ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Кафедра інженерії програмного забезпечення

**КУРСОВИЙ ПРОЄКТ**

з дисципліни

«Вебтехнології»

на тему: Розробка сервісу для читання і покупки книг

**КПІПЗ.**

Галузь знань . 12 «Інформаційні технології» .

Спеціальність . 121 «Інженерія програмного забезпечення»

(шифр, назва)

Студента І . курсу, група . ІПЗс-25-1. \_\_\_\_\_\_\_\_\_\_ Дмитро ДОСКОЧ

(шифр) (підпис) (ініціали, прізвище)

Керівник . доцент, канд.техн.наук.. \_\_\_\_\_\_\_\_\_\_\_ Оксана ЯШИНА

(посада, вчене звання, науковий ступінь) (підпис) (ініціали, прізвище)

Кількість балів \_\_\_\_\_\_\_

Оцінка за шкалою:

національною\_\_\_\_\_\_\_\_\_/ЄКТС\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ініціали, прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ініціали, прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ініціали, прізвище)

Хмельницький 2025

*Змін.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Розробив*

*Перевірив*

*Затвер.*

*Н.контр.*

*Літ*

*Аркуш*

*Аркушів*

**ЗМІСТ**

[АНОТАЦІЯ 4](#_Toc210291237)

[**ВСТУП** 5](#_Toc210291238)

[**РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ СЕРВІСУ ДЛЯ ЧИТАННЯ ТА ПОКУПКИ КНИГ** 7](#_Toc210291239)

[2 Розробка технічного та робочого проекту 8](#_Toc210291240)

[**2.1 Опис та об*ґ*рунтування вибору структури та методу організації вхідних та вихідних даних** 8](#_Toc210291241)

[**2.2 Розробка структурної схеми проєкту та об’єктної моделі задачі на основі UML діаграм** 13](#_Toc210291242)

[**2.3 Об’єктно-орієнтоване проєктування, програмування та тестування** 18](#_Toc210291243)

[Спеціальний розділ 28](#_Toc210291244)

[**3.1 Інструкція з інсталяції розробленого проєкту** 28](#_Toc210291245)

[**3.2 Інструкція з експлуатації розробленого проєкту** 28](#_Toc210291246)

[Додаток А: Код проекту 34](#_Toc210291247)

Список використаних джерел…………………………………………….35

ДОДАТОК А КОД ПРОЄКТУ……………………………………………36

Доскоч Дмитро

Руденко Тетяна

Алєксєєв Микола

]

# АНОТАЦІЯ

Курсовий проект «Сервіс для читання і покупки книг».

Автор роботи: Доскоч Д.С.

Керівник роботи: Яшина О.М.

Обсяг – с., рис., табл., додатки, джерело.

Ключові слова:

Мета курсового проекту: розробити сервіс для читання і покупки книг, що дозволить користувачам купувати та відразу на вебресурсі читати електронні версії книг, дозволить за допомогою перекладача вбудованого в сайт, автоматично перекладати книги на необхідну для них мову, та надати можливість починаючим авторам знаходити свою аудиторію.

Практичне значення одержаних результатівполягає у створенні зручного сервісу для читання та придбання електронних книг безпосередньо на вебресурсі, без необхідності виконувати непотрібні дії. Також деякі книги будуть безкоштовні для читання, що, спрощує користувачам доступ до літератури. Інтегрований перекладач усуне мовний бар’єр і дозволить користувачам читати різні видання, що не мають зрозумілої читачу локалізації. Підтримка починаючих авторів дозволить їм і знайти аудиторію.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(дата) (підпис)

**ВСТУП**

Тема курсового проєкту полягає у створенні вебсервісу, який дозволить користувачам читати та купувати книги на одному ресурсі.

Об’єктом дослідження є електронні сервіси для читання та покупки книг.

Предметом дослідження є функціональні можливості сервісів та необхідності користувачів у процесі читання електронних книг

Сьогодні кожен аспект суспільства так чи інакше підлягає цифровізації, від державних послуг до роботи. Звісно така сфера як література, не могла залишитись осторонь від цього, зараз дуже активно набуває популярності читання саме електронних книг, адже це і зручно, і дешевше ніж читання фізичних видань, та створює більшу гнучкість, від самостійного вибору мови, до вибору того чи іншого перекладу.

Через більшу зручність та меншу вартість саме сервіси для покупки та читання електронних книг набувають все більшої популярності, як серед читачів, які зацікавлені у цьому через зручність, так і серед авторів, які можуть на таких сервісах знайти свою початкову аудиторію, та отримати реальні відгуки про свої твори, без необхідності попередньо шукати видавця, який погодиться видати його книгу.

У зв’язку з великим попитом ми маємо надати користувачам необхідний для них ключовий функціонал, який буде реалізовано у нашому сервісі. Основним функціоналом сервісу будуть такі функції: можливість придбати книгу, читати книгу відразу на вебресурсі, перекласти на необхідну для користувача мову за допомогою вбудованого перекладача, якщо такого перекладу не існує, також функціонал для авторів, який дозволить виставляти свої твори на сайт, та отримувати статистику по читачам.

Більшість цих функцій в тій чи іншій мірі також присутні і на інших сайтах схожої направленості, але загалом вони там реалізовані лише частково, і сукупно цих функцій немає в жодному українському аналогу, і в майже усіх іноземних.

Отже, розробка сервісу такої направленості є актуальною та доцільною під сучасні реалії, не тільки в Україні, а й при локалізації та грамотному піарі з можливістю потрапити ще й на іноземний ринок. Також варто зазначити відносно малу кількість конкурентів, адже зачасту такі послуги надають лише частково книжкові магазини, які все ж в більшості орієнтовані на продаж саме фізичних видань, і електронні видання для них є лише субпродуктом.

**РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ СЕРВІСУ ДЛЯ ЧИТАННЯ ТА ПОКУПКИ КНИГ**

Призначенням сайту є надати можливість користувачам доступ на платній, чи безоплатній, для деяких книг, основі до літератури в електронному форматі, що є зараз дуже затребуваним, особливо у час цифровізації кожного аспекту життя.

У

# 2 Розробка технічного та робочого проекту

## **2.1 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних**

Перед початком розробки програми необхідно ретельно продумати її архітектуру, структуру даних та інтерфейс, оскільки це визначає основні принципи роботи програми і впливає на її майбутню масштабність та підтримку. Належне планування дозволяє зменшити кількість помилок у процесі розробки та експлуатації.

У даній програмі дані будуть записуватися у json файл. JSON це – текстовий формат обміну даними між комп’ютерами.JSON базується на даних, які може прочитати людина. Цей формат зачасту викорисовується для мережевих передавань, але за його зручність набув широкої популярності і у простих цілях.

Переваги використання JSON файлів:

* читабельність: файли JSON легко прочитати для людини завдяки їх представленню у вигляді зрозумілих ключів і значень ;
* проста інтерпретація:JSON є популярним форматом, який підтримується багатьма мовами програмування та легко інтерпретується у багатьох додатках та веб застосунках ;
* гнучкість: JSON файли можуть зберігати структуровані дані, забезпечуючи гнучкість у представленні інформації
* легка розширюваність: JSON файл дозволяє легко додавати нові поля до існуючих структур.

У файли буде записуватись наступна історія:

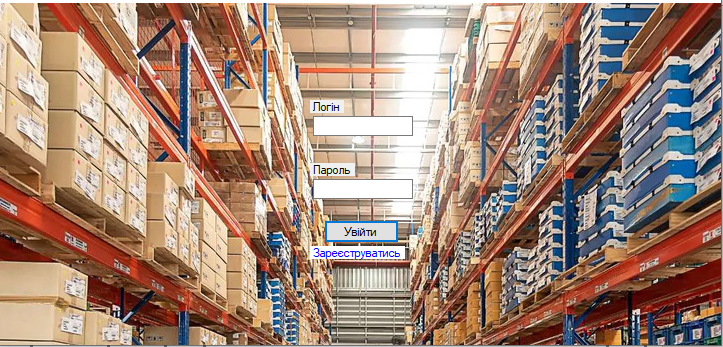
* товари на складі: перелік товарів на складі та їх властивостей;
* довідник величин: перелік величин, їх скорочень та пояснень;
* постійні постачальники: перелік постійних постачальників, з їхньою контактною інформацією;
* запити постачання: запити від постачальників на поставку певного товару на склад.

Програма забезпечуватиме функціонал для збереження результатів у JSON файли. Також буде реалізовано можливість відкривати раніше збережені файли для перегляду та редагування даних. Це дозволить виправляти певні неточності чи інші ексцеси, які можуть виникнути під час використання програми.

Таким чином, використання JSON файлів для зберігання даних у програмі забезпечує ефективне управління інформацією, гнучкість під час роботи з програмою та легше збереження цілих структур і масивів даних.

Вибір правильного макету проєкту – це важливий стратегічний крок, який має далекосяжні наслідки для успіху програмного забезпечення. Впливає на зручність для користувача, організацію та підтримку коду, масштабованість, ефективність розробки, сумісність, забезпечення якості. Приділення належної уваги цьому аспекту на початковому етапі розробки може зберегти знайти час і зусилля у майбутньому, забезпечуючи більш успішний та якісний кінцевий продукт.

Початкове вікно матиме наступний макет (див рис.2.2):



Риcунок 2.2– макет початкового вікна програми

На цій формі користувач діє стандартно, заходячи у свій аккаунт, якщо він є, або реєструючись якщо його немає.

Далі потрібно створити форму для переходу між різними функціями програми (див. рис.2.3)

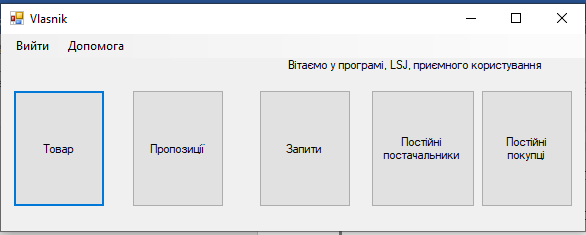


Рисунок 2.3 - макет для вибору

Форма надає макет для переходу між різними функціями програми для власника складу.

Ми використовуємо наступні елементи Toolbox:

* menustrip для введення меню зверху;
* кнопка «Товар» виконує перехід до форми, де власник складу буде керувати товаром;
* кнопка «Пропозиції» дозволяє перейти до форми з пропозиціями від постачальників;
* кнопка «Запити» показує користувачу запити від користувачів на покупку певних речей на складі;
* кнопка «Постійні покупці» дозволяє перейти до форми з постійними покупцями;
* кнопка «Постійні поcтачальники» дозволяє перейти до форми з постійними постачальниками;
* зміни у введені параметри.

Форма має чіткий і зрозумілий інтерфейс, де кожен елемент має свою роль і легко доступний для користувача.

Наступий крок – створення форми(див. рис.2.4) управління товаром на складі

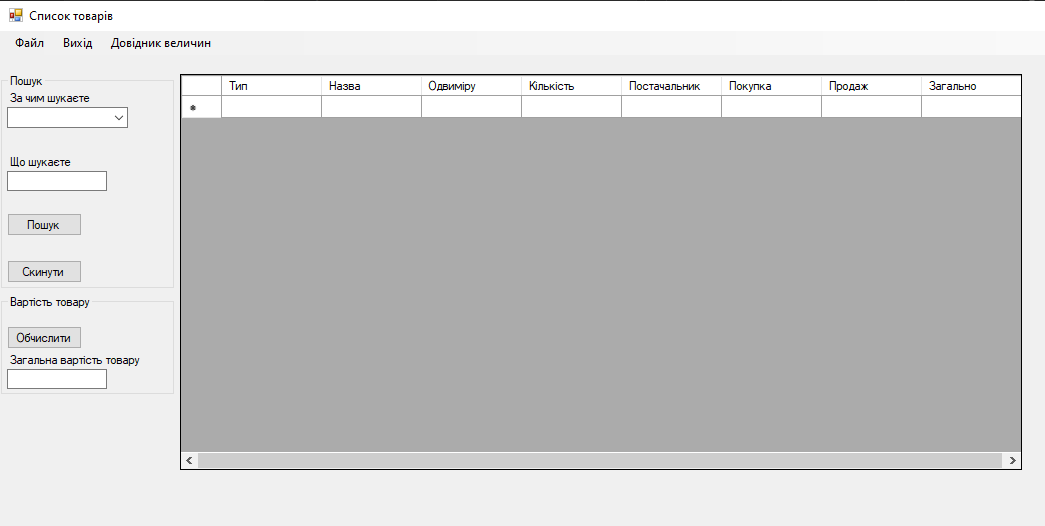


Рисунок 2.4 - макет для ведення товарів

Форма дозволяє користувачам вводити товари на склади, сортувати, шукати певні.

Форма включає кнопки для різних дій :

* «Пошук» знаходить за обраним параметром певні товари;
* «Скинути» скидає результати пошуку;
* «Обчислити» обчислює загальну вартість товарів на складі;
* «Довідник величин» відкриває форму довідник величин, в якій величини, їх скорочення та пояснення
* «Файл» включатиме в себе 3 кнопки, а саме зберегти файл, зберегти файл як, та відкрити інший файл.

Присутні також таблиця, для візуалізації обліку, comboBox для вибору за яким стовпцем буде відбуватись пощук, та textbox для виведення загальної вартості товарів на складі.

Чітке розділення функціональностей робить форму зручною для використання та розуміння. Користувачі можуть легко слідувати крокам для введення даних та виконання роботи.

Створимо форму для довідника величин(рис.2.5):

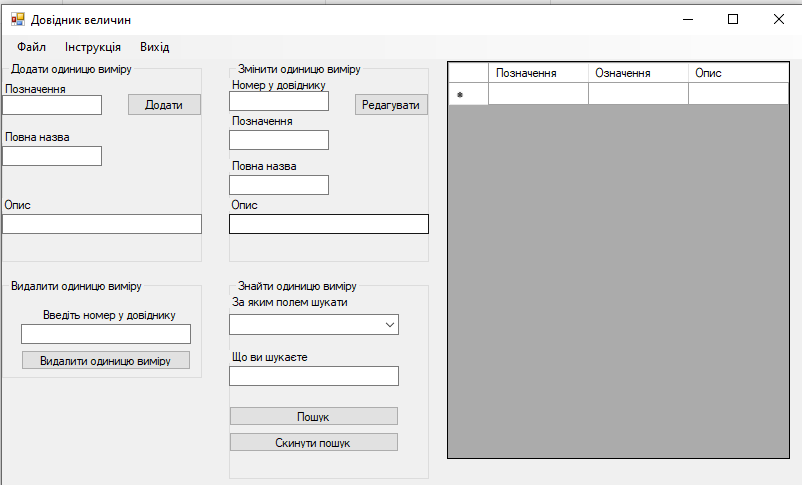


Рисунок 2.5 - макет для введення назви будинку

Використовуємо наступні елементи:

-textBox це текстове поле, в цьому вікні їх 9, завдяки ним користувач може виконувати зазначені на макети функції з величинами;

- кнопка "Пошук" дозволяє шукати певну одиницю вимірювання;

-кнопка "Скасувати пошук" дозволяє скасувати пошук та відобразити всі одиниці вимірювання;

-кнопка "Видалити" дозволяє видаляти одиниці вимірювання з довідника;

-кнопка "Редагувати" дозволяє редагувати певні одиниці вимірювання;

-кнопка "Додати" дозволяє створювати нові одиниці вимірювання.

Важливим кроком є створення форми для пропозицій від постачальників(рис.2.6) Форма потрібна для того, щоб можна було переглядати та приймати, чи відмовляти у відповідь на пропозиції постачати щось на склад:

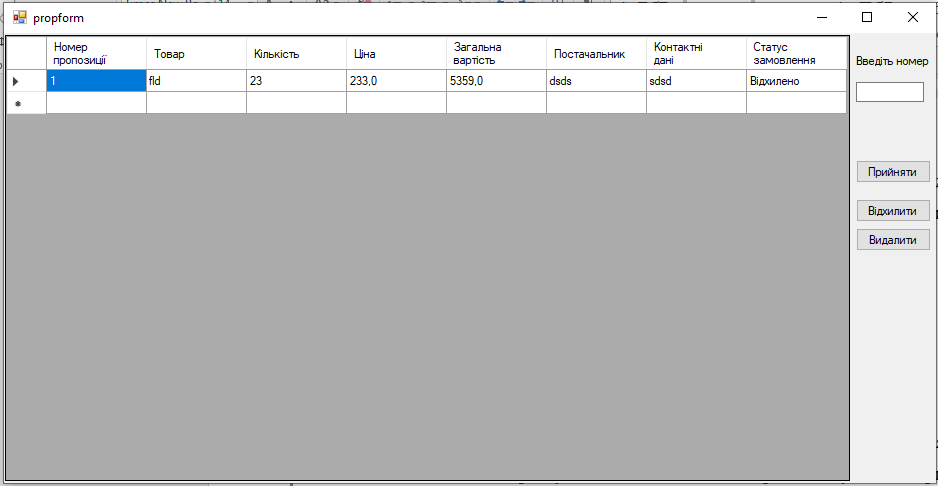


Рисунок 2.6 - макет форми для пропозицій постачальників

Для зручності користувача було додано наступні елементи:

* DatagriedView1 для перегляду пропозицій;
* кнопка "Прийняти" дозволяє прийняти пропозицію і перевести пропозицію у стан прийнятої;
* кнопка "Відхилити" дозволяє відхилити пропозицію від постачальника і перевести її у склад віхиленої;
* кнопка "Видалити" дозволяє видаляти пропозиції які втратили свою актуальність;

Ці елементи забезпечують користувачам зручний інтерфейс для управління товарообігом. Вони дозволяють виконувати різні дії та маніпулювати даними, що робить користування програмою більш ефективним та зручним.

## **2.2 Розробка структурної схеми проєкту та об’єктної моделі задачі на основі UML діаграм**

У даному розділі ми розглянемо процес розробки структурної схеми проєкту та об’єктної моделі задачі, використовуючи діаграми мовою моделювання UML (Unified Modeling Language). UML є стандартом для об'єктного моделювання і широко застосовується в програмній інженерії для візуалізації, специфікації, конструювання та документування програмних систем. Метою цього розділу є представлення всіх ключових аспектів проєкту за допомогою UML діаграм, що допоможе забезпечити чітке розуміння структури та взаємодії між різними компонентами системи.. Основні серед них:

* діаграма варіантів використання (Use Case Diagrams)(рис.2.7) ця діаграма дозволяє ідентифікувати всі можливі дії користувачів та системи, а також їх взаємодію. У випадку застосунку актором буде власник складу, який матиме можливість виконувати перелічені на діаграмі дії.

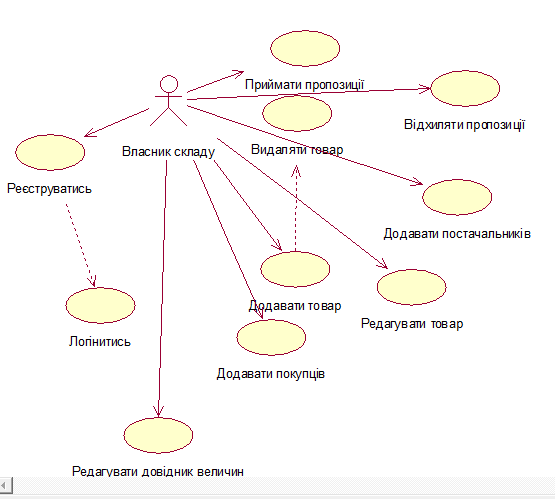


Рисунок 2.7 - Діаграма варіантів використання

* діаграма класів (Class Diagrams)(рис.2.8) це засіб, що дозволяє визначити основні класи програми, їх атрибути та методи, а також та їхні взаємозв'язки. Для правильної роботи програми необхідно реалізувати наступні класи: Product, Suplier, Customer. Діаграма класів допомагає розуміти структуру програми та виявляти можливість оптимізації та поліпшення її архітектури. Вона є важливим інструментом при проектуванні програмного забезпечення, спрощуючи розробку та підтримку системи завдяки зрозумілому відображенню взаємозв'язків між класами.

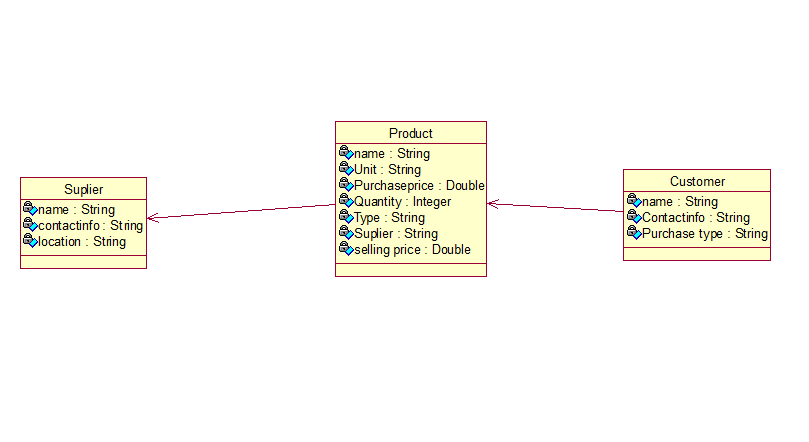


Рисунок 2.8 - Діаграма класів

* діаграма станів (Statechart Diagrams) (рис.2.9)відображає різні стани об'єктів та компонентів системи та переходи між ними відповідно до певних подій. На цій діаграмі представлені можливі стани застосунку, якими можуть керувати актори, що взаємодіють з системою. Ця діаграма допомагає зрозуміти поведінку системи в різних ситуаціях та ідентифікувати потенційні стани, в яких може знаходитися програма.

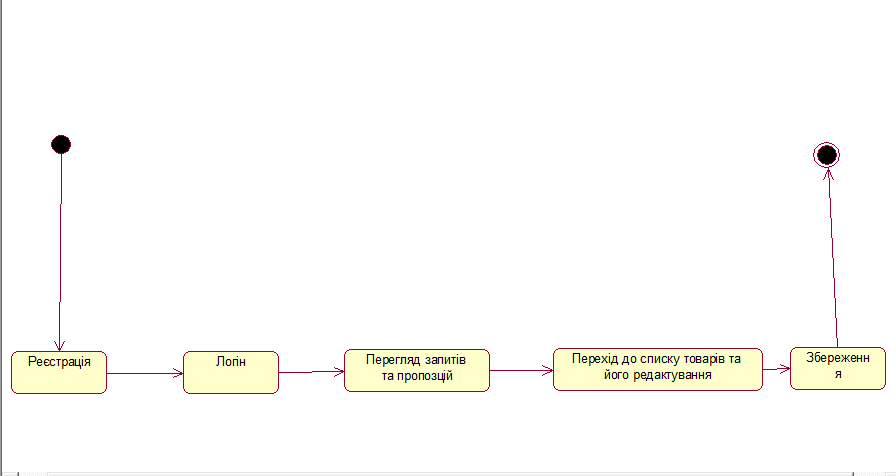


Рисунок 2.9 - Діаграма станів

* Діаграма комунікації (Collaboration diagram) для застосунку "склад для ФОПа" може ілюструвати взаємодію між різними компонентами системи, такими як користувач, система та JSON файл(див. рис2.10)

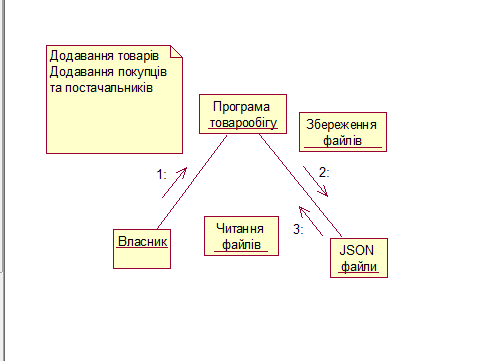


Рисунок 2.10 - Діаграма комунікації

-діаграма послідовності (Sequence Diagrams) відображає порядок дій між

різними об'єктами та компонентами системи під час виконання конкретних сценаріїв використання.(рис.2.11)

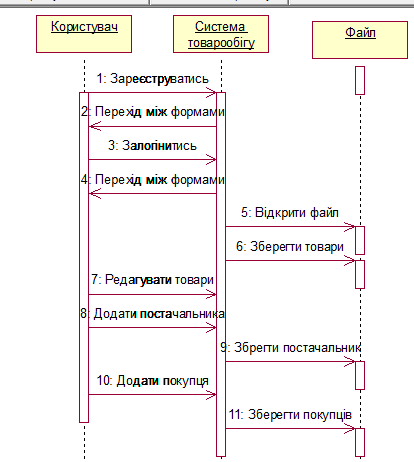


Рисунок 2.11 - Діаграма послідовності

-діаграма активності (Activity Diagrams)(рис.2.12) відображає графічне подання послідовності дій у системі, вказуючи на взаємозв'язки між різними етапами процесу. Активність діаграм допомагає уявити процес взаємодії користувача з системою та ідентифікувати можливі вузькі місця або проблеми у послідовності дій, які потребують уваги під час розробки або оптимізації програмного забезпечення. Вони також можуть бути корисними для визначення точок прийняття рішень, паралельних процесів, умовних переходів та можливих варіантів виконання дій. Крім того, діаграми активності служать важливим інструментом для комунікації між членами команди розробників, аналітиками та іншими зацікавленими сторонами, забезпечуючи зрозуміле та наочне представлення бізнес-процесів та операційних процедур (рис 2.12)

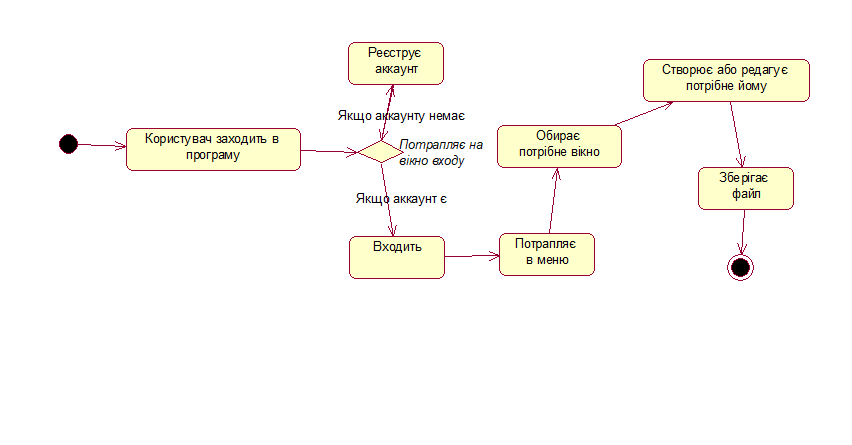


Рисунок 2.12 - Діаграма активності

## **2.3 Об’єктно-орієнтоване проєктування, програмування та тестування**

Першим кроком буде створення пустого проєкту у середовищі Visual Studio Windows Forms App(.NET Framework). Наступний крок створити форми та надати їм відповідні імена: Login, Customer, Dovidnyk, Pokupform, Register, Spysokproductiv, Suplierform, vlasnik, zamov

На Login додамо кнопку button1,textbox1 і textbox2 для логіна та пароля, також додамо label3 для того щоб натискаючи на неї переходимо на форму реєстрації(див. рис.2.13)

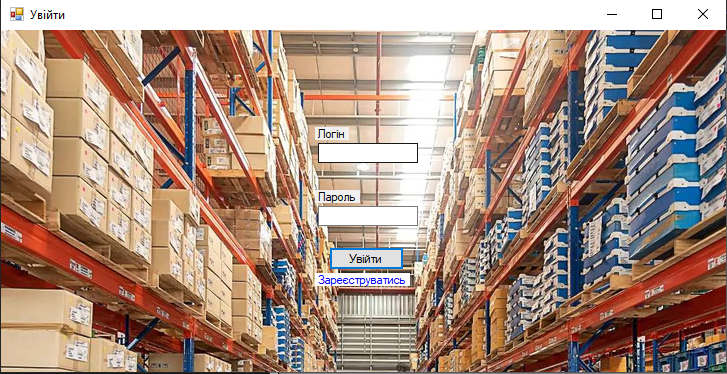


Рисунок 2.13 – Додані кнопки на Form1

Функція відповідає за перехід на наступну форму, якщо правильно введенні логін та пароль:

string login = textBox1.Text;

string password = textBox2.Text;

if (string.IsNullOrWhiteSpace(login))

{

MessageBox.Show("Логін не може бути пустим!");

return;

}

if (string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Пароль не може бути пустим!");

return;

}

User[] users = ReadUsersFromFile("users.json");

User loggedInUser = ValidateUser(users, login, password);

if (loggedInUser != null)

{

switch (loggedInUser.Role)

{

case "Власник":

MessageBox.Show("Ви увійшли в систему як Власник");

Vlasnik vlasnikForm = new Vlasnik(loggedInUser.Login);

vlasnikForm.Show();

this.Hide();

break;

case "Постачальник":

MessageBox.Show("Ви увійшли в систему як Постачальник");

postach\_form postachForm = new postach\_form();

postachForm.Show();

this.Hide();

break;

case "Покупець":

MessageBox.Show("Ви увійшли в систему як Покупець");

pokupform secondForm = new pokupform();

secondForm.Show();

this.Hide();

break;

default:

MessageBox.Show("У вас немає доступу до цієї системи");

break;

}

}

else

{

MessageBox.Show("Неправильний логін або пароль");

}

}

Функція що відповідає за перехід на форму реєстрації:

private void Label3\_Click(object sender, EventArgs e)

{

Register registerform = new Register();

registerform.Show();

this.Visible = false;}

З форми Login переходимо у форму Vlasnik там додаємо 5 кнопок.Також додаємо stripmenu з допомогою користувачу та кнопкою виходу див. малюнки(див. рис.2.14)

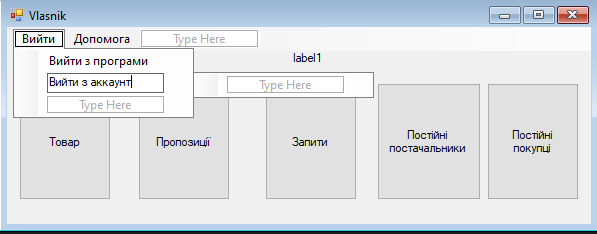


Рисунок 2.14 – Додане меню на формі Vlasnik

Перейдемо в основне вікно програми, яка є самою головною, адже саме в ній зберігається список товарів. Натискаємо кнопку з написом товар та переходимо у форму Spysoktovariv. На форму Spysoktovariv нас зустрічає наступне(див.рис.2.15)

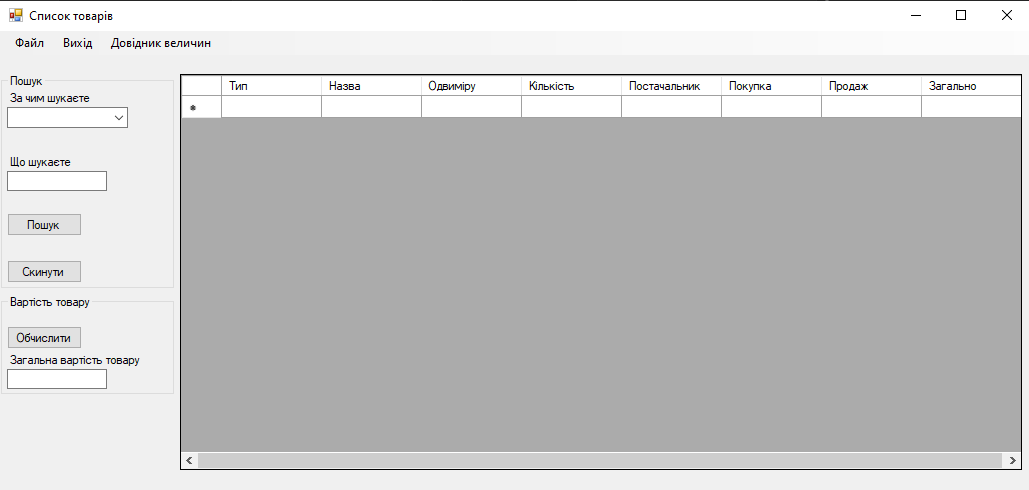


Рисунок 2.15 – форма Spysoktovariv

Функція Зберегти як…ToolStripMenuItem\_Click створює діалогове вікно для збереження файлу, дозволяючи користувачу зберегти дані у JSON файл. Основні кроки:

* Відкривається діалогове вікно для вибору місця та імені файлу.
* Після вибору файлу програма намагається зберегти файл.
* Якщо збереження проходить успішно, з'являється повідомлення про успіх.
* У випадку помилки відображається повідомлення з деталями помилки.

Ця функція зручна для збереження даних, створених або відредагованих у програмі.

Функція ВідкритиToolStripMenuItem\_Click дозволяє користувачу завантажувати дані з файлу у програму. Основні кроки цієї функції:

* Відкривається діалогове вікно для вибору файлу, де користувач може обрати файл.
* Якщо користувач вибирає файл та натискає "OK", програма намагається завантажити дані з вибраного файлу.
* Якщо завантаження проходить успішно, дані з’являються у DatagridView1.
* У разі помилки завантаження відображається повідомлення з деталями помилки.

Ця функція зручна для завантаження та перегляду даних, дозволяючи користувачу легко імпортувати файли з диску.

У окремому файлі створимо класи product

using System;

namespace DoskochKursach

{

internal class Product

{

public string Name { get; set; }

public string Unit { get; set; }

public int Quantity { get; set; }

public Suplier Suplier { get; set; }

public string Type { get; set; }

public decimal PurchasePrice { get; set; }

public decimal SellingPrice { get; set; }

public Product(string name, string unit, int quantity, Suplier suplier, string type, decimal purchasePrice, decimal sellingPrice)

{

Name = name;

Unit = unit;

Quantity = quantity;

Suplier = suplier;

Type = type;

PurchasePrice = purchasePrice;

SellingPrice = sellingPrice;

}

public override string ToString()

{

return $"Товар: {Name}, Одиниця виміру: {Unit}, Кількість: {Quantity}, Постачальник: {Suplier.Name}, Тип: {Type}, Ціна покупки: {PurchasePrice}, Ціна продажу: {SellingPrice}";

}

}

}

#### Поля класу:

* **Name** (назва): властивість типу string, яка зберігає назву товару;
* Unit(вимірювання): властивість типу string, яка зберігає одиницю виміру;
* Quantity(кількість): властивість типу int, яка зберігає кількість товару;
* Suplier(постачальник): властивість агрегації типу string,яка зберігає ім’я постачальника;
* Type(тип): властивість типу string, яка зберігає тип товару;
* PurchasePrice(ціна покупки): властивість типу децимал, яка зберігає ціну покупки товару;
* SellingPrice(ціна продажу): властивість типу децимал, яка зберігає ціну продажу товару;

#### Конструктор:

* public Product(string name, string unit, int quantity, Suplier suplier, string type, decimal purchasePrice, decimal sellingPrice).Конструктор приймає параметри і ініціалізує відповідні властивості класу.

Створюємо також клас Suplier:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DoskochKursach

{

internal class Suplier

{

public string Name { get; set; }

public string Contactinfo { get; set; }

public string Location { get; set; }

public Suplier(string name, string contactinfo, string location)

{

Name = name;

Contactinfo = contactinfo;

Location = location;

}

public override string ToString()

{

return $"Постачальник: {Name}, Контактні дані: {Contactinfo}, Розташування: {Location}";

}

}

}

Додамо у програму функції для введення даних про постачальників та покупців користувачем. Вони забезпечують діалогові вікна для ввводу ширини та висоти кожного елемента і додають ці елементи до відповідних списків після успішного введення:

private List<Suplier> supliers = new List<Suplier>();

private List<Customer> customers = new List<Customer>();

створення списків для збереження даних

у функції button1\_click зберігаємо параметри покупців:

customers.Add(customer);

у функції button1\_click у іншій формі зберігаємо параметри вікон:

supliers.Add(suplier);

Додаємо кнопку button3, яка буде рахувати загальну вартість всіх товарів на складі:

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

{

decimal totalSum = 0;

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (!row.IsNewRow)

{

decimal totalCost = Convert.ToDecimal(row.Cells["Загально"].Value);

totalSum += totalCost;

}

}

textBox2.Text = totalSum.ToString();

}

Далі окремим файлом додаємо наступний клас:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DoskochKursach

{

public class Order

{

public int Number { get; set; }

public string Product { get; set; }

public int Quantity { get; set; }

public decimal Price { get; set; }

public string Supplier { get; set; }

public string ContactInfo { get; set; }

public string Status { get; set; }

}

}

Клас, який представляє замовлення та має властивості номер, товар, кількість, постачальник, статус.

Створюємо схожий клас для покупця, та називаємо покупка

using System;

namespace DoskochKursach

{

public class Purchase

{

public int Number { get; set; }

public string Product { get; set; }

public int Quantity { get; set; }

public decimal Price { get; set; }

// Changed field from Supplier to Customer

public string Customer { get; set; }

public string ContactInfo { get; set; }

public string Status { get; set; }

}

}

Також додаємо клас OrderUtilities :

public static class OrderUtilities

{

public static decimal CalculateTotalCost(int quantity, decimal price)

{

return quantity \* price;

}

}

Це статичний клас, який відповідає за підрахування вартості в стовпці таблиці продуктів.

Додамо методи до цього класу:

public static decimal CalculateTotalCost(int quantity, decimal price)

{

return quantity \* price;

}

Також додаємо функції Loadsfromfile та Saveproductstofile.

Функція Loadsfromfile:

* + Відповідає за читання JSON файлу;
  + Заповнює таблицю зчитаними даними.

Функція Saveproductstofile:

* Cтворює лист об’єктів з таблиці для успішного перенесення в JSON файл.
* Зберігає всі дані у файл формату JSON.

Ці функції важливі для:

* Реалізації збереження та зчитування даних.
* Перевірки на правильність під час збереження.
* Перевірки на неполадки під час завантаження.

На форму Spysoktovariv додамо кнопку button1 для пошуку за певними даними

string selectedColumn = comboBox1.SelectedItem?.ToString();

string searchText = textBox1.Text.ToLower();

if (!string.IsNullOrEmpty(selectedColumn) && !string.IsNullOrEmpty(searchText))

{

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (row.Cells[selectedColumn]?.Value != null)

{

string cellValue = row.Cells[selectedColumn].Value.ToString().ToLower();

row.Visible = cellValue.Contains(searchText);

}

}

}

У цій функції додали пошук по значенню з комбобокса і введених даних у текстбокс

Додаємо кнопку скинути пошук:

textBox1.Clear();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

row.Visible = true;

}

Функція очищує дані, що шукає користувач.

Потрібно організувати перехід між формами за допомогою таких команд:

Dovidnyk dovid = new Dovidnyk();

Login login = new Login();

# Спеціальний розділ

## **3.1 Інструкція з інсталяції розробленого проєкту**

Інсталювати проєкт можна отримавши через електронний або інший носій папку з .exe файлом. Потрібно найти місце збереження та відкрити файл setup(21.05.2024) розміром 553 Кб(рис.3.1)

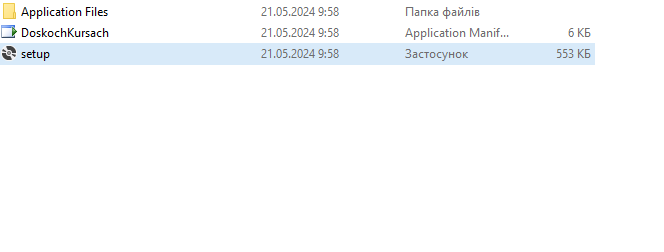


Рисунок 3.1 – Файл встановлення

Після швидкотривалого встановлення у вас буде встановлена програма.Запустити програму ви можете знайшовший в її в меню пуск за назвою “Doskochkursach”

## **3.2 Інструкція з експлуатації розробленого проєкту**

Користувач при запуску програми бачить початкове вікно, вікно авторизації(рис.3.2)

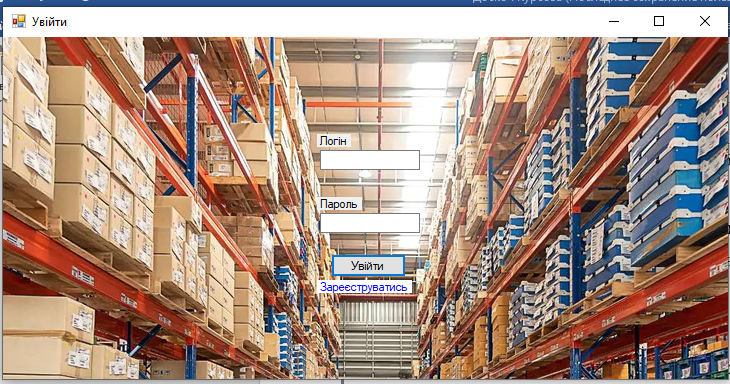


Рисунок 3.2– Початкове вікно програми

Користувач перейшов на форму реєстрації, де він створює собі аккаунт з

роллю, логіном та паролем натискаючи на кнопку «Зареєструватись»(рис. 3.3):

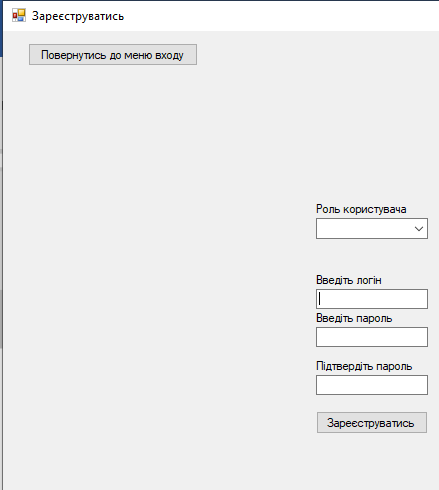


Рисунок 3.3 – форма, де є «Зареєструватись»

Після натискання цієї кнопки. Видає повідомлення що користувач зареєструвався. Тепер він повертається на вікно реєстрації за допомогою клавіші «Повернутись до меню входу». Там він заходить в ролі власника складу, де бачить наступне вікно(рис. 3.4)

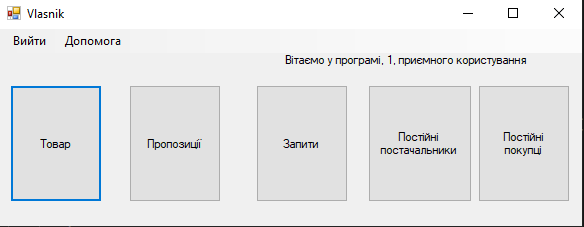


Рисунок 3.4 – форма головного меню власника

Після натиску на будь яку з 5 клавіш користувач перейде на форму відповідну до клавіші, наприклад при натискані кнопки «Товар»(рис. 3.5)

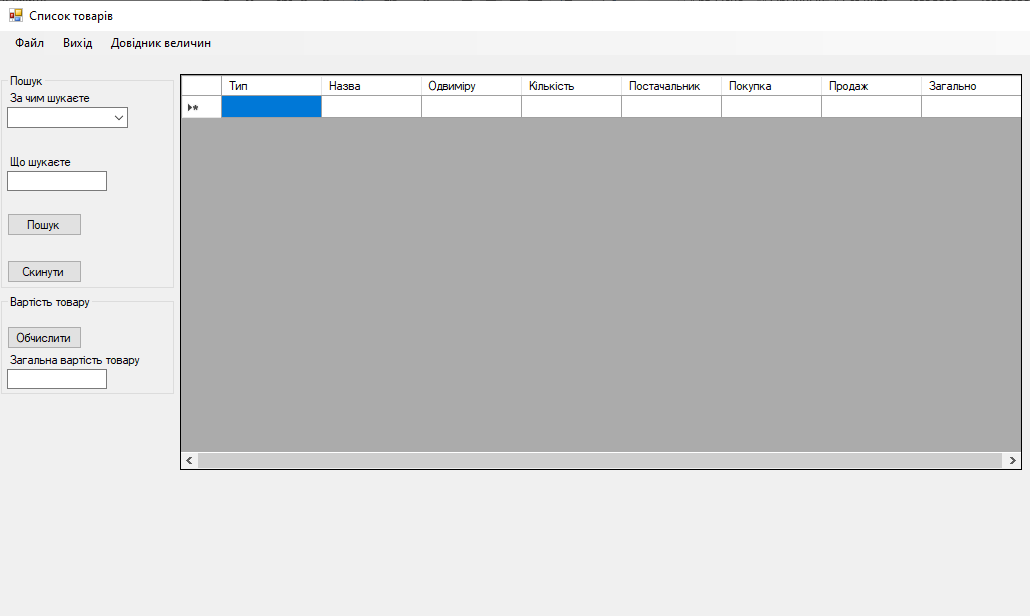


Рисунок 3.5 – форма зі списком товарів

Користувач заповнює таблицю своїми товарами, після чого натискає на кнопку зберегти, що просто збереже файл з його логіном у папці програми, або ж натисне зберегти як і відкриє вікно(рис. 3.6)

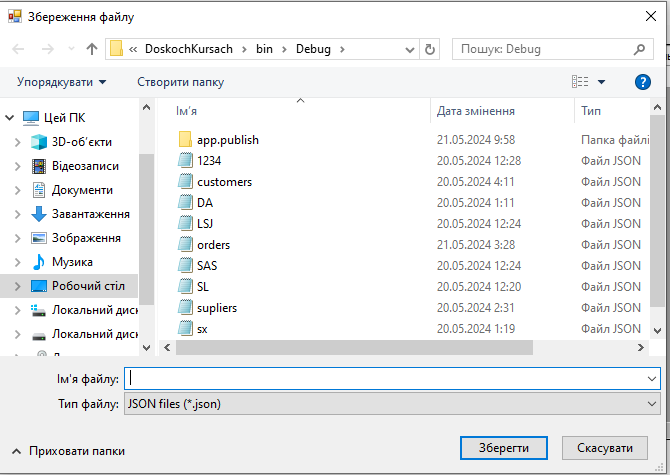
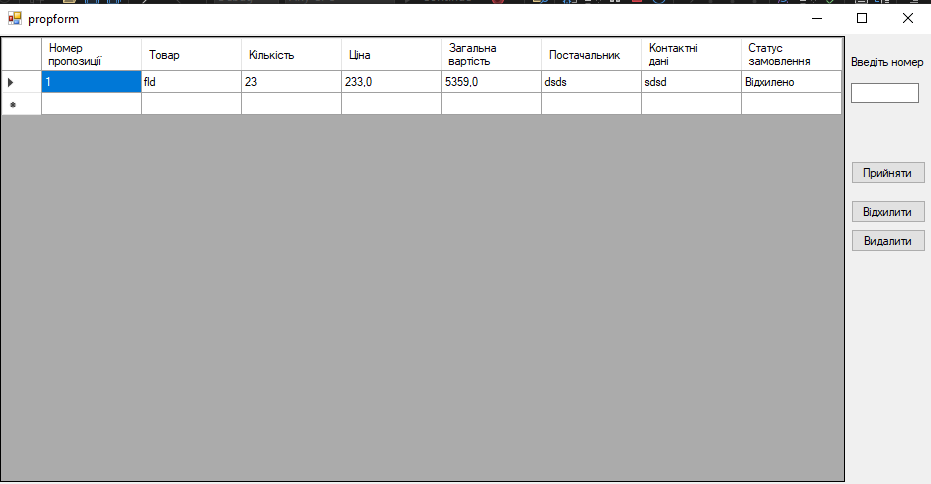


Рисунок 3.6 – Збереження файла

Тепер уявимо що користувач хоче перейти та подивитись запити від постачальників. Він виходить в головне меню. Та натискає кнопку з написом пропозиції, після чого відкривається вікно(рис. 3.7)

Рисунок 3.7 – пропозиції від постачальників

Висновки

В ході виконання курсового проєкту було значно покращено вміння працювати з об'єктно-орієнтованим програмуванням (ООП). Ознайомлення з додатковою літературою дозволило глибше зрозуміти використовувані технології та ефективніше структурувати дані в проєкті.

З метою створення простого та зрозумілого інтерфейсу для користувача, спрямованого на популяризацію застосунку, було покращено вміння працювати в середовищі розробки Visual Studio 2022. Це дозволило зосередитися на функціональності програми, спрощуючи процес розробки та полегшуючи взаємодію з користувачем.

Під час розробки курсового проєкту була проведена аналітична робота в предметній області, створені моделі об'єктів та проведено порівняння з аналогами для виявлення переваг та недоліків використаних інструментів. Цей процес дозволив збагатити розуміння ринкових тенденцій та вимог користувачів, що сприятиме подальшому вдосконаленню продукту.

У результаті розробки курсового проєкту було створено програму для управління товарообігом на складі. Перевагами цього програмного забезпечення порівняно з аналогами є простий та інтуїтивний дизайн, який поєднує в собі основні потреби для ведення товарообігу. Такий підхід дозволяє користувачам зосередитися на важливих аспектах їхньої роботи та підвищує їх продуктивність.

Після аналізу фінальної версії додатку можна зазначити його відповідність сучасним вимогам та зручність в користуванні. Для майбутніх версій планується додавання додаткового функціоналу для розширення можливостей використання програми, що допоможе залучити більше користувачів та збільшити їхню задоволеність від використання продукту.

Список використаних джерел

1. Wikipedia Діаграма класів [електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Діаграма_класів>
2. Wikipedia Діаграма послідовності [електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Діаграма_послідовності>
3. Wikipedia Діаграма комунікації [електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Діаграма_комунікації>
4. Wikipedia Діаграма станів [електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Діаграма_станів_(UML)>
5. Wikipedia Діаграма діяльності [електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Діаграма_діяльності>
6. Мова програмування C# 9 і платформа .NET 5: основні принципи і практики програмування, том 1, 10-е видання
7. ДСТУ 3008-95 "Документація. Звіти в сфері науки і техніки. Структура і правила оформлення" – Державний стандарт України.
8. Вермулен, М. (2018). **Принципи, Практики та Шаблони Впровадження Залежностей**. Manning Publications.
9. Мартін, Р. (2019). Чиста архітектура. Мистецтво розробки програмного забезпечення
10. Річардс, М. (2023). Сучасний підхід до програмної архітектури.Складні компроміси

# Додаток А: Код проекту

Код до форми LOGIN

using Newtonsoft.Json;

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 System.IO;

namespace DoskochKursach

{

public partial class Login : Form

{

public Login()

{

InitializeComponent();

}

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

{

string login = textBox1.Text;

string password = textBox2.Text;

if (string.IsNullOrWhiteSpace(login))

{

MessageBox.Show("Логін не може бути пустим!");

return;

}

if (string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Пароль не може бути пустим!");

return;

}

User[] users = ReadUsersFromFile("users.json");

User loggedInUser = ValidateUser(users, login, password);

if (loggedInUser != null)

{

switch (loggedInUser.Role)

{

case "Власник":

MessageBox.Show("Ви увійшли в систему як Власник");

Vlasnik vlasnikForm = new Vlasnik(loggedInUser.Login);

vlasnikForm.Show();

this.Hide();

break;

case "Постачальник":

MessageBox.Show("Ви увійшли в систему як Постачальник");

postach\_form postachForm = new postach\_form();

postachForm.Show();

this.Hide();

break;

case "Покупець":

MessageBox.Show("Ви увійшли в систему як Покупець");

pokupform secondForm = new pokupform();

secondForm.Show();

this.Hide();

break;

default:

MessageBox.Show("У вас немає доступу до цієї системи");

break;

}

}

else

{

MessageBox.Show("Неправильний логін або пароль");

}

}

private User[] ReadUsersFromFile(string filePath)

{

if (!File.Exists(filePath))

{

return new User[0];

}

string json = File.ReadAllText(filePath);

return JsonConvert.DeserializeObject<User[]>(json);

}

private User ValidateUser(User[] users, string login, string password)

{

foreach (User user in users)

{

if (user.Login == login && user.Password == password)

{

return user;

}

}

return null;

}

public class User

{

public string Login { get; set; }

public string Password { get; set; }

public string Role { get; set; }

}

private void Label3\_Click(object sender, EventArgs e)

{

Register registerform = new Register();

registerform.Show();

this.Visible = false;

}

}

}

Код до форми Register

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Newtonsoft.Json;

namespace DoskochKursach

{

public partial class Register : Form

{

public Register()

{

InitializeComponent();

}

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

{

string login = textBox1.Text;

string password = textBox2.Text;

string confirmPassword = textBox3.Text;

string role = comboBox1.SelectedItem?.ToString();

if (string.IsNullOrWhiteSpace(login))

{

MessageBox.Show("Логін не може бути пустим!");

return;

}

if (string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Пароль не може бути пустим!");

return;

}

if (string.IsNullOrWhiteSpace(confirmPassword))

{

MessageBox.Show("Підтвердження пароля не може бути пустим!");

return;

}

if (string.IsNullOrWhiteSpace(role))

{

MessageBox.Show("Роль не може бути пустою!");

return;

}

if (password == confirmPassword)

{

User user = new User(login, password, role);

SaveUser(user);

MessageBox.Show("Користувач успішно зареєстрований!");

}

else

{

MessageBox.Show("Паролі не співпадають!");

}

}

private void SaveUser(User user)

{

List<User> userList = ReadUsersFromFile("users.json").ToList();

userList.Add(user);

string json = JsonConvert.SerializeObject(userList);

using (StreamWriter writer = new StreamWriter("users.json"))

{

writer.Write(json);

}

}

private IEnumerable<User> ReadUsersFromFile(string filePath)

{

if (!File.Exists(filePath))

{

return Enumerable.Empty<User>();

}

string json = File.ReadAllText(filePath);

return JsonConvert.DeserializeObject<IEnumerable<User>>(json);

}

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

{

Login loginform = new Login();

loginform.Show();

this.Visible = false;

}

}

public class User

{

public string Login { get; set; }

public string Password { get; set; }

public string Role { get; set; }

public User(string login, string password, string role)

{

Login = login;

Password = password;

Role = role;

}

}

}

Код до форми vlasnik

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;

namespace DoskochKursach

{

public partial class Vlasnik : Form

{

private string \_userLogin; // Поле для збереження userLogin

public Vlasnik(string userLogin)

{

InitializeComponent();

\_userLogin = userLogin; // Зберігаємо значення userLogin у полі

label1.Text = $"Вітаємо у програмі, {\_userLogin}, приємного користування";

}

private void button5\_Click(object sender, EventArgs e)

{

// Логіка для button5

}

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

{

Spysoktovariv spysoktovariv = new Spysoktovariv(\_userLogin); // Використовуємо фактичне значення userLogin

spysoktovariv.Show();

this.Hide();

}

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

{

Suplierform suplierform = new Suplierform();

suplierform.Show();

this.Visible = false;

}

private void button5\_Click\_1(object sender, EventArgs e)

{

Customerform customerform = new Customerform();

customerform.Show();

this.Visible = false;

}

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

{

propform propform = new propform();

propform.Show();

this.Hide();

}

}

}

Код до форми Spysoktovariv

using System;

using System.Collections.Generic;

using System.IO;

using System.Windows.Forms;

using Newtonsoft.Json;

namespace DoskochKursach

{

public partial class Spysoktovariv : Form

{

private string userLogin;

public Spysoktovariv(string userLogin)

{

InitializeComponent();

this.userLogin = userLogin;

LoadDataFromJson();

PopulateComboBox();

}

private void LoadDataFromJson()

{

string fileName = $"{userLogin}.json";

if (File.Exists(fileName))

{

string json = File.ReadAllText(fileName);

List<object[]> rows = JsonConvert.DeserializeObject<List<object[]>>(json) ?? new List<object[]>();

PopulateDataGridView(rows);

}

}

private void PopulateDataGridView(List<object[]> rows)

{

dataGridView1.Rows.Clear();

foreach (object[] row in rows)

{

dataGridView1.Rows.Add(row);

}

}

private void SaveDataToJson()

{

List<object[]> rows = new List<object[]>();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (!row.IsNewRow)

{

object[] rowData = new object[dataGridView1.ColumnCount];

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

rowData[i] = row.Cells[i].Value;

}

rows.Add(rowData);

}

}

string fileName = $"{userLogin}.json";

string json = JsonConvert.SerializeObject(rows, Formatting.Indented);

File.WriteAllText(fileName, json);

}

private void PopulateComboBox()

{

comboBox1.Items.Clear();

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

comboBox1.Items.Add(column.HeaderText);

}

}

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

{

string selectedColumn = comboBox1.SelectedItem?.ToString();

string searchText = textBox1.Text.ToLower();

if (!string.IsNullOrEmpty(selectedColumn) && !string.IsNullOrEmpty(searchText))

{

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (row.Cells[selectedColumn]?.Value != null)

{

string cellValue = row.Cells[selectedColumn].Value.ToString().ToLower();

row.Visible = cellValue.Contains(searchText);

}

}

}

}

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

{

textBox1.Clear();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

row.Visible = true;

}

}

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

{

SaveDataToJson();

MessageBox.Show("Data saved successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

SaveProductsAs();

}

private void SaveProductsAs()

{

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "JSON files (\*.json)|\*.json|All files (\*.\*)|\*.\*",

FilterIndex = 1,

RestoreDirectory = true

};

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

string fileName = saveFileDialog.FileName;

SaveProductsToFile(fileName);

MessageBox.Show("Data saved successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void SaveProductsToFile(string fileName)

{

List<object[]> rows = new List<object[]>();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (!row.IsNewRow)

{

object[] rowData = new object[dataGridView1.ColumnCount];

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

rowData[i] = row.Cells[i].Value;

}

rows.Add(rowData);

}

}

string json = JsonConvert.SerializeObject(rows, Formatting.Indented);

File.WriteAllText(fileName, json);

}

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

{

OpenProducts();

}

private void OpenProducts()

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

Filter = "JSON files (\*.json)|\*.json|All files (\*.\*)|\*.\*",

FilterIndex = 1,

RestoreDirectory = true

};

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

string fileName = openFileDialog.FileName;

LoadProductsFromFile(fileName);

MessageBox.Show("Data loaded successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void LoadProductsFromFile(string fileName)

{

string json = File.ReadAllText(fileName);

List<object[]> rows = JsonConvert.DeserializeObject<List<object[]>>(json) ?? new List<object[]>();

PopulateDataGridView(rows);

}

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

{

Login login = new Login();

login.Show();

this.Visible = false;

}

private void вихідToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

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

{

decimal totalSum = 0;

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (!row.IsNewRow)

{

decimal totalCost = Convert.ToDecimal(row.Cells["Загально"].Value);

totalSum += totalCost;

}

}

textBox2.Text = totalSum.ToString();

}

private void dataGridView1\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (!row.IsNewRow)

{

int quantity = Convert.ToInt32(row.Cells["Кількість"].Value);

decimal sellingPrice = Convert.ToDecimal(row.Cells["Продаж"].Value);

decimal totalCost = OrderUtilities.CalculateTotalCost(quantity, sellingPrice);

row.Cells["Загально"].Value = totalCost;

}

}

}

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

{

Dovidnyk dovid = new Dovidnyk();

dovid.Show();

this.Visible = false;

}

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

{

dataGridView1.Rows.Clear();

}

}

public static class OrderUtilities

{

public static decimal CalculateTotalCost(int quantity, decimal price)

{

return quantity \* price;

}

}

}

Код до форми Dovidnyk:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace DoskochKursach

{

public partial class Dovidnyk : Form

{

private MySqlConnection connection;

private string server;

private string database;

private string uid;

private string password;

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

{

LoadDataFromDatabase();

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

comboBox1.Items.Add(column.HeaderText);

}

}

public Dovidnyk()

{

InitializeComponent();

InitializeConnection();

CheckConnection();

}

private void InitializeConnection()

{

server = "localhost";

database = "kursovadata";

uid = "root";

password = "OpppoA31";

string connectionString = $"SERVER={server};DATABASE={database};UID={uid};PASSWORD={password};";

connection = new MySqlConnection(connectionString);

}

private void CheckConnection()

{

try

{

connection.Open();

MessageBox.Show("Connection successful!");

}

catch (MySqlException ex)

{

MessageBox.Show($"Connection failed: {ex.Message}");

}

finally

{

if (connection.State == ConnectionState.Open)

connection.Close();

}

}

private void ExecuteQuery(string query)

{

try

{

connection.Open();

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

}

catch (MySqlException ex)

{

MessageBox.Show($"Error: {ex.Message}");

}

finally

{

if (connection.State == ConnectionState.Open)

connection.Close();

}

}

private void LoadDataFromDatabase()

{

dataGridView1.Rows.Clear();

string query = "SELECT \* FROM dovidnyk";

try

{

connection.Open();

MySqlCommand cmd = new MySqlCommand(query, connection);

MySqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

string column1Value = reader["Value1"].ToString();

string column2Value = reader["Value2"].ToString();

string column3Value = reader["Value3"].ToString();

dataGridView1.Rows.Add(column1Value, column2Value, column3Value);

}

reader.Close();

}

catch (MySqlException ex)

{

MessageBox.Show($"Error: {ex.Message}");

}

finally

{

if (connection.State == ConnectionState.Open)

connection.Close();

}

}

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

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "Binary File|\*.bin";

saveFileDialog1.Title = "Save as Binary File";

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

try

{

using (BinaryWriter writer = new BinaryWriter(File.Open(saveFileDialog1.FileName, FileMode.Create)))

{

foreach (DataGridViewRow row in dataGridView1.Rows)

{

foreach (DataGridViewCell cell in row.Cells)

{

if (cell.Value != null)

{

writer.Write(cell.Value.ToString());

}

else

{

writer.Write(string.Empty);

}

}

}

}

MessageBox.Show("Data saved successfully!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("Error saving data: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter = "Binary File|\*.bin";

openFileDialog1.Title = "Open Binary File";

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

try

{

using (BinaryReader reader = new BinaryReader(File.Open(openFileDialog1.FileName, FileMode.Open)))

{

dataGridView1.Rows.Clear();

int rowsCount = reader.ReadInt32();

int columnsCount = reader.ReadInt32();

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

{

dataGridView1.Rows.Add();

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

{

string cellValue = reader.ReadString();

dataGridView1.Rows[i].Cells[j].Value = cellValue;

}

}

}

MessageBox.Show("Data loaded successfully!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("Error loading data: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

dataGridView1.Rows.Clear();

}

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

{

if (int.TryParse(textBox8.Text, out int rowIndexToDelete))

{

if (rowIndexToDelete > 0 && rowIndexToDelete <= dataGridView1.Rows.Count)

{

int primaryKeyValue = rowIndexToDelete;

string queryDelete = $"DELETE FROM dovidnyk WHERE IDds = {primaryKeyValue}";

ExecuteQuery(queryDelete);

string queryUpdate = $"UPDATE dovidnyk SET IDds = IDds - 1 WHERE IDds > {primaryKeyValue}";

ExecuteQuery(queryUpdate);

LoadDataFromDatabase();

MessageBox.Show("Видалено успішно.", "Виконано", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Введене значення поза межами таблиці.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Invalid row index. Please enter a valid number.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

if (comboBox1.SelectedIndex != -1)

{

string selectedColumn = comboBox1.SelectedItem.ToString();

string searchText = textBox11.Text;

if (!string.IsNullOrEmpty(searchText))

{

foreach (DataGridViewRow row in dataGridView1.Rows)

{

row.DefaultCellStyle.BackColor = dataGridView1.DefaultCellStyle.BackColor;

foreach (DataGridViewCell cell in row.Cells)

{

if (cell.Value == null)

cell.Value = "";

}

}

bool found = false;

foreach (DataGridViewRow row in dataGridView1.Rows.Cast<DataGridViewRow>().Where(r => !r.IsNewRow))

{

string cellValue = row.Cells[selectedColumn].Value?.ToString();

if (!string.IsNullOrEmpty(cellValue) && cellValue.Contains(searchText))

{

row.DefaultCellStyle.BackColor = Color.Yellow;

found = true;

}

else

{

row.Visible = false;

}

}

if (!found)

{

MessageBox.Show("Немає відповідників.", "Пошук", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else

{

MessageBox.Show("Введіть текст для пошуку.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Виберіть стовпець для пошуку.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button5\_Click(object sender, EventArgs e)

{

foreach (DataGridViewRow row in dataGridView1.Rows)

{

row.Visible = true;

row.DefaultCellStyle.BackColor = dataGridView1.DefaultCellStyle.BackColor;

}

}

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

{

string value1 = textBox3.Text;

string value2 = textBox2.Text;

string value3 = textBox1.Text;

string query = $"INSERT INTO dovidnyk (Value1, Value2, Value3) VALUES ('{value1}', '{value2}', '{value3}')";

ExecuteQuery(query);

LoadDataFromDatabase();

textBox3.Clear();

textBox2.Clear();

textBox1.Clear();

}

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

{

if (int.TryParse(textBox6.Text, out int rowIndex))

{

if (rowIndex >= 0 && rowIndex < dataGridView1.Rows.Count)

{

string primaryKeyValue = dataGridView1.Rows[rowIndex].Cells[0].Value?.ToString();

if (!string.IsNullOrEmpty(primaryKeyValue))

{

string value5 = textBox5.Text;

string value7 = textBox7.Text;

string value4 = textBox4.Text;

// Оновлений запит для оновлення даних, використовуючи ім'я поля "IDds"

string query = $"UPDATE dovidnyk SET Value1 = '{value5}', Value2 = '{value7}', Value3 = '{value4}' WHERE IDds = '{primaryKeyValue}'";

ExecuteQuery(query);

LoadDataFromDatabase();

MessageBox.Show("Зміни збережено успішно.", "Виконано", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Некоректне значення ID рядка.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Номер рядка виходить за межі таблиці.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Некоректно введений номер рядка.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

Login login = new Login();

login.Show();

this.Visible = false;

}

private void вихідЗПрограмиToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

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

{

MessageBox.Show("Вітаю вас у вікні довідник. У цьому вікні ви зможете переглянути та редагувати список величин які ви можете використовувати у функціонуванні програми. Зауважте!якщо ви створили величину або редагували вже існуючи вам потрібно зберегти те що ви зробили, адже жодні інщі способи не збережуть ваші дані ");

}

}

}

Код до форми Suplierform:

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using System.IO;

namespace DoskochKursach

{

public partial class Suplierform : Form

{

private List<Suplier> supliers = new List<Suplier>();

public Suplierform()

{

InitializeComponent();

}

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

{

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

comboBox1.Items.Add(column.HeaderText);

}

}

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

{

dataGridView1.Rows.Clear();

}

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

{

string name = textBox3.Text;

string contactInfo = textBox2.Text;

string location = textBox1.Text;

Suplier suplier = new Suplier(name, contactInfo, location);

supliers.Add(suplier);

dataGridView1.Rows.Add(suplier.Name, suplier.Contactinfo, suplier.Location);

textBox3.Clear();

textBox2.Clear();

textBox1.Clear();

}

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

{

if (int.TryParse(textBox8.Text, out int rowIndexToDelete))

{

if (rowIndexToDelete > 0 && rowIndexToDelete <= dataGridView1.Rows.Count)

{

if (dataGridView1.Rows[rowIndexToDelete - 1].Cells.Cast<DataGridViewCell>().Any(cell => !string.IsNullOrEmpty(cell.Value?.ToString())))

{

supliers.RemoveAt(rowIndexToDelete - 1);

dataGridView1.Rows.RemoveAt(rowIndexToDelete - 1);

MessageBox.Show("Видалено успішно.", "Виконано", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Даний рядок і так є пустим, отже не підлягає видаленню.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Введене значення поза межами таблиці.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Invalid row index. Please enter a valid number.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

if (comboBox1.SelectedIndex != -1)

{

string selectedColumn = comboBox1.SelectedItem.ToString();

string searchText = textBox11.Text;

if (!string.IsNullOrEmpty(searchText))

{

foreach (DataGridViewRow row in dataGridView1.Rows)

{

row.DefaultCellStyle.BackColor = dataGridView1.DefaultCellStyle.BackColor;

foreach (DataGridViewCell cell in row.Cells)

{

if (cell.Value == null)

cell.Value = "";

}

}

bool found = false;

foreach (DataGridViewRow row in dataGridView1.Rows.Cast<DataGridViewRow>().Where(r => !r.IsNewRow))

{

string cellValue = row.Cells[selectedColumn].Value?.ToString();

if (!string.IsNullOrEmpty(cellValue) && cellValue.Contains(searchText))

{

row.DefaultCellStyle.BackColor = Color.Yellow;

found = true;

}

else

{

row.Visible = false;

}

}

if (!found)

{

MessageBox.Show("Немає відповідників.", "Пошук", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else

{

MessageBox.Show("Введіть текст для пошуку.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Виберіть стовпець для пошуку.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button5\_Click(object sender, EventArgs e)

{

foreach (DataGridViewRow row in dataGridView1.Rows)

{

row.Visible = true;

row.DefaultCellStyle.BackColor = dataGridView1.DefaultCellStyle.BackColor;

}

}

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

{

}

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

{

if (int.TryParse(textBox6.Text, out int rowIndex))

{

if (rowIndex >= 0 && rowIndex < dataGridView1.Rows.Count)

{

Suplier suplierToUpdate = supliers[rowIndex];

string newName = textBox5.Text;

string newContactInfo = textBox7.Text;

string newLocation = textBox4.Text;

if (!string.IsNullOrEmpty(newName))

{

suplierToUpdate.Name = newName;

}

if (!string.IsNullOrEmpty(newContactInfo))

{

suplierToUpdate.Contactinfo = newContactInfo;

}

if (!string.IsNullOrEmpty(newLocation))

{

suplierToUpdate.Location = newLocation;

}

dataGridView1.Rows[rowIndex].Cells[0].Value = suplierToUpdate.Name;

dataGridView1.Rows[rowIndex].Cells[1].Value = suplierToUpdate.Contactinfo;

dataGridView1.Rows[rowIndex].Cells[2].Value = suplierToUpdate.Location;

}

else

{

MessageBox.Show("Номер рядка виходить за межі таблиці.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Некоректно введений номер рядка.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

Dovidnyk dovid = new Dovidnyk();

dovid.Show();

this.Visible = false;

}

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

{

}

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

{

MessageBox.Show("У цьому вікні ви можете керувати вашими постійними постачальниками, а саме додавати, видаляти, редагувати, це допоможе тримати дані важливих постачальників в одному місці");

}

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

{

SaveDataToJson("supliers.json");

}

private void зберегтиЯкToolStripMenuItem1\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "JSON files (\*.json)|\*.json|All files (\*.\*)|\*.\*";

saveFileDialog.RestoreDirectory = true;

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

SaveDataToJson(saveFileDialog.FileName);

}

}

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

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "JSON files (\*.json)|\*.json|All files (\*.\*)|\*.\*";

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

LoadDataFromJson(openFileDialog.FileName);

}

}

private void SaveDataToJson(string fileName)

{

try

{

string jsonData = JsonConvert.SerializeObject(supliers, Formatting.Indented);

File.WriteAllText(fileName, jsonData);

MessageBox.Show("Дані успішно збережено у JSON файл.", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при збереженні даних: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void LoadDataFromJson(string fileName)

{

try

{

string jsonData = File.ReadAllText(fileName);

supliers = JsonConvert.DeserializeObject<List<Suplier>>(jsonData);

// Очищення DataGridView перед завантаженням нових даних

dataGridView1.Rows.Clear();

// Заповнення DataGridView з завантаженими даними

foreach (var suplier in supliers)

{

dataGridView1.Rows.Add(suplier.Name, suplier.Contactinfo, suplier.Location);

}

MessageBox.Show("Дані успішно завантажено з JSON файлу.", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при завантаженні даних: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

Код до класу product

using System;

namespace DoskochKursach

{

internal class Product

{

public string Name { get; set; }

public string Unit { get; set; }

public int Quantity { get; set; }

public Suplier Suplier { get; set; }

public string Type { get; set; }

public decimal PurchasePrice { get; set; }

public decimal SellingPrice { get; set; }

public Product(string name, string unit, int quantity, Suplier suplier, string type, decimal purchasePrice, decimal sellingPrice)

{

Name = name;

Unit = unit;

Quantity = quantity;

Suplier = suplier;

Type = type;

PurchasePrice = purchasePrice;

SellingPrice = sellingPrice;

}

public override string ToString()

{

return $"Товар: {Name}, Одиниця виміру: {Unit}, Кількість: {Quantity}, Постачальник: {Suplier.Name}, Тип: {Type}, Ціна покупки: {PurchasePrice}, Ціна продажу: {SellingPrice}";

}

}

}

Код до класу suplier:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DoskochKursach

{

internal class Suplier

{

public string Name { get; set; }

public string Contactinfo { get; set; }

public string Location { get; set; }

public Suplier(string name, string contactinfo, string location)

{

Name = name;

Contactinfo = contactinfo;

Location = location;

}

public override string ToString()

{

return $"Постачальник: {Name}, Контактні дані: {Contactinfo}, Розташування: {Location}";

}

}

}

**КП.ПО.02.ПІ.211.08.ПЗ**