння вібрації та шуму.

Робоче місце з персональним комп’ютером слід обладнати пюпітром для документів, що легко переміщуються [6]. Наглядно побачити заходи з ергономіки на робочому місці за комп’ютером можна на рисунку 5.1.

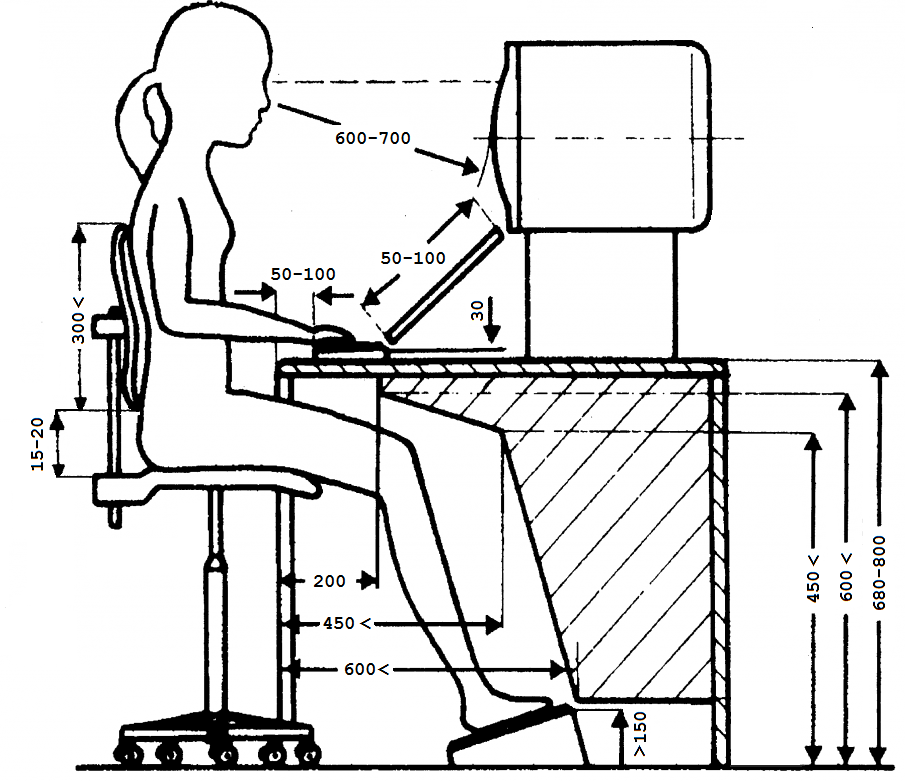


Рисунок 5.1 – Заходи з ергономіки на робочому місці за комп’ютером

1. ЕКОНОМІЧНА ЧАСТИНА

Згідно із завданням дипломного проекту необхідно визначити собівартість програмного продукту на тему «Електронний освітній ресурс. Програмний модуль інформаційного забезпечення навчально-виховного процесу». Для виконання розрахунку були використані початкові дані, представлені в таблиці 6.1.

Таблиця 6.1 — Початкові дані для розрахунку

| Найменування початкових даних | Показник | Джерело отримання |
| --- | --- | --- |
| 1. Трудомісткість складання (розробки) програми | 20 днів | Фактичні витрати часу на розробку програми |
| 1. Місячна ставка оператора — укладача програми | 5600,00 грн. | Дані переддипломної практики |
| 1. Кількість годин в місяці | 160 год. | Кількість робочих днів —  20 днів |
| 1. Додаткова зарплата | 10% | Дані переддипломної практики |
| 1. Нарахування єдиного соціального внеску | 22% | Дані переддипломної практики |
| 1. Загальновиробничі витрати (%) | 15% | Дані переддипломної практики |

Стаття 1. Матеріали : не використовуються

Стаття 2. Електроенергія:

*ВЕ =Tгод × РЕ* , (6.1)

де *Tгод* — кількість годин, необхідних для виконання програмного продукту;

РЕ — ціна за 1 кВт-год, грн.

*ВЕ*  = (42 \* 8) \* 1,68 = 564,48 грн.

Стаття 3. Основна заробітна плата:

 , (6.2)

де  — годинна тарифна ставка оператора, грн.;

 — кількість годин у місяці, приймається 160 – вихідні дані.

Визначаємо годинну тарифну ставку оператора:

, (6.3)

де  — місячна ставка оператора, грн.

*L* = 5600,00/160 = 35,00 грн/год.

Стаття 4. Додаткова заробітна плата:

, (6.4)

де — додаткова заробітна плата, грн.;

— відсоток додаткової заробітної плати, приймається 10% — вихідні дані.

З*дод* = 5600,00\*10% = 560,00 грн.

Стаття 5. Нарахування єдиного соціального внеску:

, (6.5)

З*соц* = (5600,00+560,00)\*22% = 1355,2 грн

Стаття 6. Загальновиробничі витрати:

, (6.6)

де  — загальновиробничі витрати, грн.;

 — відсоток загальновиробничих витрат, приймається 15% — вихідні дані.

З *заг* = 5600,00\*15% = 840,00 грн.

Виробнича собівартість:

, (6.7)

*Sпп* = 564,48 +5600,00+560,00+1355,2+840,00 = 8919,68 грн.

Таблиця 4.2 — Планова калькуляція виробничої собівартості розробки програми «Електронний освітній ресурс. Програмний модуль інформаційного забезпечення навчально-виховного процесу»

|  |  |
| --- | --- |
| Статті калькуляції | Сума, грн. |
| Стаття 1 Матеріали | — |
| Стаття 2 Комплектуючи вироби | 564,48 |
| Стаття 3 Основна заробітна плата | 5600,00 |
| Стаття 4 Додаткова заробітна плата | 560,00 |
| Стаття 5 Відрахування в соціальні фонди | 1355,2 |
| Стаття 6 Загальновиробничі витрати | 840,00 |
| Виробнича собівартість | 8919,68 |

Висновок: таким чином розрахунок показав, що собівартість програми «Електронний освітній ресурс. Програмний модуль інформаційного забезпечення навчально-виховного процесу»складає 8919,68 грн.

ВИСНОВКИ

В ході виконання дипломного проекту було розроблено програмний модуль,

який включає в себе весь передбачений завданням функціонал, а саме:

* зручне відображення інформації про документи, які зберігаються у базі даних;
* можливість налаштовування відображеної інформації;
* пошук документів: за допомогою параметрів, за ключовим словам;
* моніторинг статистичних даних про викладачів, дисципліни та документи, у різних типах: у графічному та табличному вигляді.
* можливість налаштовувати пошук даних;
* можливість переглядати зміст документів;
* можливість зберігати документи.

Таким чином в результаті було розроблено програмний продукт за допомогою якого має поліпшитися не тільки взаємодія між викладачем та студентом в плані автоматизації певних процесів, а ще й поліпшитися загальний рівень успішності студентів, за рахунок того, що даний програмний продукт об’єднує та зберігає корисну інформацію: про поточні або майбутні дисципліни, розклади екзаменів, корисну літературу в електронному вигляді тощо.

За бажанням даний програмний модуль можна пристосувати до різних програмних продуктів, до різних баз даних, таким чином оптимізуючи ще більшу кількість ситуацій.

У майбутньому програму можна поліпшити новим функціоналом, це може бути як додавання нових форм та компонентів, так і перенесення програми на web-додаток.

Також даний програмний модуль є прикладом та стимулом для студентів навчатися та створювати подібні, і не тільки, програми, які були б користі іншим людям.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Троелсен Э. Язык программирования С# 2010 и платформа .NET 4.0, 5-

е изд. / Э Троелсен. — М.: ООО “И.Д. Вильямс”, 2011. – 1392 с.

1. Офіційний сайт розробника середовища програмування Visual Studio [Електронний ресурс] —visualstudio.microsoft.com/ru/vs/features/ide/.
2. Основи охорони праці: Підручник. / К.Н. Ткачук, М.О. Халімовський, В.В. Зацарний та ін. — К.: Основа, 2003 — 472 с.
3. Інформація та документація. Структура та правила оформлення : ДСТУ 3008:2015. – Введ. в дію 01.07.2017. — К. : Держстандарт України, 2016. – 26 с.
4. Голінько В.І., Чеберячко С.І. Практикум з охорони праці. Навчальний посібник - Д.: Державний ВНЗ «НГУ», 2011. – 270 с.
5. ГОСТ 7.1:2006 «Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання».
6. ГОСТ 12.1.045-84. ССБТ. «Электростатические поля. Допустимые уровни на рабочих местах и требования к проведению контроля».
7. ГОСТ 12.1.006-84. «Допустимые поля радиочастот. Допустимые уровни на рабочих местах и требования к проведению контроля».
8. ГОСТ 12.1.005-88. «ССБТ. Общие санитарно-гигиенические требования к воздуху рабочей зоны».
9. СН 245-71 «Санитарные нормы проектирования промышленных предприятий».
10. ДБН В 2.5-13-98 «Інженерне обладнання будинків і споруд . Пожежна автоматика будинків.».

Додаток А

Календарний план робіт представлений у вигляді діаграми Ганта. Також до діаграми включається перелік та терміни виконання завдань, який представлений на рисунку А.1. Діаграма Ганта представлена на рисунку А.2.

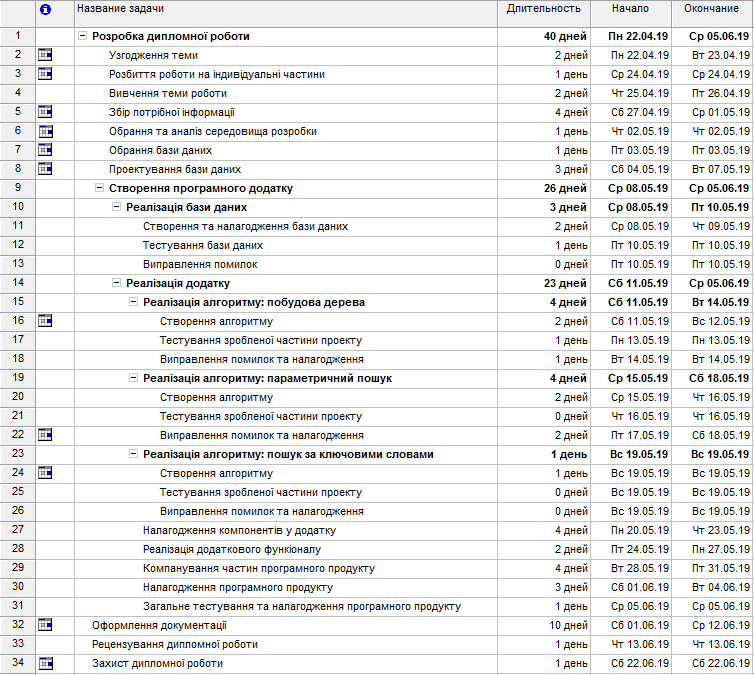


Рисунок А.1 – Перелік та терміни виконання завдань

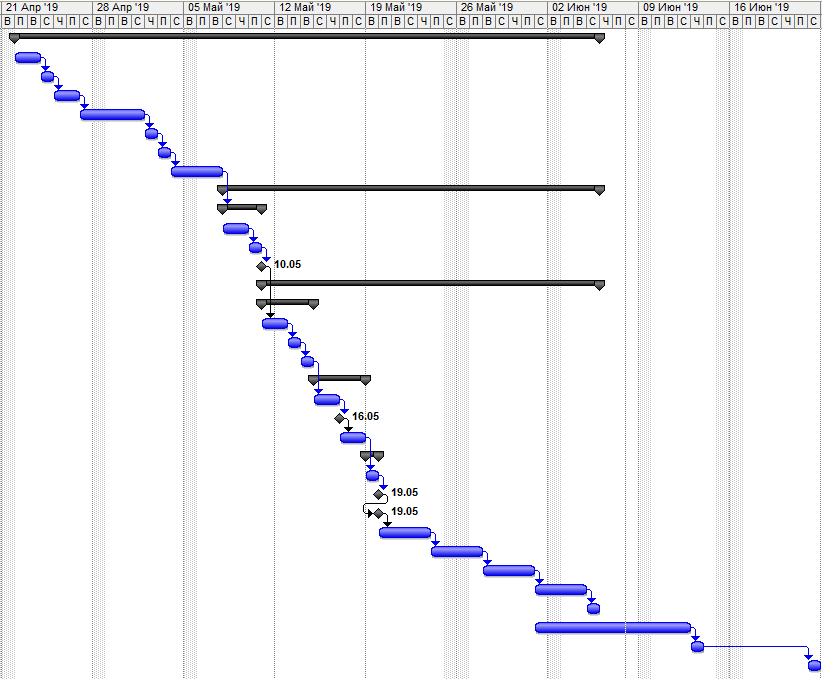


Рисунок А.2 – Діаграма Ганта

Додаток Б

Заповнені таблиці даними зображені на рисунках Б.1–Б.5.

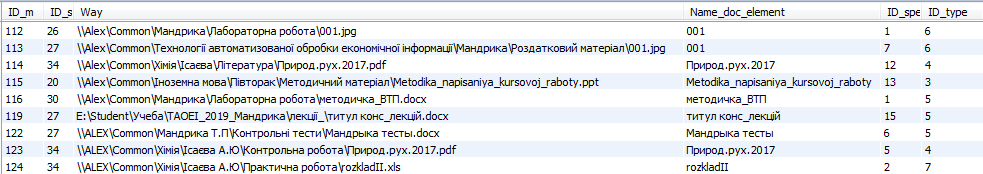


Рисунок Б.1 –Таблиця «materials»

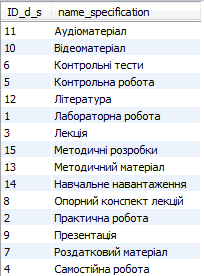


Рисунок Б.2 – Таблиця «document\_specification»

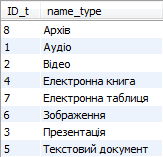


Рисунок Б.3 – Таблиця «type»

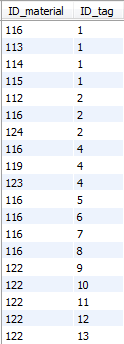


Рисунок Б.4 – Таблиця «material\_tag»

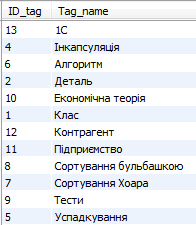


Рисунок Б.5 – Таблиця «tags»

Додаток В

Діаграма сценаріїв предметної області індивідуального завдання зображена на рисунку В.1.

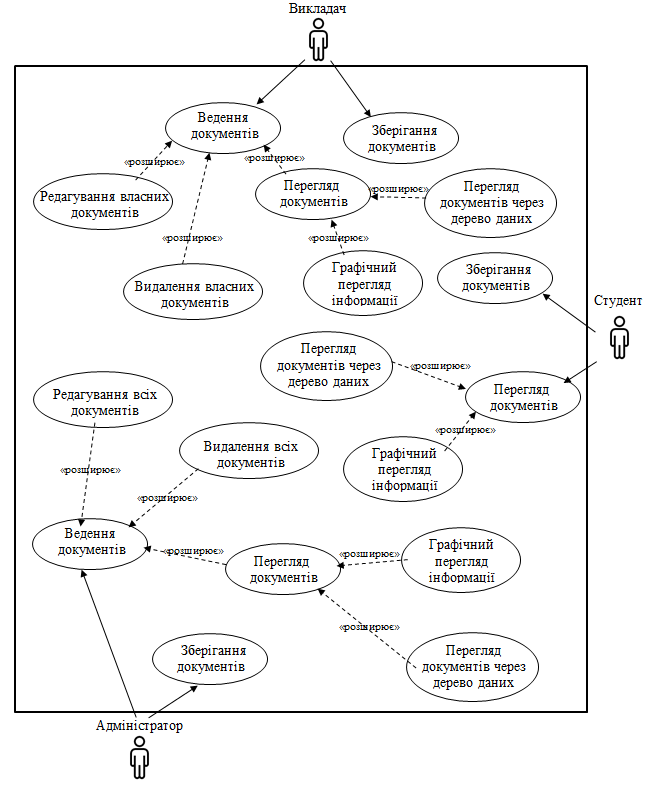


Рисунок В.1 – Діаграма сценаріїв предметної області

Додаток Г

Схема взаємодії модулів програмного додатку наведена на рисунку Г.1.

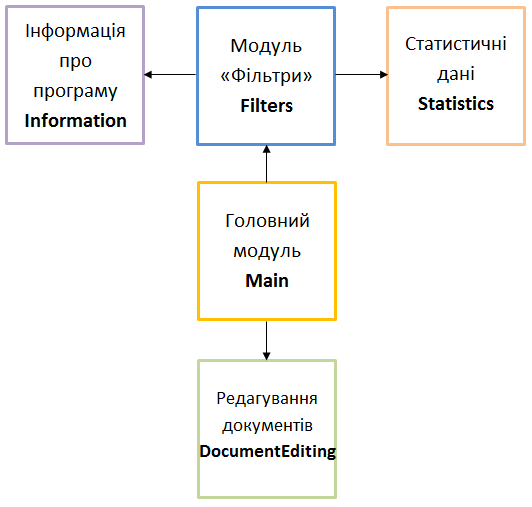


Рисунок Г.1 – Схема взаємодії модулів програмного додатку

Додаток Ґ

Код основних елементів програми можно переглянути у лістиграх Ґ.1– Ґ.3.

Лістинг Ґ.1 – Форма «Фільтри»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using System.IO;

namespace FormSeek

{

public partial class Form1 : Form

{

ClassOptionsSeek obTypeSeek = new ClassOptionsSeek();

ClassMaterials[] arrayMaterials = new ClassMaterials[50];

ClassCountElementsInTreeView countElements = new ClassCountElementsInTreeView();

int countArrayMaterials = 0;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

checkBox1.Checked = true;

checkBox2.Checked = true;

checkBox3.Checked = true;

checkBox4.Checked = true;

checkBox5.Checked = true;

defaultOptions();

loadDefoltTree();

getTypeForDocument();

infomationAboutElement();

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

obTypeSeek.radioButton = 1;

groupBox3.Enabled = true;

if (checkBox1.Checked == false && checkBox2.Checked == false && checkBox3.Checked == false && checkBox4.Checked == false && checkBox5.Checked == false) button1.Enabled = false;

else button1.Enabled = true;

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

obTypeSeek.radioButton = 2;

button1.Enabled = true;

groupBox3.Enabled = false;

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton3.Checked == true) obTypeSeek.radioButtonSecond = 1;

else obTypeSeek.radioButtonSecond = 2;

bool tmpForSeek = false;

int countNodes = treeView1.GetNodeCount(tmpForSeek);

for (int i = 0; i < countNodes; i++) treeView1.Nodes[0].Remove();

loadDefoltTree();

getTypeForDocument();

if (obTypeSeek.radioButtonSecond == 1)

{

if (textBox1.Text.Length != 0)

{

if (checkBox1.Checked == true || checkBox2.Checked == true || checkBox3.Checked == true || checkBox4.Checked == true || checkBox5.Checked == true) buttonClickFunction();

}

}

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton4.Checked == true) obTypeSeek.radioButtonSecond = 2;

else obTypeSeek.radioButtonSecond = 1;

if (obTypeSeek.radioButtonSecond == 2)

{

if (textBox1.Text.Length != 0)

{

if (checkBox1.Checked == true || checkBox2.Checked == true || checkBox3.Checked == true || checkBox4.Checked == true || checkBox5.Checked == true) buttonClickFunction();

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text.Length != 0)

{

buttonClickFunction();

}

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

bool tmp = checkBox5.Checked;

if (checkBox1.Checked == false) obTypeSeek.inputCheckBox(false, 0);

else obTypeSeek.inputCheckBox(true, 0);

if (checkBox1.Checked == false && checkBox5.Checked == true) checkOtherCheckBox();

if (checkBox1.Checked == true && checkBox2.Checked == true && checkBox3.Checked == true && checkBox4.Checked == true && checkBox5.Checked == false) checkBox5.Checked = true;

if (checkBox1.Checked == false && checkBox2.Checked == false && checkBox3.Checked == false && checkBox4.Checked == false && checkBox5.Checked == false) button1.Enabled = false;

else button1.Enabled = true;

}

private void checkBox2\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox2.Checked == false) obTypeSeek.inputCheckBox(false, 1);

else obTypeSeek.inputCheckBox(true, 1);

if (checkBox2.Checked == false && checkBox5.Checked == true) checkOtherCheckBox();

if (checkBox1.Checked == true && checkBox2.Checked == true && checkBox3.Checked == true && checkBox4.Checked == true && checkBox5.Checked == false) checkBox5.Checked = true;

if (checkBox1.Checked == false && checkBox2.Checked == false && checkBox3.Checked == false && checkBox4.Checked == false && checkBox5.Checked == false) button1.Enabled = false;

else button1.Enabled = true;

}

private void checkBox3\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox3.Checked == false) obTypeSeek.inputCheckBox(false, 2);

else obTypeSeek.inputCheckBox(true, 2);

if (checkBox3.Checked == false && checkBox5.Checked == true) checkOtherCheckBox();

if (checkBox1.Checked == true && checkBox2.Checked == true && checkBox3.Checked == true && checkBox4.Checked == true && checkBox5.Checked == false) checkBox5.Checked = true;

if (checkBox1.Checked == false && checkBox2.Checked == false && checkBox3.Checked == false && checkBox4.Checked == false && checkBox5.Checked == false) button1.Enabled = false;

else button1.Enabled = true;

}

private void checkBox4\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox4.Checked == false) obTypeSeek.inputCheckBox(false, 3);

else obTypeSeek.inputCheckBox(true, 3);

if (checkBox4.Checked == false && checkBox5.Checked == true) checkOtherCheckBox();

if (checkBox1.Checked == true && checkBox2.Checked == true && checkBox3.Checked == true && checkBox4.Checked == true && checkBox5.Checked == false) checkBox5.Checked = true;

if (checkBox1.Checked == false && checkBox2.Checked == false && checkBox3.Checked == false && checkBox4.Checked == false && checkBox5.Checked == false) button1.Enabled = false;

else button1.Enabled = true;

}

private void checkBox5\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox5.Checked == false)

{

if (checkBox1.Checked == true && checkBox2.Checked == true && checkBox3.Checked == true && checkBox4.Checked == true)

{

button1.Enabled = false;

obTypeSeek.inputCheckBox(false, false, false, false, false);

checkBox1.Checked = false;

checkBox2.Checked = false;

checkBox3.Checked = false;

checkBox4.Checked = false;

}

}

else

{

obTypeSeek.inputCheckBox(true, true, true, true, true);

checkBox1.Checked = true;

checkBox2.Checked = true;

checkBox3.Checked = true;

checkBox4.Checked = true;

}

}

private void checkBox6\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox6.Checked == true) obTypeSeek.masMessage[0] = true;

else obTypeSeek.masMessage[0] = false;

}

private void checkBox7\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox7.Checked == true) obTypeSeek.masMessage[1] = true;

else obTypeSeek.masMessage[1] = false;

}

private void openMenuItem\_Click(object sender, EventArgs e)

{

seekFocusNode(Convert.ToString(treeView1.SelectedNode));

}

private void button2\_Click(object sender, EventArgs e)

{

treeView1.CollapseAll();

}

private void button3\_Click(object sender, EventArgs e)

{

bool tmpForSeek = false;

int countNodes = treeView1.GetNodeCount(tmpForSeek);

for (int i = 0; i < countNodes; i++) treeView1.Nodes[0].Remove();

loadDefoltTree();

getTypeForDocument();

}

private void button4\_Click(object sender, EventArgs e)

{

if (obTypeSeek.sortFlag == false) obTypeSeek.sortFlag = true;

else obTypeSeek.sortFlag = false;

bool tmpForSeek = false;

int countNodes = treeView1.GetNodeCount(tmpForSeek);

for (int i = 0; i < countNodes; i++) treeView1.Nodes[0].Remove();

loadDefoltTree();

getTypeForDocument();

}

private void ToolStripMenuItem1\_Click(object sender, EventArgs e)

{

InformationInTable form2 = new InformationInTable();

Hide();

form2.ShowDialog();

Show();

}

private void ToolStripMenuItem3\_Click(object sender, EventArgs e)

{

InformationAboutProgram form3 = new InformationAboutProgram();

Hide();

form3.ShowDialog();

Show();

}

void defaultOptions()

{

obTypeSeek.radioButton = 1;

obTypeSeek.radioButtonSecond = 1;

}

void buttonClickFunction()

{

if (obTypeSeek.radioButtonSecond == 2)

{

bool tmpForSeek = false;

int countNodes = treeView1.GetNodeCount(tmpForSeek);

for (int i = 0; i < countNodes; i++) treeView1.Nodes[0].Remove();

loadDefoltTree();

getTypeForDocument();

}

treeView1.CollapseAll();

if (obTypeSeek.masCheckBox[4] == true || obTypeSeek.radioButton == 2)

{

treeView1.Visible = false;

if (obTypeSeek.radioButton == 1)

{

if (seekByName(textBox1.Text) == false)

{

if (checkOptionsForMessageNotFound() == true) MessageBox.Show("По даному запиту - \"" + textBox1.Text + "\" нічного не знайдено!");

}

else

{

showFoundNodes();

}

}

else

{

int resSeekByKeywords = seekByKeywords(textBox1.Text);

if (resSeekByKeywords == 0)

{

if (checkOptionsForMessageNotFound() == true) MessageBox.Show("По даному запиту - \"" + textBox1.Text + "\" нічного не знайдено!");

}

else

{

if (resSeekByKeywords > 1)

{

if (checkOptionsForMessageFound() == true) MessageBox.Show("По даному запиту - \"" + textBox1.Text + "\" знайдено декілька матеріалів(" + resSeekByKeywords + ")!");

}

showFoundNodes();

}

}

treeView1.Visible = true;

}

else

{

int[] masResSeekWithParameters = new int[4];

if (obTypeSeek.masCheckBox[0] == true)

{

masResSeekWithParameters[0] = Convert.ToInt32(seekByName\_subject(textBox1.Text));

}

if (obTypeSeek.masCheckBox[1] == true)

{

masResSeekWithParameters[1] = Convert.ToInt32(seekByName\_name(textBox1.Text));

}

if (obTypeSeek.masCheckBox[2] == true)

{

masResSeekWithParameters[2] = Convert.ToInt32(seekByName\_nameofdocument(textBox1.Text));

}

if (obTypeSeek.masCheckBox[3] == true)

{

masResSeekWithParameters[3] = Convert.ToInt32(seekByName\_materials(textBox1.Text));

}

int[] masFlagForSeek = new int[4];

int countMasFlagForSeek = 0;

for (int i = 0; i < obTypeSeek.masCheckBox.Length; i++)

{

if (obTypeSeek.masCheckBox[i] == true)

{

masFlagForSeek[countMasFlagForSeek++] = masResSeekWithParameters[i];

}

}

int resSeek = 0;

for (int i = 0; i < countMasFlagForSeek; i++)

{

if (masFlagForSeek[i] > 0) resSeek += masFlagForSeek[i];

}

if (resSeek == 0)

{

if (checkOptionsForMessageNotFound() == true) MessageBox.Show("По даному запиту - \"" + textBox1.Text + "\" нічного не знайдено!");

}

else

{

if (resSeek > 1)

{

if (checkOptionsForMessageFound() == true) MessageBox.Show("По даному запиту - \"" + textBox1.Text + "\" знайдено декілька матеріалів(" + resSeek + ")!"); ;

}

showFoundNodes();

}

}

}

void checkOtherCheckBox()

{

checkBox5.Checked = false;

obTypeSeek.inputCheckBox(false, 4);

}

void infomationAboutElement()

{

countElements.countAllElements = countElements.countSubject + countElements.countName + countElements.countTypeDocuments + countElements.countDocuments;

label1\_1.Text = Convert.ToString(countElements.countAllElements);

label2\_1.Text = Convert.ToString(countElements.countSubject);

label3\_1.Text = Convert.ToString(countElements.countName);

label4\_1.Text = Convert.ToString(countElements.countTypeDocuments);

label5\_1.Text = Convert.ToString(countElements.countDocuments);

}

void loadDefoltTree()

{

countArrayMaterials = 0;

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql;

if(obTypeSeek.sortFlag == false) sql = "SELECT \* FROM disciplines;";

else sql = "SELECT \* FROM disciplines order by disciplines.discipline\_name;";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

treeView1.BeginUpdate();

treeView1.Nodes.Add(reader[1].ToString());

countElements.countSubject++;

treeView1.EndUpdate();

}

reader.Close();

conn.Close();

conn.Open();

//sql = "SELECT users.fio, oneuser\_onesubject.id\_user, oneuser\_onesubject.id\_subject, subject.name\_subject FROM users, oneuser\_onesubject, subject WHERE users.id\_u = oneuser\_onesubject.id\_user and oneuser\_onesubject.id\_subject = subject.id\_s;";

sql = "SELECT users.fio, studyload.id\_user, studyload.id\_discipline, disciplines.discipline\_name FROM users, studyload, disciplines WHERE users.id\_u = studyload.id\_user and studyload.id\_discipline = disciplines.id\_d;";

command = new MySqlCommand(sql, conn);

reader = command.ExecuteReader();

while (reader.Read())

{

bool countFirstNodesFlag = false;

int countFirstNodes = Convert.ToInt32(treeView1.GetNodeCount(countFirstNodesFlag));

int positionNodes = 0;

treeView1.BeginUpdate();

for (int i = 0; i < countFirstNodes; i++)

{

if (treeView1.Nodes[i].Text == reader[3].ToString())

{

positionNodes = i;

break;

}

}

treeView1.Nodes[positionNodes].Nodes.Add(reader[0].ToString());

arrayMaterials[countArrayMaterials] = new ClassMaterials();

arrayMaterials[countArrayMaterials].name = reader[0].ToString();

arrayMaterials[countArrayMaterials++].discipline = reader[3].ToString();

countElements.countName++;

treeView1.EndUpdate();

}

reader.Close();

conn.Close();

conn.Open();

sql = "SELECT users.fio,studyload.id\_s\_l,disciplines.discipline\_name, document\_specification.name\_specification, material.name\_doc\_element, material.way FROM material, document\_specification, studyload, users, disciplines, type WHERE material.id\_specification = document\_specification.id\_d\_s and studyload.id\_s\_l = material.id\_study\_load and users.id\_u = studyload.id\_user and disciplines.id\_d = studyload.id\_discipline and material.id\_type = type.id\_t;";

command = new MySqlCommand(sql, conn);

reader = command.ExecuteReader();

while (reader.Read())

{

bool writeNode = false;

bool countFirstNodesFlag = false;

int countFirstNodes = Convert.ToInt32(treeView1.GetNodeCount(countFirstNodesFlag));

bool tmp = false;

int tmpCount = 0; // количесво веток

treeView1.BeginUpdate();

for (int i = 0; i < countFirstNodes; i++) // проход по всем основным папкам (дисциплины) count - подсчет всех дисциплин

{

tmpCount = Convert.ToInt32(treeView1.Nodes[i].GetNodeCount(tmp)); // подсчет количесва веток

for (int j = 0; j < tmpCount; j++) // проход по подветкам (группам)

{

if (treeView1.Nodes[i].Text == reader[2].ToString() && treeView1.Nodes[i].Nodes[j].Text == reader[0].ToString()) // проверка на нужную подветку (на нужную группу)

{

for (int countArray = 0; countArray < countArrayMaterials; countArray++)

{

if (arrayMaterials[countArray].name == reader[0].ToString() && arrayMaterials[countArray].discipline == reader[2].ToString())

{

if (arrayMaterials[countArray].checkPrint(reader[3].ToString()) == false)

{

arrayMaterials[countArray].printToMaterials(reader[3].ToString()); treeView1.Nodes[i].Nodes[j].Nodes.Add(reader[3].ToString());

countElements.countTypeDocuments++;

}

bool secondTmp = false;

int t = Convert.ToInt32(treeView1.Nodes[i].Nodes[j].GetNodeCount(secondTmp));

for (int oneElement = 0; oneElement < Convert.ToInt32(treeView1.Nodes[i].Nodes[j].GetNodeCount(secondTmp)); oneElement++)

{

if (treeView1.Nodes[i].Nodes[j].Nodes[oneElement].Text == reader[3].ToString())

{ treeView1.Nodes[i].Nodes[j].Nodes[oneElement].Nodes.Add(reader[4].ToString() + Path.GetExtension(reader[5].ToString()));

countElements.countDocuments++;

writeNode = true;

break;

}

}

}

}

if (writeNode == true) break;

}

treeView1.EndUpdate();

}

reader.Close();

conn.Close();

}

void getTypeForDocument()

{

bool countFirstNodesFlag = false;

int countFirstNodes = Convert.ToInt32(treeView1.GetNodeCount(countFirstNodesFlag));

treeView1.BeginUpdate();

for (int tmpFirst = 0; tmpFirst < countFirstNodes; tmpFirst++)

{

bool countSecondNodesFlag = false;

int countSecondNodes = Convert.ToInt32(treeView1.Nodes[tmpFirst].GetNodeCount(countSecondNodesFlag));

for (int tmpSecond = 0; tmpSecond < countSecondNodes; tmpSecond++)

{

bool countThirdNodesFlag = false;

int countThirdNodes = Convert.ToInt32(treeView1.Nodes[tmpFirst].Nodes[tmpSecond].GetNodeCount(countThirdNodesFlag));

for (int tmpThird = 0; tmpThird < countThirdNodes; tmpThird++)

{

bool countFourthNodesFlag = false;

int countFourthNodes = Convert.ToInt32(treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].GetNodeCount(countFourthNodesFlag));

for (int tmpFourth = 0; tmpFourth < countFourthNodes; tmpFourth++)

{

string tmpString = treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].Text;

if (tmpString.IndexOf("doc") != -1 || tmpString.IndexOf("docx") != -1 || tmpString.IndexOf("docm") != -1 || tmpString.IndexOf("dotm") != -1 || tmpString.IndexOf("dot") != -1 || tmpString.IndexOf("dotx") != -1)

{

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].SelectedImageKey = "icons8-microsoft-word-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ImageKey = "icons8-microsoft-word-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ContextMenuStrip = contextMenuStrip1;

continue;

}

else

{

if (tmpString.IndexOf("pdf") != -1)

{

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].SelectedImageKey = "icons8-pdf-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ImageKey = "icons8-pdf-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ContextMenuStrip = contextMenuStrip1;

continue;

}

else

{

if (tmpString.IndexOf("xls") != -1 || tmpString.IndexOf("xlsx") != -1 || tmpString.IndexOf("xlsm") != -1 || tmpString.IndexOf("xltx") != -1 || tmpString.IndexOf("xltm") != -1 || tmpString.IndexOf("xlsb") != -1 || tmpString.IndexOf("xlam") != -1 || tmpString.IndexOf("xla") != -1)

{ treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].SelectedImageKey = "icons8-microsoft-excel-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ImageKey = "icons8-microsoft-excel-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ContextMenuStrip = contextMenuStrip1;

continue;

}

else

{

if (tmpString.IndexOf("pptx") != -1 || tmpString.IndexOf("ppt") != -1 || tmpString.IndexOf("pptm") != -1 || tmpString.IndexOf("ppsx") != -1 || tmpString.IndexOf("xltm") != -1 || tmpString.IndexOf("pps") != -1 || tmpString.IndexOf("ppsm") != -1 || tmpString.IndexOf("potx") != -1 || tmpString.IndexOf("pot") != -1 || tmpString.IndexOf("potm") != -1 || tmpString.IndexOf("ppam") != -1 || tmpString.IndexOf("ppa") != -1)

{

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].SelectedImageKey = "icons8-microsoft-powerpoint-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ImageKey = "icons8-microsoft-powerpoint-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ContextMenuStrip = contextMenuStrip1;

continue;

}

else

{

if (tmpString.IndexOf("zip") != -1 || tmpString.IndexOf("7z") != -1 || tmpString.IndexOf("rar") != -1 || tmpString.IndexOf("jap") != -1 || tmpString.IndexOf("gz") != -1)

{

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].SelectedImageKey = "icons8-archive-folder-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ImageKey = "icons8-archive-folder-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ContextMenuStrip = contextMenuStrip1;

continue;

}

else

{

if (tmpString.IndexOf("bmp") != -1 || tmpString.IndexOf("jpg") != -1 || tmpString.IndexOf("gif") != -1 || tmpString.IndexOf("png") != -1 || tmpString.IndexOf("tiff") != -1)

{

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].SelectedImageKey = "icons8-image-file-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ImageKey = "icons8-image-file-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ContextMenuStrip = contextMenuStrip1;

continue;

}

else

{

if (tmpString.IndexOf("mpg") != -1 || tmpString.IndexOf("mov") != -1 || tmpString.IndexOf("wmv") != -1 || tmpString.IndexOf("avi") != -1 || tmpString.IndexOf("mp4") != -1)

{

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].SelectedImageKey = "icons8-video-file-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ImageKey = "icons8-video-file-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ContextMenuStrip = contextMenuStrip1; continue;

}

else

{

if (tmpString.IndexOf("wav") != -1 || tmpString.IndexOf("aif") != -1 || tmpString.IndexOf("mp3") != -1 || tmpString.IndexOf("mid") != -1)

{ treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].SelectedImageKey = "icons8-audio-file-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ImageKey = "icons8-audio-file-48.png";

treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ContextMenuStrip = contextMenuStrip1;

continue;

}

else

{ treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].SelectedImageKey = "icons8-file-48.png"; treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ImageKey = "icons8-file-48.png"; treeView1.Nodes[tmpFirst].Nodes[tmpSecond].Nodes[tmpThird].Nodes[tmpFourth].ContextMenuStrip = contextMenuStrip1;

continue;

}

}

}

}

}

}

}

}

treeView1.EndUpdate();

}

bool seekByName(string tmp)

{

int resSeekByName\_subject = seekByName\_subject(tmp);

if (resSeekByName\_subject != 0)

{

if (resSeekByName\_subject > 1)

{

if (checkOptionsForMessageFound() == true) MessageBox.Show("По даному запиту - \"" + textBox1.Text + "\" знайдено декілька матеріалів(" + resSeekByName\_subject + ")!");

}

return true;

}

else

{

int resSeekByName\_name = seekByName\_name(tmp);

if (resSeekByName\_name != 0)

{

if (resSeekByName\_name > 1)

{

if (checkOptionsForMessageFound() == true) MessageBox.Show("По даному запиту - \"" + textBox1.Text + "\" знайдено декілька матеріалів(" + resSeekByName\_name + ")!");

}

return true;

}

else

{

int resSeekByName\_nameofdocument = seekByName\_nameofdocument(tmp);

if (resSeekByName\_nameofdocument != 0)

{

if (resSeekByName\_nameofdocument > 1)

{

if (checkOptionsForMessageFound() == true) MessageBox.Show("По даному запиту - \"" + textBox1.Text + "\" знайдено декілька матеріалів(" + resSeekByName\_nameofdocument + ")!");

}

return true;

}

else

{

int resSeekByName\_materials = seekByName\_materials(tmp);

if (resSeekByName\_materials != 0)

{

if (resSeekByName\_materials > 1)

{

if (checkOptionsForMessageFound() == true) MessageBox.Show("По даному запиту - \"" + textBox1.Text + "\" знайдено декілька матеріалів(" + resSeekByName\_materials + ")!");

}

return true;

}

else return false;

}

}

}

}

int seekByName\_subject(string tmp)

{

tmp = tmp.ToUpper();

int FlagFindElement = 0;

bool FlagFindFirstElement = false;

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT \* FROM disciplines where discipline\_name LIKE ('%" + relatedWords(tmp) + "%');";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

if (FlagFindFirstElement == false) FlagFindFirstElement = true;

else break;

if (relatedWordsFor\_materials(reader[1].ToString(), tmp) == true)

{

FlagFindElement++;

bool tmpForSeek = false;

for (int i = 0; i < treeView1.GetNodeCount(tmpForSeek); i++)

{

if (relatedWordsFor\_materials(treeView1.Nodes[i].Text, tmp) == true)treeView1.Nodes[i].Toggle();

}

}

}

reader.Close();

conn.Close();

return FlagFindElement;

}

int seekByName\_name(string tmp)

{

int FlagFindElement = 0;

bool FlagFindFirstElement = false;

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT \* FROM users WHERE fio LIKE('%" + relatedWords(tmp) + "%');";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

if (FlagFindFirstElement == false) FlagFindFirstElement = true;

else break;

if (relatedWordsFor\_materials(reader[4].ToString(), tmp) == true)

{

bool tmpForSeek = false;

for (int i = 0; i < treeView1.GetNodeCount(tmpForSeek); i++)

{

for (int j = 0; j < treeView1.Nodes[i].GetNodeCount(tmpForSeek); j++)

{

if (relatedWordsFor\_materials(treeView1.Nodes[i].Nodes[j].Text, tmp) == true)

{

FlagFindElement++;

if (treeView1.Nodes[i].IsExpanded == true) treeView1.Nodes[i].Nodes[j].Toggle();

else

{

treeView1.Nodes[i].Toggle();

treeView1.Nodes[i].Nodes[j].Toggle();

}

}

}

}

}

}

reader.Close();

conn.Close();

return FlagFindElement;

}

int seekByName\_nameofdocument(string tmp)

{

int FlagFindElement = 0;

bool FlagFindFirstElement = false;

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT \* FROM document\_specification WHERE name\_specification LIKE ('%" + relatedWords(tmp) + "%');";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

if (FlagFindFirstElement == false) FlagFindFirstElement = true;

else break;

if (relatedWordsFor\_materials(reader[1].ToString(), tmp) == true)

{

bool tmpForSeek = false;

for (int i = 0; i < treeView1.GetNodeCount(tmpForSeek); i++)

{

for (int j = 0; j < treeView1.Nodes[i].GetNodeCount(tmpForSeek); j++)

{

for (int m = 0; m < treeView1.Nodes[i].Nodes[j].GetNodeCount(tmpForSeek); m++)

{

if (relatedWordsFor\_materials(treeView1.Nodes[i].Nodes[j].Nodes[m].Text, tmp) == true)

{

FlagFindElement++;

if (treeView1.Nodes[i].IsExpanded == true)

{

if (treeView1.Nodes[i].Nodes[j].IsExpanded == true) treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle();

else

{

treeView1.Nodes[i].Nodes[j].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle();

}

}

else

{

treeView1.Nodes[i].Toggle();

treeView1.Nodes[i].Nodes[j].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle();

}

}

}

}

}

}

}

reader.Close();

conn.Close();

return FlagFindElement;

}

int seekByName\_materials(string tmp)

{

int FlagFindElement = 0;

bool FlagFindFirstElement = false;

int []masNumberLastElement = new int[3] { 0, 0, 0 };

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT material.name\_doc\_element, type.name\_type FROM material, type WHERE material.ID\_type = type.ID\_t and name\_doc\_element LIKE ('%" + relatedWords(tmp) + "%');";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

if (FlagFindFirstElement == false) FlagFindFirstElement = true;

else break;

bool tmpForSeek = false;

for (int i = 0; i < treeView1.GetNodeCount(tmpForSeek); i++)

{

for (int j = 0; j < treeView1.Nodes[i].GetNodeCount(tmpForSeek); j++)

{

for (int m = 0; m < treeView1.Nodes[i].Nodes[j].GetNodeCount(tmpForSeek); m++)

{

for (int n = 0; n < treeView1.Nodes[i].Nodes[j].Nodes[m].GetNodeCount(tmpForSeek); n++)

{

if (relatedWordsFor\_materials(treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Text, tmp) == true)

{

FlagFindElement++;

if (treeView1.Nodes[i].IsExpanded == true)

{

if (treeView1.Nodes[i].Nodes[j].IsExpanded == true)

{

if(treeView1.Nodes[i].Nodes[j].Nodes[m].IsExpanded == true) treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Toggle();

else

{

treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Toggle();

}

}

else

{

treeView1.Nodes[i].Nodes[j].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Toggle();

}

}

else

{

treeView1.Nodes[i].Toggle();

treeView1.Nodes[i].Nodes[j].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Toggle();

treeView1.Nodes[i].Nodes[j].Nodes[m].Nodes[n].Toggle();

}

}

}

}

}

}

}

reader.Close();

conn.Close();

return FlagFindElement;

}

string relatedWords(string tmp)

{

string[] tmpStr\_1\_3 = tmp.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

string tmpStr\_1\_3\_2 = "";

foreach (string i in tmpStr\_1\_3)

{

tmpStr\_1\_3\_2 += i;

}

return tmpStr\_1\_3\_2;

}

bool relatedWordsFor\_materials(string firstStr, string secondStr)

{

string tmpStr\_1\_1 = firstStr.ToLower();

string tmpStr\_2\_1 = secondStr.ToLower();

string[] tmpStr\_1\_3 = tmpStr\_1\_1.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

string tmpStr\_1\_3\_2 = "";

foreach (string i in tmpStr\_1\_3)

{

tmpStr\_1\_3\_2 += i;

}

string[] tmpStr\_2\_3 = tmpStr\_2\_1.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

string tmpStr\_2\_3\_2 = "";

foreach (string i in tmpStr\_2\_3)

{

tmpStr\_2\_3\_2 += i;

}

if (firstStr == secondStr) return true;

if (tmpStr\_1\_1 == tmpStr\_2\_1) return true;

if (tmpStr\_1\_3\_2 == tmpStr\_2\_3\_2) return true;

if (tmpStr\_1\_1.IndexOf(tmpStr\_2\_1) != -1) return true;

if (tmpStr\_1\_3\_2.IndexOf(tmpStr\_2\_3\_2) != -1) return true;

return false;

}

int checkTypeDocument(string tmp)

{

int indexOfSubstring = -1;

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT \* FROM type;";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

indexOfSubstring = tmp.IndexOf(reader[1].ToString());

if (indexOfSubstring != -1) break;

}

reader.Close();

conn.Close();

return indexOfSubstring;

}

string createTypeForDocument(string tmp)

{

int position = 0;

for (int i = tmp.Length - 1; i > 0; i--)

{

if (tmp[i] == '.')

{

position = i;

break;

}

}

tmp = tmp.Substring(0, position);

return tmp;

}

conn.Close();

return FlagFindElement;

}

void showFoundNodes()

{

bool tmpForSeek = false;

int countNodes = treeView1.GetNodeCount(tmpForSeek);

if (obTypeSeek.radioButtonSecond == 2)

{

tmpForSeek = false;

countNodes = treeView1.GetNodeCount(tmpForSeek);

int positionRemove = 0;

for (int i = 0; i < countNodes; i++)

{

if (treeView1.Nodes[positionRemove].IsExpanded == true) positionRemove++;

else treeView1.Nodes[positionRemove].Remove();

}

}

}

bool checkOptionsForMessageFound()

{

if (checkBox6.Checked == true) return true;

else return false;

}

bool checkOptionsForMessageNotFound()

{

if (checkBox7.Checked == true) return true;

else return false;

}

void seekFocusNode(string tmp)

{

string tmpSecond = tmp.Substring(10);

tmpSecond = deleteTypeDocument(tmpSecond);

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT users.fio, disciplines.discipline\_name, document\_specification.name\_specification, material.name\_doc\_element, type.name\_type, tags.tag\_name FROM material, document\_specification, studyload, users, disciplines, material\_tag, tags, type WHERE material.id\_specification = document\_specification.id\_d\_s and studyload.id\_s\_l = material.id\_study\_load and users.id\_u = studyload.id\_user and disciplines.id\_d = studyload.id\_discipline and material.id\_m = material\_tag.id\_material and material\_tag.id\_tag = tags.id\_tag and material.id\_type = type.id\_t and material.name\_doc\_element LIKE('%" + tmpSecond + "%');";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

bool tmpForSeek = false;

for (int i = 0; i < treeView1.GetNodeCount(tmpForSeek); i++)

{

if (treeView1.Nodes[i].Text == reader[1].ToString())

{

for (int j = 0; j < treeView1.Nodes[i].GetNodeCount(tmpForSeek); j++)

{

if (treeView1.Nodes[i].Nodes[j].Text == reader[0].ToString())

{

for (int m = 0; m < treeView1.Nodes[i].Nodes[j].GetNodeCount(tmpForSeek); m++)

{

void openFileInWay(string way1, string way2, string way3, string way4)

{

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT users.fio, disciplines.discipline\_name, document\_specification.name\_specification, material.name\_doc\_element, material.way FROM material, document\_specification, studyload, users, disciplines WHERE material.id\_specification = document\_specification.id\_d\_s and studyload.id\_s\_l = material.id\_study\_load and users.id\_u = studyload.id\_user and disciplines.id\_d = studyload.id\_discipline;";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

if(way1 == reader[1].ToString() && way2 == reader[0].ToString() && way3 == reader[2].ToString() && way4 == reader[3].ToString())

{

try

{

Process.Start(reader[4].ToString());

}

catch

{

MessageBox.Show("Документ не знайдено!");

}

break;

}

}

reader.Close();

conn.Close();

}

}

}

Лістинг Ґ.2 – Форма «Редагування»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace TreeNodeInMainMenu

{

public partial class FormRedaction : Form

{

InformationAboutDocumentAll[] masTypeDocument = new InformationAboutDocumentAll[50];

int countMasTypeDocument = 0;

int positionType = 0;

InformationAboutDocumentAll[] masDisciplineDocument = new InformationAboutDocumentAll[10];

int countMasDisciplineDocument = 0;

int positionDiscipline = 0;

InformationAboutDocumentAll[] masKeyWords = new InformationAboutDocumentAll[100];

int countKeyWords = 0;

int positionKeyWords = 0;

InformationAboutDocumentAll[] masOldKeyWords = new InformationAboutDocumentAll[50];

int countOldKeyWords = 0;

int positionOldKeyWords = 0;

public FormRedaction()

{

InitializeComponent();

}

private void FormRedaction\_Load(object sender, EventArgs e)

{

toolStripStatusLabel6.Text += Form1.strWay[1];

toolStripStatusLabel1.Text += Form1.strWay[0];

toolStripStatusLabel2.Text += Form1.strWay[3];

toolStripStatusLabel4.Text += Form1.strWay[4];

toolStripStatusLabel3.Text += Form1.strWay[2];

createMasType();

comboBox1.Text = Form1.strWay[2];

checkNewType();

createMasDiscipline();

comboBox2.Text = Form1.strWay[1];

checkNewDiscipline();

createMasKeyWords();

createMasOldKeywords();

}

private void menuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

private void statusStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

checkNewType();

checkNewDiscipline();

if (checkElementWithSameName() == true) MessageBox.Show("В одному файлі не може бути документів зі схожими назвами!", "Повідомлення");

else

{

changeInformationAboutDocument();

changeInformationAboutKeywords();

Close();

}

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBox1.SelectedItem != null)

{

DialogResult resMessange = MessageBox.Show("Бажаєте додати до документу ключове слово - " + listBox1.SelectedItem.ToString() + "?", "Підтвердження дій", MessageBoxButtons.YesNo);

if (resMessange == DialogResult.Yes)

{

if (checkDocumentSimilarity(listBox1.SelectedItem.ToString()) == false)

{

MessageBox.Show("Ключове слово успішно додано!");

masOldKeyWords[countOldKeyWords] = new InformationAboutDocumentAll();

masOldKeyWords[countOldKeyWords].id = seekInOneElementInMasKeyWords(listBox1.SelectedItem.ToString());

masOldKeyWords[countOldKeyWords].NewElement = true;

masOldKeyWords[countOldKeyWords++].keyword = listBox1.SelectedItem.ToString();

listBox2.Items.Add(listBox1.SelectedItem.ToString());

}

else MessageBox.Show("Дане ключове слово вже є у списку ключових слів, які прив’язані к документу!", "Повідомлення");

}

}

}

private void listBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBox2.SelectedItem != null)

{

DialogResult resMessange = MessageBox.Show("Бажаєте додати до документу ключове слово - " + listBox2.SelectedItem.ToString() + "?", "Підтвердження дій", MessageBoxButtons.YesNo);

if (resMessange == DialogResult.Yes)

{

if (deleteElementFromMasOldkeywords(listBox2.SelectedItem.ToString()) == true) MessageBox.Show("Ключове слово успішно видалено!");

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

deleteAllElementInListBox1();

createMasKeyWords();

seekInListBox1(textBox1.Text);

}

private void ToolStripMenuItem1\_Click(object sender, EventArgs e)

{

DialogResult resMessange = MessageBox.Show("Ви дійсно бажаєте залишити форму?", "Підтвердження дій", MessageBoxButtons.YesNo);

if (resMessange == DialogResult.Yes)

{

Close();

}

}

void createMasType()

{

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT \* FROM diploma.document\_specification;";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

masTypeDocument[countMasTypeDocument] = new InformationAboutDocumentAll();

masTypeDocument[countMasTypeDocument].id = Convert.ToInt32(reader[0]);

masTypeDocument[countMasTypeDocument++].type = reader[1].ToString();

comboBox1.Items.Add(reader[1].ToString());

}

reader.Close();

conn.Close();

}

void checkNewType()

{

for (int i = 0; i < countMasTypeDocument; i++)

{

if (comboBox1.Text == masTypeDocument[i].type)

{

positionType = i;

break;

}

}

}

void changeInformationAboutDocument()

{

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "UPDATE material SET ID\_specification = " + masTypeDocument[positionType].id + ", ID\_study\_load = " + masDisciplineDocument[positionDiscipline].id + " WHERE ID\_m = " + Form1.strWay[5] + ";";

MySqlCommand command = new MySqlCommand(sql, conn);

command.ExecuteNonQuery();

conn.Close();

MessageBox.Show("Документ - " + Form1.strWay[3] + " успішно змінено!");

}

void changeInformationAboutKeywords()

{

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "DELETE FROM material\_tag WHERE ID\_material = " + Form1.strWay[5] + ";";

MySqlCommand command = new MySqlCommand(sql, conn);

command.ExecuteNonQuery();

conn.Close();

conn.Open();

for (int i = 0; i < countOldKeyWords; i++)

{

sql = "INSERT INTO material\_tag(ID\_material, ID\_tag) VALUES('" + Form1.strWay[5] + "', '" + masOldKeyWords[i].id + "');";

command = new MySqlCommand(sql, conn);

command.ExecuteNonQuery();

}

conn.Close();

}

void createMasDiscipline()

{

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT ID\_s\_l, ID\_discipline, discipline\_name FROM studyload, users, disciplines WHERE studyload.ID\_user = users.ID\_u and studyload.ID\_discipline = disciplines.ID\_d and fio LIKE ('%" + Form1.strWay[0] + "%');";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

masDisciplineDocument[countMasDisciplineDocument] = new InformationAboutDocumentAll();

masDisciplineDocument[countMasDisciplineDocument].id = Convert.ToInt32(reader[0]);

masDisciplineDocument[countMasDisciplineDocument++].discipline = reader[2].ToString();

comboBox2.Items.Add(reader[2].ToString());

}

reader.Close();

conn.Close();

}

void checkNewDiscipline()

{

for (int i = 0; i < countMasDisciplineDocument; i++)

{

if (comboBox2.Text == masDisciplineDocument[i].discipline)

{

positionDiscipline = i;

break;

}

}

}

void createMasKeyWords()

{

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT \* FROM diploma.tags;";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

masKeyWords[countKeyWords] = new InformationAboutDocumentAll();

masKeyWords[countKeyWords].id = Convert.ToInt32(reader[0]);

masKeyWords[countKeyWords++].keyword = reader[1].ToString();

listBox1.Items.Add(reader[1].ToString());

}

reader.Close();

conn.Close();

}

void seekInListBox1(string tmp)

{

int position = 0;

int countElementInListBox = listBox1.Items.Count;

for (int i = 0; i < countElementInListBox; i++)

{

if (relatedWordsFor\_materials(listBox1.Items[position].ToString(), tmp) == false) listBox1.Items.RemoveAt(position);

else position++;

}

}

int seekInOneElementInMasKeyWords(string tmp)

{

for (int i = 0; i < countKeyWords; i++)

{

if (masKeyWords[i].keyword == tmp) return masKeyWords[i].id;

}

return 0;

}

void deleteAllElementInListBox1()

{

listBox1.Items.Clear();

for (int i = 0; i < countKeyWords; i++) masKeyWords[i] = null;

countKeyWords = 0;

}

bool relatedWordsFor\_materials(string firstStr, string secondStr)

{

string tmpStr\_1\_1 = firstStr.ToLower();

string tmpStr\_2\_1 = secondStr.ToLower();

string[] tmpStr\_1\_3 = tmpStr\_1\_1.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

string tmpStr\_1\_3\_2 = "";

foreach (string i in tmpStr\_1\_3)

{

tmpStr\_1\_3\_2 += i;

}

string[] tmpStr\_2\_3 = tmpStr\_2\_1.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

string tmpStr\_2\_3\_2 = "";

foreach (string i in tmpStr\_2\_3)

{

tmpStr\_2\_3\_2 += i;

}

if (firstStr == secondStr) return true;

if (tmpStr\_1\_1 == tmpStr\_2\_1) return true;

if (tmpStr\_1\_3\_2 == tmpStr\_2\_3\_2) return true;

if (tmpStr\_1\_1.IndexOf(tmpStr\_2\_1) != -1) return true;

if (tmpStr\_1\_3\_2.IndexOf(tmpStr\_2\_3\_2) != -1) return true;

return false;

}

void createMasOldKeywords()

{

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT ID\_material, tags.ID\_tag, Tag\_name FROM material\_tag, tags WHERE material\_tag.ID\_tag = tags.ID\_tag and ID\_material = " + Form1.strWay[5] + ";";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

masOldKeyWords[countOldKeyWords] = new InformationAboutDocumentAll();

masOldKeyWords[countOldKeyWords].id = Convert.ToInt32(reader[1]);

masOldKeyWords[countOldKeyWords].NewElement = false;

masOldKeyWords[countOldKeyWords++].keyword = reader[2].ToString();

listBox2.Items.Add(reader[2].ToString());

}

reader.Close();

conn.Close();

}

bool checkDocumentSimilarity(string tmp)

{

int countSimilarity = 0;

for (int i = 0; i < countOldKeyWords; i++)

{

if(masOldKeyWords[i].keyword == tmp)

{

countSimilarity++;

break;

}

}

if (countSimilarity == 0) return false;

else return true;

}

bool deleteElementFromMasOldkeywords(string tmp)

{

for (int i = 0; i < countOldKeyWords; i++)

{

if (masOldKeyWords[i].keyword == tmp)

{

positionOldKeyWords = i;

break;

}

}

if (listBox2.Items.Count > 1)

{

InformationAboutDocumentAll tmpObForSwapInformation = new InformationAboutDocumentAll();

tmpObForSwapInformation.id = masOldKeyWords[positionOldKeyWords].id;

tmpObForSwapInformation.keyword = masOldKeyWords[positionOldKeyWords].keyword;

masOldKeyWords[positionOldKeyWords].id = masOldKeyWords[countOldKeyWords - 1].id;

masOldKeyWords[positionOldKeyWords].keyword = masOldKeyWords[countOldKeyWords - 1].keyword;

masOldKeyWords[countOldKeyWords - 1].id = tmpObForSwapInformation.id;

masOldKeyWords[countOldKeyWords - 1].keyword = tmpObForSwapInformation.keyword;

masOldKeyWords[countOldKeyWords - 1] = null;

countOldKeyWords--;

for (int i = 0; i < listBox2.Items.Count; i++)

{

if (listBox2.Items[i].ToString() == tmp)

{

listBox2.Items.RemoveAt(i);

break;

}

}

return true;

}

else

{

MessageBox.Show("Документ має включати хоча б одне ключове слово!", "Повідомлення");

return false;

}

}

bool checkElementWithSameName()

{

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;charset=cp1251";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT ID\_m, Name\_doc\_element FROM material WHERE ID\_specification = " + masTypeDocument[positionType].id + " and ID\_study\_load = " + masDisciplineDocument[positionDiscipline].id + " and Name\_doc\_element = '" + Form1.strWay[3] + "';";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

if (Form1.strWay[5] != reader[0].ToString()) return true;

}

reader.Close();

conn.Close();

return false;

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

Лістинг Ґ.3 – Форма «Статистичні дані»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace FormSeek

{

public partial class InformationInTable : Form

{

ClassInfoInTable[] masOb = new ClassInfoInTable[50];

int countMasOb = 0;

int []masTopCountSubject = new int[5] { 0, 0, 0, 0, 0 };

int countSubject = 0;

int[] masTopCountMaterials = new int[5] { 0, 0, 0, 0, 0 };

int countMaterials = 0;

int chartChoice;

public InformationInTable()

{

InitializeComponent();

}

private void InformationInTable\_Load(object sender, EventArgs e)

{

createMasOb();

inputDataGridView();

createChart1();

createChart2();

chart1.Visible = true;

chart2.Visible = false;

chartChoice = 1;

chart1.Series["Дисципліни"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;

}

public void createMasOb()

{

string connStr = "server=localhost;user=root;database=diploma;password=s1233212;";

MySqlConnection conn = new MySqlConnection(connStr);

conn.Open();

string sql = "SELECT users.id\_u, users.fio FROM users, roles WHERE roles.id\_r = users.id\_role and roles.id\_r = 2;";

MySqlCommand command = new MySqlCommand(sql, conn);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

masOb[countMasOb] = new ClassInfoInTable();

masOb[countMasOb].id\_fio = Convert.ToInt32(reader[0]);

masOb[countMasOb].fio = reader[1].ToString();

countMasOb++;

}

reader.Close();

sql = "SELECT users.id\_u, users.fio, count(disciplines.discipline\_name) FROM users, studyload, disciplines WHERE users.id\_u = studyload.id\_user and disciplines.id\_d = studyload.id\_discipline GROUP BY users.fio;";

command = new MySqlCommand(sql, conn);

reader = command.ExecuteReader();

while (reader.Read())

{

for (int i = 0; i < countMasOb; i++)

{

if (Convert.ToInt32(reader[2]) > 0)

{

if (masOb[i].id\_fio == Convert.ToInt32(reader[0]))

{

masOb[i].countSubject = Convert.ToInt32(reader[2]);

inputMasTopSubject(Convert.ToInt32(reader[2]));

}

}

else break;

}

}

reader.Close();

sql = "SELECT users.id\_u, users.fio, count(material.name\_doc\_element) FROM users, studyload, disciplines, material WHERE users.id\_u = studyload.id\_user and disciplines.id\_d = studyload.id\_discipline and studyload.id\_s\_l = material.id\_study\_load GROUP BY users.fio;";

command = new MySqlCommand(sql, conn);

reader = command.ExecuteReader();

while (reader.Read())

{

for (int i = 0; i < countMasOb; i++)

{

if (Convert.ToInt32(reader[2]) > 0)

{

if (masOb[i].id\_fio == Convert.ToInt32(reader[0]))

{

masOb[i].countMaterials = Convert.ToInt32(reader[2]);

inputMasTopMaterials(Convert.ToInt32(reader[2]));

}

}

else break;

}

}

reader.Close();

conn.Close();

}

public void inputDataGridView()

{

for (int i = 0; i < countMasOb; i++)

{

dataGridView1.Rows.Add(masOb[i].fio, masOb[i].countSubject, masOb[i].countMaterials);

}

}

public void createChart1()

{

countSubject = 0;

for (int i = 0; i < countMasOb; i++)

{

if (masOb[i].countSubject > masTopCountSubject[4] && countSubject < 10)

{

chart1.Series["Дисципліни"].Points.AddXY(masOb[i].fio, masOb[i].countSubject);

countSubject++;

}

else

{

if (countSubject > 11) break;

}

}

}

public void createChart2()

{

countMaterials = 0;

for (int i = 0; i < countMasOb; i++)

{

if (masOb[i].countMaterials > masTopCountMaterials[4] && countMaterials < 10)

{

chart2.Series["Документи"].Points.AddXY(masOb[i].fio, masOb[i].countMaterials);

countMaterials++;

}

else

{

if (countMaterials > 11) break;

}

}

}

public void changeChart()

{

if (chartChoice == 1)

{

chart1.Visible = true;

chart2.Visible = false;

}

else

{

chart2.Visible = true;

chart1.Visible = false;

}

}

private void pieToolStripMenuItem1\_Click(object sender, EventArgs e)

{

chartChoice = 1;

chart1.Series["Дисципліни"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;

changeChart();

}

private void columnToolStripMenuItem1\_Click(object sender, EventArgs e)

{

chartChoice = 1;

chart1.Series["Дисципліни"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;

changeChart();

}

private void barToolStripMenuItem1\_Click(object sender, EventArgs e)

{

chartChoice = 1;

chart1.Series["Дисципліни"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Bar;

changeChart();

}

private void barToolStripMenuItem1\_Click(object sender, EventArgs e)

{

chartChoice = 1;

chart1.Series["Дисципліни"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Bar;

changeChart();

}

private void pieToolStripMenuItem2\_Click(object sender, EventArgs e)

{

chartChoice = 2;

chart2.Series["Документи"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;

changeChart();

}

private void doughnutToolStripMenuItem2\_Click(object sender, EventArgs e)

{

chartChoice = 2;

chart2.Series["Документи"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Doughnut;

changeChart();

}

private void columnToolStripMenuItem2\_Click(object sender, EventArgs e)

{

chartChoice = 2;

chart2.Series["Документи"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;

changeChart();

}

private void pointToolStripMenuItem2\_Click(object sender, EventArgs e)

{

chartChoice = 2;

chart2.Series["Документи"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;

changeChart();

}

Pr

ivate void barToolStripMenuItem2\_Click(object sender, EventArgs e)

{

chartChoice = 2;

chart2.Series["Документи"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Bar;

changeChart();

}

private void barToolStripMenuItem2\_Click(object sender, EventArgs e)

{

chartChoice = 2;

chart2.Series["Документи"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Bar;

changeChart();

}

private void barToolStripMenuItem2\_Click(object sender, EventArgs e)

{

chartChoice = 2;

chart2.Series["Документи"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Bar;

changeChart();

}

private void barToolStripMenuItem2\_Click(object sender, EventArgs e)

{

chartChoice = 2;

chart2.Series["Документи"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Bar;

changeChart();

}

private void ToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Hide();

}

void inputMasTopMaterials(int tmp)

{

if(countMaterials < 5)

{

masTopCountMaterials[countMaterials] = tmp;

countMaterials++;

}

else

{

sortMasTopMaterials();

if (tmp > masTopCountMaterials[0])

{

masTopCountMaterials[4] = masTopCountMaterials[3];

masTopCountMaterials[3] = masTopCountMaterials[2];

masTopCountMaterials[2] = masTopCountMaterials[1];

masTopCountMaterials[1] = masTopCountMaterials[0];

masTopCountMaterials[0] = tmp;

}

}

}

void sortMasTopMaterials()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

if(masTopCountMaterials[j] < masTopCountMaterials[j+1])

{

int tmp = masTopCountMaterials[j];

masTopCountMaterials[j] = masTopCountMaterials[j+1];

masTopCountMaterials[j] = tmp;

}

}

}

}

void inputMasTopSubject(int tmp)

{

if (countSubject < 5)

{

masTopCountSubject[countSubject] = tmp;

countSubject++;

}

else

{

sortMasTopSubject();

if (tmp > masTopCountSubject[0])

{

masTopCountSubject[4] = masTopCountSubject[3];

masTopCountSubject[3] = masTopCountSubject[2];

masTopCountSubject[2] = masTopCountSubject[1];

masTopCountSubject[1] = masTopCountSubject[0];

masTopCountSubject[0] = tmp;

}

}

}

void sortMasTopSubject()

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 4; j++)

{

if (masTopCountSubject[j] < masTopCountSubject[j+1])

{

int tmp = masTopCountSubject[j];

masTopCountSubject[j] = masTopCountSubject[j+1];

masTopCountSubject[j+1] = tmp;

}

}

}

}

}

}