ЗМІСТ

С.

[Вступ 4](#_Toc136195327)

[1 Технічні характеристики 5](#_Toc136195328)

[1.1 Постановка задачі 5](#_Toc136195329)

[1.1.1 Характеристика об’єкта автоматизації 5](#_Toc136195330)

[1.1.2 Вимоги до програми 6](#_Toc136195331)

[1.2 Опис математичного методу рішення задачі 7](#_Toc136195332)

[1.3 Визначення структури вхідних даних 9](#_Toc136195333)

[1.4 Визначення структури вихідних даних 15](#_Toc136195334)

[2 Комп’ютерна система 17](#_Toc136195335)

[2.1 Технічні характеристики комп’ютера та зовнішніх пристроїв 17](#_Toc136195336)

[2.2 Вибір програмних засобів та операційної системи 17](#_Toc136195337)

[3 Об’єктно-орієнтоване проєктування 21](#_Toc136195338)

[3.1 Інтерфейс програми 21](#_Toc136195339)

[3.2 Алгоритм рішення задачі 23](#_Toc136195340)

[4 Програмування та тестування 25](#_Toc136195341)

[4.1 Розробка програми 25](#_Toc136195342)

[4.2 Етапи налагодження 28](#_Toc136195343)

[4.3 Типи помилок 28](#_Toc136195344)

[5 Очікуванні економічні показники 31](#_Toc136195345)

[5.1 Розрахунок вартості програмного продукту 31](#_Toc136195346)

[5.2 Розрaхунок eкономiчної eфeктивностi вiд впровaджeння прогрaмного зaбeзпeчeння 40](#_Toc136195347)

[5.3. Нaуковa оргaнiзaцiя прaцi нa робочому мiсцi прогрaмiстa 46](#_Toc136195348)

[6 Охорона праці 48](#_Toc136195349)

[Висновки 59](#_Toc136195350)

[Перелік джерел посилання 60](#_Toc136195351)

[Додаток А 61](#_Toc136195352)

[Додаток Б 62](#_Toc136195353)

[Додаток В 67](#_Toc136195354)

[Перелік інших частин:](#_Toc136195355)

1. [Текст програми ДП.ХПФК.00452 – 41 12 01 68](#_Toc136195356)

2. [Посібник програміста ДП.ХПФК.00452 – 41 33 01 106](#_Toc136195357)

3. [Посібник оператора ДП.ХПФК.00452 – 41 34 01 108](#_Toc136195358)

4. [Специфікація ДП.ХПФК.00452 – 41 113](#_Toc136195359)

# ВСТУП

Автоматизація є дуже важливою частиною в бізнесі, особливо в торговому. Також автоматизація процесів зменшить кількість помилок, спричинених людським фактором. Часто підприємці автоматизовують свій бізнес через збільшення даних та неможливість керування ними. У чому саме полягає автоматизація, це запровадження додатку в свій бізнес, до цього додатку прив’язується база даних, з якої будуть братись дані. Такий додаток буде застосовуватись в сфері торгівлі господарських товарів.

Область торгівлі продуктових товарів включає великий асортимент продукції. Саме тому автоматизація даної галузі є виправданою і доцільною. Тому що клієнт може задовольнити більшість своїх потреб в одній оптовій базі.

Метою дипломного проєкту є створення програмного продукту, який буде використовуватись менеджером з продаж та адміністратором оптовій базі.

Завдання дипломного проєкту – є створення програмного продукту, робота якого буде спрямована на мінімізацію помилок і на пришвидшення роботи оптової бази та підвищення прибутковості з мінімальною кількістю працівників, що призведе до зменшення витрат на заробітні плати та підвищення прибутковості оптової бази.

# ТЕХНІЧНІ ХАРАКТЕРИСТИКИ

## Постановка задачі

### Характеристика об’єкта автоматизації

Оптова база - це комплексна торгівельна організація, яка спеціалізується на продажі оптових партій продуктів харчування, товарів для дому та особистої гігієни. Оптові бази мають розгалужену мережу підрозділів, які знаходяться в різних районах міста та регіонів.

Основні функції оптової бази включають продаж продуктів харчування, управління запасами та планування закупівель, керування продажами та клієнтською базою даних, ведення бухгалтерського обліку.

Для автоматизації діяльності оптової бази необхідне програмне забезпечення, яке дозволить ефективно управляти складським обліком, контролювати рух товарів, планувати закупівлі, керувати продажами та клієнтською базою даних, а також вести бухгалтерський облік. Важливим параметром є також прибутковість продуктів та обсяги продажів.

Для забезпечення ефективної автоматизації оптової бази, програмне забезпечення повинно бути інтегровано з програмами обліку та управління запасами, які вже використовуються в компанії. Це дозволить створити єдину інформаційну систему, яка забезпечить повну інтеграцію всіх підрозділів та ефективний контроль за управлінням компанією.

### 1.1.2 Вимоги до програми

Призначення програмного забезпечення та програмного продукту: розробка системи автоматизації управління та аналізу продажами в оптовій базі.

Перелік задач, які будуть вирішуватися в результаті використання розробленого програмного продукту:

* Облік товарів на складі;
* керування закупівлями товарів;
* ведення обліку продажів;
* взаємодія з клієнтами та програма лояльності;
* генерація звітів та аналіз даних.

Структура програмного продукту має наступний вигляд:

* Модуль перегляду та додавання продуктів;
* модуль замовлення товарів;
* модуль аналізу даних;
* модуль налаштування системи.

Опис функціонування програмного продукту та його частин:

* Модуль перегляду та додавання продуктів надає можливість переглянути список всіх продуктів зареєстрованих в системі, додавати нові продукти та редагувати уже існуючі;
* модуль замовлення товарів надає можливість закупляти товари та редагувати уже існуючі документи закупівель;
* модуль аналізу даних аналізує дані та генерує звіти для управління та оптимізації бізнес-процесів;
* модуль налаштування системи надає можливість створювати, редагувати та активовувати/деактивовувати облікові записи користувачів системи.

Користувачами програмного продукту можуть бути менеджер з продаж, що використовує програму на робочому комп’ютері під час взаємодії з клієнтами. Також програма може використовуватись адміністратором оптової бази, який може виконувати всі вище згадані задачі.

## Опис математичного методу рішення задачі

При розробці поставленої задачі було використано метод об’єктно-орієнтованого програмування.

Об’єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину взаємодіючих об'єктів. У ній використано декілька технологій від попередніх парадигм, включаючи успадкування, модульність, поліморфізм та інкапсуляцію. Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування Симула в 1960-тих роках, одночасно з посиленням [дискусій](http://ua-referat.com/%D0%94%D0%B8%D1%81%D0%BA%D1%83%D1%81%D1%8B) про кризу програмного забезпечення. Незважаючи на те, що ця [парадигма](http://ua-referat.com/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0) з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих.

Об'єктно-орієнтовані програми можна розглядати як множину взаємодіючих об'єктів, на відміну від традиційних поглядів, коли програма розглядається як набір підпрограм, або як перелік інструкцій комп'ютеру. [Відповідно](http://ua-referat.com/%D0%92%D1%96%D0%B4%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D1%8C) до парадигми ООП, кожен об'єкт здатен отримувати повідомлення, обробляти дані, та надсилати повідомлення іншим об'єктам. Кожний об'єкт можна розглядати як незалежний [автомат](http://ua-referat.com/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82) з окремим призначенням або відповідальністю.

Об'єктно-орієнтований підхід полягає в наступному наборі основних принципів:

* все є об'єктами;
* всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об'єктами, при якій один об'єкт потребує, щоб інший об'єкт виконав деяку дію. Об'єкти взаємодіють, надсилаючи і отримуючи повідомлення. Повідомлення - це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії;
* кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів;
* кожен об'єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об'єктів;
* у класі задається поведінка (функціональність) об'єкта. Таким чином усі об'єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії;
* класи організовані у єдину деревовидну структуру з загальним корінням, яка називається ієрархією [успадкування](http://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29). Пам'ять та поведінка, зв'язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Об’єктно-орієнтоване програмування засноване на трьох принципах:

* інкапсуляції;
* наслідуванні;
* поліморфізмі.

Спостережуване в об'єктах об'єднання даних і операцій в одне ціле було позначене [терміном](http://ua-referat.com/%D0%A2%D0%B5%D1%80%D0%BC%D1%96%D0%BD%D0%B8) інкапсуляція. Застосування інкапсуляції зробило об'єкти схожими на маленькі програмні модулі і забезпечило приховування їх внутрішньої структури. Для об'єктів з'явилося [поняття](http://ua-referat.com/%D0%9F%D0%BE%D0%BD%D1%8F%D1%82%D1%82%D1%8F) інтерфейсу, що значно підвищило їх надійність і цілісність.

Наслідування означає, що при створенні нового класу, немає необхідності в переписуванні заново всіх полів, методів і властивостей. Ви оголошуєте, що новий клас є нащадком (або дочірнім класом) наявного класу, так званим предком (або батьківським класом), і додаєте до нього нові поля, методи та властивості. Іншими словами, додається те, що потрібне для переходу від загального до приватного. [Процес](http://ua-referat.com/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81) створення нових класів на основі інших класів називається наслідуванням. Нові класи мають як успадковані ознаки, так і нові.

Поліморфізм означає, що в похідних класах можна змінювати роботу методів, що вже існують в базовому класі. При цьому весь програмний код, що керує об'єктами батьківського класу, придатний для управління об'єктами дочірнього класу без будь-якої модифікації.

Також у програмі використано ієрархію класів для створення динамічного графічного інтерфейсу та взаємодії даних.

## Визначення структури вхідних даних

Вхідними даними називають такі змінні (параметри, дані), які надаються виконавцеві до початку його роботи над алгоритмом.

Застосування алгоритму до неприпустимих(або неперевірених) вхідних даних може призводити до того, що алгоритм ніколи не зупиниться, або потрапить в тупиковий стан (зависання), з якого не зможе продовжитися виконання процесу.

Таблиця 1.1 – Вхідна інформація

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва | Форма подання | Термін і частота використання |
| 1 | Товар | Масив оперативної інформації | Не обмежено |
| 2 | Працівники | Масив оперативної інформації | Не обмежено |
| 3 | Клієнт | Масив оперативної інформації | Не обмежено |
| 4 | Товарна накладна | Масив оперативної інформації | Не обмежено |
| 6 | Виробник | Масив нормативно-довідкової інформації | Не обмежено |
| 7 | Постачальник | Масив нормативно-довідкової інформації | Не обмежено |
| 8 | Кошик | Масив оперативної інформації | Не обмежено |
| 9 | Користувач | Масив нормативно-довідкової інформації | Не обмежено |
| 10 | Документ списання | Масив оперативної інформації | Не обмежено |

Вхідні дані зазвичай поділяють на два класи: довідкову та оперативну інформацію. Довідкові дані змінюються рідше, ніж оперативні, які можуть змінюватися навіть в режимі реального часу.

Вхідні дані представлені у вигляді таблиць, що зберігаються в базі данних.

Таблиця 1.2 – Структура вхідних даних таблиці «Товар»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Pr\_Id | int | Довге ціле | Унікальний ідентифікатор товару |
| 2. | Pr\_BarCode | varchar(20) |  | Штрих-код товару |
| 3. | Pr\_Description | varchar(max) |  | Опис товару |
| 4. | Pr\_Brand | int | Довге ціле | Ідентифікатор бренду товару |
| 5. | Pr\_Title | varchar(200) |  | Назва товару |
| 6. | Pr\_Category | int | Довге ціле | Ідентифікатор категорії товару |
| 7. | Pr\_PriceOpt | decimal(18,2) |  | Оптова ціна товару |
| 8. | Pr\_Price | decimal(18,2) |  | Роздрібна ціна товару |
| 9. | Pr\_Qty | varchar(20) |  | Одиниці виміру |
| 10. | Pr\_Reorder | int | Довге ціле | Критичний залишок |
| 12. | Pr\_Image | varbinary(MAX) |  | Зображення товару |

Таблиця 1.3 – Структура вхідних даних таблиці «Списання»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Adj\_Id | int | Довге ціле | Унікальний ідентифікатор |
| 2. | Adj\_Product | int | Довге ціле | Ідентифікатор товару |
| 3. | Adj\_Qty | int | Довге ціле | Кількість списаного товару |
| 4. | Adj\_Remarks | varchar | 0..50 | Додаткові зауваження до налаштування товару |
| 5. | Adj\_Date | datetime |  | Дата |
| 6. | Adj\_User | varchar | 0.50 | Користувач |

Таблиця 1.4 – Структура вхідних даних таблиці «Кошик»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Crt\_Id | int | Довге ціле | Унікальний ідентифікатор |
| 2. | Crt\_Transno | varchar | 0..50 | Номер транзакції |
| 3. | Crt\_Product | int | Довге ціле | Ідентифікатор товару |
| 4. | Crt\_Price | decimal |  | Ціна товару |
| 5. | Crt\_Qty | int | Довге ціле | Кількість товару |
| 6. | Crt\_Disc\_Percent | decimal |  | Відсоток знижки |
| 7. | Crt\_Discount | decimal |  | Сума знижки |
| 8. | Crt\_Total | decimal |  | Загальна сума |

Продовження таблиці 1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 9. | Crt\_Date | datetime |  | Дата та час |
| 10. | Crt\_Status | int | Довге ціле | Статус |
| 11. | Crt\_Cashier | int | Довге ціле | Ідентифікатор менеджера |

Таблиця 1.5 – Структура вхідних даних таблиці «Категорія»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Catg\_Id | int | Довге ціле | Унікальний ідентифікатор |
| 2. | Catg\_Title | varchar | 0..50 | Назва категорії |
| 3. | Catg\_Description | text |  | Опис категорії |

Таблиця 1.6 – Структура вхідних даних таблиці «Клієнт»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Cl\_Id | int | Довге ціле | Унікальний ідентифікатор |
| 2. | Cl\_Name | varchar | 0..50 | ПІБ клієнта |
| 3. | Cl\_CardNumber | varchar | 10 | Номер картки клієнта |

Таблиця 1.7 – Структура вхідних даних таблиці «Працівник»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Emp\_Id | int | Довге ціле | Унікальний ідентифікатор співробітника |
| 2. | Emp\_Name | varchar | 0..20 | Ім'я співробітника |
| 3. | Emp\_Surname | varchar | 0..20 | Прізвище співробітника |
| 4. | Emp\_FatherName | varchar | 0..20 | По-батькові співробітника |
| 5. | Emp\_Position | varchar | 0..50 | Посада співробітника |
| 6. | Emp\_Salary | money |  | Зарплата співробітника |
| 7. | Emp\_JoiningDate | datetime |  | Дата прийняття співробітника на роботу |
| 8. | Emp\_Gender | char |  | Стать співробітника |

Таблиця 1.8 – Структура вхідних даних таблиці «Закупівлі»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Sti\_Id | int | Довге ціле | Унікальний ідентифікатор документу закупівлі |
| 2. | Sti\_Date | datetime |  | Дата надходження товару |
| 3. | Sti\_StockInBy | varchar | 0..50 | Ім'я особи, яка привезла товар |
| 4. | Sti\_Status | int | 1 | Статус надходження |
| 5. | Sti\_SupplierId | int | Довге ціле | Унікальний ідентифікатор постачальника |
| 6. | Sti\_IsConfirmed | bit | 0-1 | Підтвердження надходження товару |

Таблиця 1.9 – Структура вхідних даних таблиці «Постачальник»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Sup\_Id | int | Довге ціле | Унікальний ідентифікатор постачальника |
| 2. | Sup\_Name | varchar | 0..50 | Назва постачальника |
| 3. | Sup\_Address | text |  | Адреса постачальника |
| 4. | Sup\_ContactPerson | varchar | 0..50 | Контактна особа постачальника |
| 5. | Sup\_Phone | varchar | 0..50 | Телефон постачальника |
| 6. | Sup\_Email | varchar | 0..50 | Електронна адреса постачальника |
| 7. | Sup\_Fax | varchar | 0..50 | Факс постачальника |

Таблиця 1.10 – Структура вхідних даних таблиці «Користувач»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Ідентифікатор | Тип | Діапазон значень | Пояснення |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Usr\_Id | int | Довге ціле | Унікальний ідентифікатор користувача |
| 2. | Usr\_Employee | int | Довге ціле | Ідентифікатор співробітника, який є користувачем |
| 3. | Usr\_Username | varchar | 0..50 | Ім'я користувача |
| 4. | Usr\_Password | varchar | 0..50 | Пароль користувача |

Продовження таблиці 1.10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 5. | Usr\_Role | int | Довге ціле | Роль користувача (наприклад, адміністратор) |
| 6. | Usr\_IsActivate | bit | 0-1 | Чи активний обліковий запис користувача |

## Визначення структури вихідних даних

Вихідною інформацією задачі для користувача будуть звіти та форми, які будуть відображатися на під час роботи з програмою як таблиці та форми. Також вихідна інформація може бути роздрукована . Вихідні дані – це дані, які виконавець видає як результат здійсненої роботи. Вихідними даними у програмному додатку є інформація про товари, чеки, клієнтів та працівників. В таблиці 1.11 показані вихідні дані, які використовуються в дипломному проєкті.

Таблиця 1.11 - Вихідна інформація

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Форма | Звіт |
| 1 | Товари | + |  |
| 2 | Чек |  | + |
| 3 | Залишки на складі | + | + |
| 4 | Звіт Топ продаж | + | + |
| 5 | Транзакції покупок | + | + |
| 6 | Працівники | + | + |
| 7 | Звіт Прибуток | + | + |

Таблиця 1.12 - Структура звіту « Прибуток»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  з/п | Ідентифікатор | Тип | Діапазон  значень | Пояснення |
| 1. | Місяць | Текстовий | 0…50 | - |
| 2. | Сума | Грошовий | - | - |

Таблиця 1.13 - Структура звіту « Топ продаж»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  з/п | Ідентифікатор | Тип | Діапазон  значень | Пояснення |
| 1. | Код товару | Текстовий | 0…50 | - |
| 2. | Назва товару | Грошовий | - | - |
| 3. | Кількість | Числовий | Довге ціле | - |

Таблиця 1.14 - Структура чеку

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  з/п | Ідентифікатор | Тип | Діапазон  значень | Пояснення |
| 1. | Назва товару | Текстовий | 0…50 | - |
| 2. | Ціна | Грошовий | - | - |
| 3. | Кількість | Числовий | Довге ціле | - |
| 4. | Знижка % | Числовий | 0-100 | - |
| 5. | Знижка | Грошовий | - | - |
| 6. | Сума | Грошовий | - | - |
| 7. | Касир | Текстовий | 0-50 | - |
| 8. | Дата | Дата/час | - | - |
| 9. | Назва оптової бази | Текстовий | 0-50 | - |
| 10. | Адреса | Текстовий | 0-50 | - |

# КОМП’ЮТЕРНА СИСТЕМА

## Технічні характеристики комп’ютера та зовнішніх пристроїв

Для роботи розробленої прикладки використовувався персональний комп’ютер (ПК) з наступними характеристиками:

* Процесор AMD Ryzen 7 4800 H 2.9 GHz;
* оперативна пам’ять 16 ГБ DDR4 3200 MHz;
* відеокарта NVIDIA GTX 1650 Ti;
* монітор 1920x1080 60Hz;
* операційна система Windows 11 Pro;

Програма може функціонувати без втрати всіх функціональних можливостей на комп’ютері з такими технічними характеристиками:

- процесор з частотою 1ггц;

- RAM 128 MB ;

- відеокарта Intel HD Graphics;

- операційна система Windows, Mac, Linux, Chrome OS.

## Вибір програмних засобів та операційної системи

Для розробки веб-сайту було обрано такі засоби реалізації:

* мова програмування С# на платформі .NET;
* СУБД MSSQL Server;
* Microsoft RDLC дизайнер звітів.

C# є об'єктно-орієнтованою мовою програмування, що дозволяє розробникам створювати програми, які можуть бути легко розширені та підтримувані з часом. Мова підтримує багато функцій, які дозволяють розробникам писати якісний та ефективний код. C# використовується для створення різноманітних програм, включаючи веб-додатки, мобільні додатки, ігри та програми для роботи з базами даних. Мова є основою для платформи .NET, що дозволяє розробникам створювати програми, які можуть працювати на різних операційних системах, таких як Windows, Linux та macOS.

Переваги мови програмування C#:

1. Об'єктно-орієнтований підхід: C# є повноцінною об'єктно-орієнтованою мовою програмування, що дозволяє розробникам створювати більш прості, ефективні та складні програми.
2. Безпека: C# має вбудований механізм безпеки, який дозволяє розробникам створювати більш безпечні програми, знижуючи ризик вразливостей та атак зловмисників.
3. Платформно-незалежний код: С# розроблена для роботи на платформі .NET, що дозволяє розробникам створювати програми, які можуть працювати на різних операційних системах.
4. Бібліотеки та фреймворки: C# має велику кількість сторонніх бібліотек та фреймворків, що дозволяє розробникам створювати програми швидко та ефективно.

Недоліки мови програмування C#:

1. Залежність від платформи: C# була розроблена для платформи .NET, тому програми, написані на C#, можуть працювати тільки на операційній системі Windows або на платформі .NET.
2. Швидкість виконання: Порівняно з мовами програмування, такими як C та C++, виконання програм на C# може бути менш ефективним та повільним.
3. Низький рівень контролю над пам'яттю: Хоча C# має вбудований збірник сміття, що допомагає управляти пам'яттю, розробникам може бути складніше контролювати пам'ять.

MS SQL є однією з найпоширеніших реляційних баз даних, розроблених корпорацією Microsoft. Ця система управління базами даних дозволяє ефективно зберігати та організовувати великі об'єми даних. MS SQL забезпечує високу надійність, безпеку та продуктивність завдяки своїм функціям, таким як механізми резервного копіювання, відновлення даних та захист від вторгнень. Ця система має широкий спектр можливостей, включаючи розширені можливості звітування, інструменти аналізу даних та можливість створювати складні запити для отримання потрібної інформації.

Переваги MS SQL:

1. Надійність та стабільність: MS SQL має високий рівень надійності та стабільності завдяки своїм механізмам резервного копіювання та відновлення даних.
2. Висока продуктивність: MS SQL забезпечує високу продуктивність завдяки своїм оптимізаційним можливостям та підтримці паралельної обробки даних.
3. Розширені можливості звітування: MS SQL має вбудовані інструменти для створення різноманітних звітів та аналізу даних.
4. Інтеграція з продуктами Microsoft: MS SQL інтегрується з іншими продуктами Microsoft, що дозволяє ефективно працювати з даними та ділитися ними між різними системами.
5. Масштабованість: MS SQL може легко масштабуватися залежно від потреб користувачів та організацій.

Недоліки MS SQL:

1. Високі витрати: MS SQL є комерційним продуктом, що означає високі витрати на ліцензування та підтримку.
2. Складність: MS SQL може бути складним у використанні для новачків, оскільки має велику кількість функцій та можливостей.
3. Обмежена підтримка інших платформ: MS SQL більш спрямований на платформу Windows та менш підтримує інші операційні системи.
4. Невелика спільнота користувачів: порівняно з деякими іншими реляційними базами даних, MS SQL має меншу спільноту користувачів, що може ускладнити пошук допомоги та розв'язання проблем.

Microsoft RDLC (Report Definition Language Client-Side) Designer є інструментом для створення звітів у середовищі Visual Studio. Він дозволяє створювати різноманітні звіти, використовуючи різні джерела даних, такі як бази даних, веб-сервіси, Excel-файли тощо. RDLC Designer забезпечує зручний інтерфейс для налаштування форматування звітів, додавання тексту, зображень, графіків, таблиць, графіків та інших елементів. Після створення звіту, його можна зберегти в форматі RDLC або скомпілювати в DLL-файл, щоб використовувати його в програмному коді. RDLC Designer є частиною Microsoft SQL Server Reporting Services (SSRS) та може бути використаний для створення звітів у багатьох типах проектів, включаючи ASP.NET, WinForms, WPF, Silverlight тощо.

# ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ

## Інтерфейс програми

Цінність будь-якої програми визначається не тільки її здатністю безпомилково і якісно вирішувати поставлену задачу, а й тим, наскільки легко і зручно цією програмою можна користуватися. Навіть найпростіша, і в той же час, функціональна програма виявиться незатребуваною, якщо користувач (тобто кінцевий споживач програми) при роботі з нею швидко втомлюється, дратується, відчуває себе некомфортно. Під призначеним для користувача інтерфейсом (англ. User interface) будемо розуміти своєрідний комунікаційний канал (сполучна ланка), за допомогою якого користувач взаємодіє з комп'ютером.

З точки зору користувача саме інтерфейс є кінцевим продуктом. Інтерфейс являє собою сукупність використовуваних в програмі засобів для введення даних, способів відображення інформації на екрані дисплея, елементів для управління різними етапами рішення задачі. Основне призначення (головна мета) призначеного для користувача інтерфейсу - забезпечити короткий і зручний шлях до отримання результату.

Інтерфейс користувача комп’ютерного додатку включає:

* Ясність;
* Виразність;
* Відповідність;
* Послідовність;
* Продуктивність;

Невдалі інтерфейси сильно уповільнюють роботу користувачів, сприяють швидкої стомлюваності, великому числу помилок і, як наслідок, відбивають у користувачів подальше бажання працювати з програмою. У зв'язку з цим розробнику програми слід враховувати відомі рекомендації провідних фахівців в області створення призначеного для користувача інтерфейсу.

Саме тому було розроблено максимально комфортний і зручний для користування інтерфейс (див. рис. 4.1 та рис. 4.2):

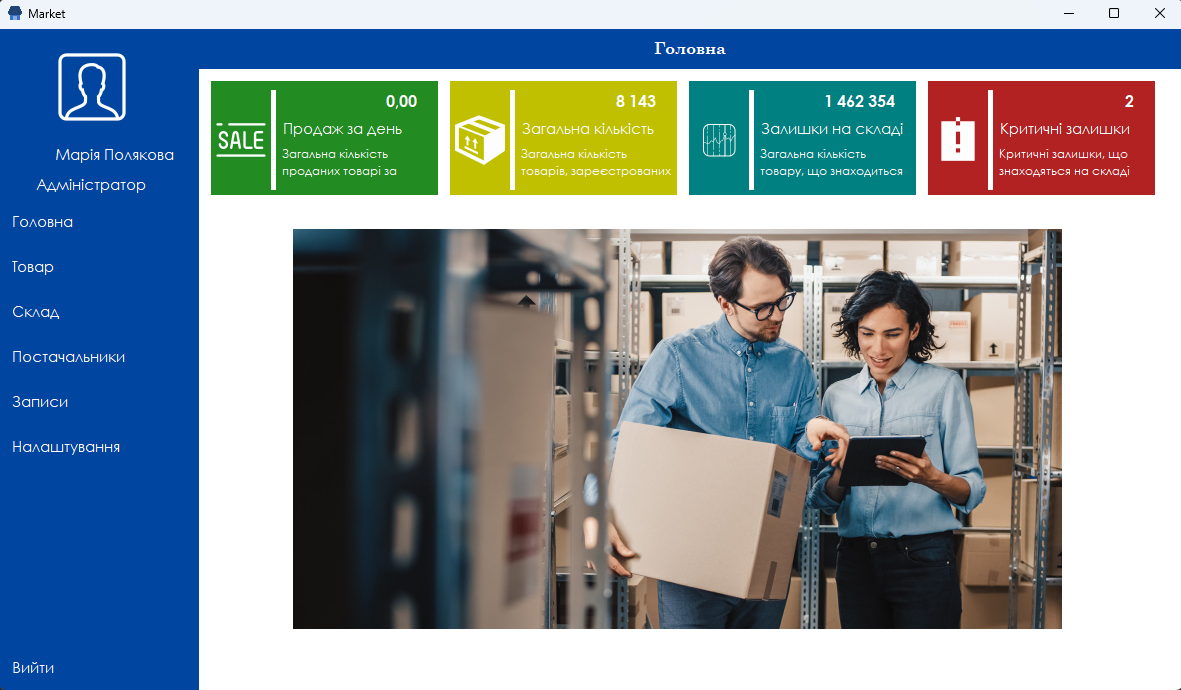


Рисунок 3.1 – Головна форма адміністратора

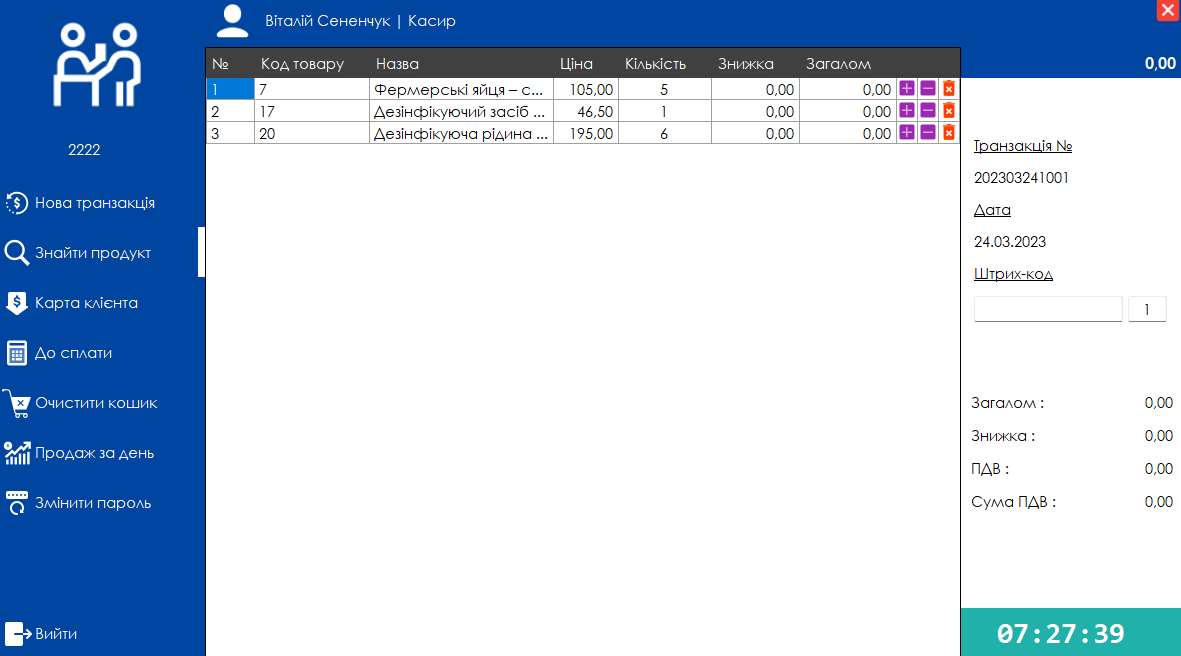


Рисунок 3.2 – Головна форма менеджеру з продаж

## 3.2 Алгоритм рішення задачі

Програмний додаток складається з багатьох екранних форм, кожна за яких виконує певну функцію та має унікальну логіку роботи.

При запуску програми спочатку відкривається форма де користувачу потрібно ввести логін та пароль. Після чого на основі ролі користувача відкриється або головна форма для адміністратора або головна форма для касира. Кожна з яких має свій унікальний функціонал та інтерфейс.

Нижче представленний перелік усіх функцій, що представленні в програмі.

private void btnLogin\_Click(object sender, EventArgs e) – перевірка користувача та відкриття форми відповідно до його ролі;

private void btnProductList\_Click(object sender, EventArgs e) – відкриття форми зі списком продуктів;

private void dgvProduct\_CellContentClick(object sender, DataGridViewCellEventArgs e) – відкриття форми із інформацією про товар;

private void btnAdd\_Click(object sender, EventArgs e) – відкриття форми реєстрації продукту;

private void btnCategory\_Click(object sender, EventArgs e) – відкриття форми зі списком категорій;

private void dgvCategory\_CellContentClick(object sender, DataGridViewCellEventArgs e) – видалення категорії;

private void btnAdd\_Click(object sender, EventArgs e) – відкриття форми для реєстрації нової категорії;

private void btnStockEntry\_Click(object sender, EventArgs e) – відкриття форми зі списком закупок;

private void dgvStockIn\_CellDoubleClick(object sender, DataGridViewCellEventArgs) – відкриття форми з інформацією про закупівлю;

private void btnAdd\_Click(object sender, EventArgs e) – відкриття форми для створення нової закупки;

private void btnStockAdjustment\_Click(object sender, EventArgs e) – відкриття форми для списання продукції;

private void btnSupplier\_Click(object sender, EventArgs e) – відкриття форми зі списком постачальників;

private void btnPosRecord\_Click(object sender, EventArgs e) – відкриття форми для перегляду звітів;

private void btnUser\_Click(object sender, EventArgs e) – відкриття форми налаштування користувачів;

private void btnStore\_Click(object sender, EventArgs e) – відкриття форми налаштування оптової бази;

private void btnNTran\_Click(object sender, EventArgs e) – оновлення форми длял створення нової транзакції;

private void btnSearch\_Click(object sender, EventArgs e) – відкриття форми зі списком продуктів для їх додавання в кошик;

private void btnDiscount\_Click(object sender, EventArgs e) – відкриття форми для додавання картки клієнта;

private void btnSettle\_Click(object sender, EventArgs e) – відкриття форми для підтвердження суми та видачі решти;  
private void btnClear\_Click(object sender, EventArgs e) – оновлення форми з метою видалення всіх продуктів з кошика;

private void btnPass\_Click(object sender, EventArgs e) – відкриття форми для зміни паролю для поточного користувача;

private void btnLogout\_Click(object sender, EventArgs e) – вихід з програми;

# Програмування та тестування

## Розробка програми

Розроблена програма призначена лише для операційної системи Windows, тому програма несумісна з іншими операційними системами (наприклад: Linux, Debian, MacOS тощо).

Після запуску програми, користувач повинен буде ввести логін та пароль, після чого відкриється форма призначена для адміністратора або для менеджера з продажу в залежності від ролі користувача.

В середовищі MS-SQL Server було створено схему даних, яка зображена на рисунку 4.1.

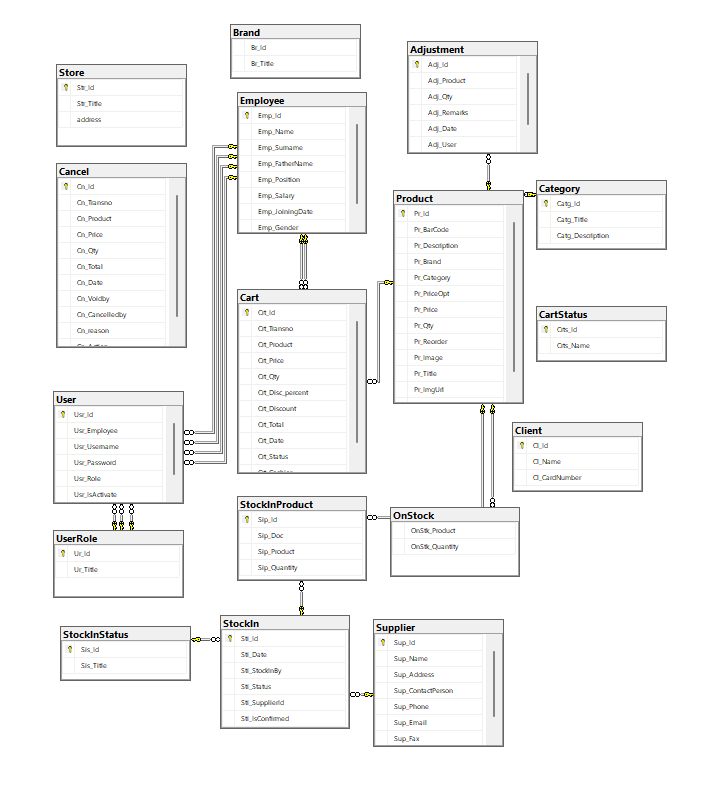


Рисунок 4.1 – Схема даних

Для переходу на інші форми, користувач повинен обрати потрібний пункт меню.

На головній формі адміністратора в меню доступні наступні пункти:

* Головна;
* Товар:
* Список товарів;
* Категорії;
* Виробник;
* Склад:
* Список замовлень;
* Списання
* Постачальники;
* Записи;
* Налаштування:
* Користувач;
* Оптова база;

Обравши пункт “Товар”, в меню з’являється 3 підпункти: Список товарів, Категорії, Виробник. Натиснувши на Список товарів, користувачеві відображається форма зі списком товарів та інформацією про нього при натискані певної кнопки. Натиснувши на Категорії, користувачеві відображається форма зі списком категорій. Натиснувши на Виробник, користувачеві відображається форма зі списком виробників товарів.

Обравши пункт “Склад”, в меню з’являється 2 підпункти: Список замовлень та Списання. Натиснувши на Список замовлень, користувачеві відображається форма зі списком замовлень. На цій формі також можна створити нове замовлення або редагувати існуючі. Натиснувши на Списання, користувачеві відображається форма для списання товарів.

Обравши пункт Постачальники, користувачеві відображається форма зі списком постачальників, на якій також можна зареєструвати нового постачальника.

Обравши пункт Записи, користувачеві відображається форма з усіма звітами та інструментами аналізу.

Обравши пункт “Налаштування”, в меню з’являється 2 підпункти: Користувач та Оптова база. Натиснувши на пункт Користувач, користувачеві відображається форма для налаштування всіх користувачів зареєестрованих в системі. Натиснувши на пункт Оптова база, користувачеві відображається форма для налаштування інформації про оптову базу.

На головній формі адміністратора в меню доступні наступні пункти:

* Нова транзакція;
* Знайти продукт;
* Карта клієнта;
* До сплати;
* Очисти кошик;
* Змінити пароль;

Натиснувши на пункт Нова транзакція на формі очищається список товарів, що були додані до кошика та створюється нова транзакція.

Обравши пункт Знайти продукт, користувачеві відображається форма зі списком продуктів, на якій він повинен обрати товар та підтвердити кількість, яка буде додана до кошика.

Обравши пункт Карта клієнта , користувачеві відображається форма для введення коду карти клієнта та реєстрації нової в випадку її відсутності. Карта клієнта надасть 5-ти відсоткову знижку на кожен продукт.

Обравши пункт До сплати, на екрані відобразиться форма, в якій потрібно ввести суму, яку надав клієнт, потім на формі підрахується решта. Після натиснення кнопки Підтвердити транзакція закриється та видрукується чек.

Обравши пункт Очистити кошик, на формі очиститься таблиця зі списком товарів в кошику, при цьому транзакція залишиться тією ж самою.

Обравши пункт Змінити пароль, касиру відобразиться форма на якій він може змінити пароль для свого профілю.

## Етапи налагодження

Налагодження - це процес виявлення причин виникнення помилки та її подальшого виправлення (на відміну від тестування, що є процесом виявлення самого факту існування помилки).

Процес Налагодження включає в себе:

* Вироблення системних вимог;
* Вироблення вимог до програмного забезпечення;
* Аналіз та проектування;
* Реалізацію - процес написання коду;
* Продуктизацію – процес встановлення відношення програмного забезпечення вимогам і доведення до них;
* Експлуатацію – процес використання програми.

Під час створення програмного продукту в середовищі Visual Studio використовувались інструменти для відлагодження програмного коду. Один з них - це відлагоджувальник, який дозволяє розробникам встановлювати точки зупинки в програмному коді та відстежувати значення змінних та виконання програмного коду по крокам. Відлагоджувальник також дозволяє розробникам виконувати розгалуження в програмному коді та виконувати відлагодження в певних умовах. Інші інструменти для відлагодження в Visual Studio включають в себе вікна «Вихідні теки» та «Вихідний вивід», які дозволяють розробникам переглядати вихідний код та відслідковувати виконання програмного коду. Вікна «Локальні» та «Глобальні» змінні дозволяють розробникам переглядати значення змінних під час виконання програмного коду.

## Типи помилок

Під час розробки програмного продукту можуть виникати різні типи помилок, які можуть бути класифіковані в кілька категорій. Нижче перераховано деякі з найпоширеніших типів помилок:

* Синтаксичні помилки: ці помилки виникають під час компіляції програмного коду та пов'язані з невірним використанням синтаксичних елементів мови програмування, таких як знаки пунктуації, ключові слова, типи даних тощо.
* Логічні помилки: ці помилки пов'язані з невірним виконанням програмного коду під час його виконання. Наприклад, програма може вивести неправильні результати через неправильні умови вибору, помилки у логіці обчислень або неправильний алгоритм.
* Помилки виконання: ці помилки виникають під час виконання програми та пов'язані з недостатнім обсягом ресурсів, таких як пам'ять або процесорний час, що призводить до аварійного завершення програми.
* Помилки взаємодії: ці помилки пов'язані з невірною взаємодією програмного продукту з іншими програмами, операційним середовищем або залежними компонентами. Наприклад, програма може взаємодіяти з базою даних, але через помилки підключення вона не може здійснювати запити.
* Помилки конфігурації: ці помилки пов'язані з неправильним налаштуванням програмного продукту або залежних компонентів, що можуть призводити до неправильної роботи програми або до аварійного завершення.
* Помилки безпеки: ці помилки пов'язані з недостатньою безпекою програмного продукту, що може призвести до несанкціонованого доступу до системи, крадіжки даних або вірусного атаки.

Кожен помилок має свої особливості та специфічні методи їх виявлення та виправлення. Для виявлення синтаксичних та логічних помилок використовують методи відлагодження та тестування. Для виявлення помилок виконання та взаємодії можуть використовуватись моніторинг, аналіз логів та діагностика проблем. Помилки конфігурації можуть бути виявлені за допомогою перевірки налаштувань системи та залежних компонентів. Для виявлення помилок безпеки можуть використовуватись методи тестування на проникнення та аудит безпеки програмного продукту.

Важливо виявляти та виправляти помилки якомога раніше в процесі розробки, оскільки чим пізніше помилка буде виявлена, тим важче її буде виправити. Тому важливо забезпечувати високу якість коду та правильну його організацію, а також використовувати ефективні методи виявлення та виправлення помилок.

Також під час розробки програмного продукту активно використовувався механізм try-catch. Це механізм обробки помилок, який дозволяє контролювати помилки в програмі та забезпечує їх відловлювання та обробку, щоб уникнути краху програми. Конструкція try включає блок коду, який потенційно може породити виключення або помилку, і блок catch, який оброблює ці виключення та надає інструкції щодо їх обробки. Наприклад, у випадку, коли ми виконуємо ділення на нуль, програма породить помилку DivisionByZeroException. Ми можемо використати конструкцію try-catch для перехоплення цієї помилки та виведення користувачеві повідомлення про помилку замість краху програми.

В ході розробки програмного забезпечення виникали деякі помилки різного типу. Приклад синтаксичної помилки в курсовому проекті (див. рис. 4.1)

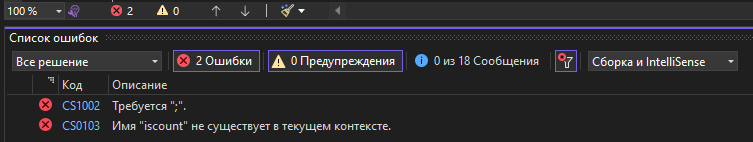


Рисунок 4.1 – Синтаксична помилка

# ОЧІКУВАННІ ЕКОНОМІЧНІ ПОКАЗНИКИ

## 5.1 Розрахунок вартості програмного продукту

Прогрaмний продукт– сукупнiсть прогрaм систeми обробки iнформaцiї i прогрaмних докумeнтiв, нeобхiдних для eксплуaтaцiї цих прогрaм.

Комп'ютeрнa прогрaмa - нaбiр iнструкцiй у виглядi слiв, цифр, кодiв, схeм, символiв чи у будь-якому iншому виглядi, вирaжeних у формi, придaтнiй для зчитувaння комп'ютeром, якi приводять його у дiю для досягнення певної мeти aбо рeзультaту (цe поняття охоплює як опeрaцiйну систeму, тaк i приклaдну прогрaму, вирaжeнi у вихiдному aбо об'єктному кодaх).

Прогрaмнa докумeнтaцiя — сукупнiсть докумeнтiв, що мiстять вiдомостi, нeобхiднi для розробки, виготовлeння, супроводу тa eксплуaтaцiї прогрaм. Прогрaмнa докумeнтaцiя є одним з видiв тeхнiчної докумeнтaцiї.

Для визнaчeння вaртостi прогрaмного зaбeзпeчeння нeобхiдно, пeрш зa всe, встaновити трудомiсткiсть його розробки в тaблицi 5.1.

Тaблиця 5.1 - Трудомiсткiсть прогрaмного зaбeзпeчeння

|  |  |
| --- | --- |
| Пeрeлiк робiт | Трудомiсткiсть, год |
| I Тeхнiчнe зaвдaння  1 Aнaлiз прeдмeтної облaстi  2 Постaновкa зaдaчi  Рaзом | 11  30  41 |
| II Тeхнiчний проeкт  1 Опис мaтeмaтичного мeтоду рiшeння зaдaчi  2 Структурa вхiдних i вихiдних дaних  Рaзом | 70  22  92 |

Продовжeння тaблицi 5.1

|  |  |
| --- | --- |
| Пeрeлiк робiт | Трудомiсткiсть, год |
| III Робочий проeкт  1 Вибiр прогрaмного зaсобу  2 Визнaчeння вимог до тeхнiчних зaсобiв  3 Опис aлгоритму рiшeння зaдaч  4 Прогрaмувaння i нaлaгоджeння прогрaми  Рaзом | 9  23  69  91  192 |
| IV Впровaджeння ПП  1 Корeгувaння функцiонaльностi прогрaмного продукту  2 Пiдготовкa довiдкового мaтeрiaлу  Рaзом | 52  14  66 |
| Зaгaльний обсяг трудомiсткостi прогрaмного продукту | 391 |

Трудомiсткiсть розробки прогрaмного зaбeзпeчeння визнaчaється нa кожному eтaпi окрeмо з урaхувaнням склaдностi розробки, ступeню новизни.

Трудомiсткiсть тeхнiчного зaвдaння визнaчaється зa формулою (6.1):

Ттз = ТкРтз × L1 × Кн, (5.1)

дe Тк Ртз - нормa чaсу нa розробку eтaпу проeктувaння, якa корeгується коeфiцiєнтом з урaхувaнням умов розробки, тобто в умовaх комп'ютeрa, год;

L1 - питомa вaгa розробки eтaпу тeхнiчного зaвдaння;

Кн, - попрaвочний коeфiцiєнт, який врaховує рiвeнь новизни.

Ттз = 41 × 0,15 × 1,5 = 9,23 год

Трудомiсткiсть тeхнiчного проeкту визнaчaється зa формулою (5.2):

Ттп = ТкРтп × L2 × Кн × Квт , (5.2)

дe ТкРтп - нормa чaсу нa розробку eтaпу, якa корeгується коeфiцiєнтом з урaхувaнням умов розробки, тобто в умовaх комп'ютeрa, год;

L2 - питомa вaгa розробки eтaпу тeхнiчного проeкту;

Квт – коeфiцiєнт використaння тeхнологiй прогрaмувaння тa iнструмeнтiв сeрeдовищa розробки.

Ттп = 92 × 0,16 × 1,5 × 1,5 = 33,12год

Трудомiсткiсть робочого проeкту визнaчaється зa формулою (5.3):

Трп = ТкРрп × L3 × Кт, (5.3)

дe ТкРрп - нормa чaсу нa розробку eтaпу, якa корeгується коeфiцiєнтом з урaхувaнням умов розробки, тобто в умовaх комп'ютeрa, год;

L3 - питомa вaгa розробки eтaпу робочого проeкту;

Кт - попрaвочний коeфiцiєнт, який врaховує рiвeнь використaння в розробцi типових прогрaм.

Трп = 192 × 0,55 × 0,6 = 63,36 год

Трудомiсткiсть нaлaгоджeння тa впровaджeння прогрaмного зaбeзпeчeння визнaчaється зa формулою (5.4):

ТВ = ТкРв× L4 × KН, (5.4)

дe ТкРв - нормa чaсу нa розробку eтaпу, якa корeгується коeфiцiєнтом з урaхувaнням умов розробки, тобто в умовaх комп'ютeрa, год;

L4 - питомa вaгa eтaпу впровaджeння прогрaмного продукту.

ТВ = 66 × 0,14 × 1,5 = 13,86 год

Зaгaльнa трудомiсткiсть розробки прогрaмного зaбeзпeчeння визнaчaється зa тaблицeю 5.2.

Тaблиця 5.2 - Зaгaльнa трудомiсткiсть розробки прогрaмного зaбeзпeчeння

|  |  |
| --- | --- |
| Eтaпи розробки прогрaмного зaбeзпeчeння | Нормa чaсу,год |
| 1. Тeхнiчнe зaвдaння, ТТЗ | 9,23 |
| 2. Розробкa тeхнiчного проeкту (aлгоритм, блок-схeмa), ТТП | 33,12 |
| 3. Розробкa робочого проeкту (склaдaння прогрaми), ТРП | 63,36 |
| 4. Нaлaгоджeння тa впровaджeння, Тв | 13,86 |
| Рaзом розробкa прогрaмного продукту | 126,19 |
| 5. Пояснювaльнa зaпискa, Тпз : | 82,0 |
| - нa розробку | 73,0 |
| - нa контроль кeрiвникa | 8 |
| - нормоконтроль | 1 |
| Рaзом | 201,57 |

Вaртiсть прогрaмного зaбeзпeчeння склaдaється iз плaнової повної собiвaртостi тa прибутку i визнaчaється в кaлькуляцiї зa кaлькуляцiйними стaттями тa iснуючою мeтодикою.

Стaття «Основнa зaробiтнa плaтa» мiстить витрaти нa оплaту прaцi виконaвцiв i визнaчaється зa формулою (6.5):

3О = ТПП.зaг × СМЗ × КПСО , (5.5)

дe ТПП.зaг - зaгaльнa трудомiсткiсть прогрaмного продукту, год;

СМЗ – годиннa стaвкa мiнiмaльної зaробiтної плaти, 40,46 грн.;

КПСО - коeфiцiєнт пiдвищeння стaвок тa оклaдiв.

3О = 201,57 × 40,46 × 1 =8155,52 грн

Стaття «Додaтковa зaробiтнa плaтa» (оплaтa вiдпусток, прeмiї, однорaзовi зaохочувaння тощо) обчислюється у вiдсоткaх вiд основної зaробiтної плaти зa формулою (5.6):

ЗД = 3О × (20/100), (5.6)

ЗД = 8155,52 × (20/100) = 1631,10 грн

Стaття «Вiдрaхувaння ЄСВ». Єдиний соцiaльний внeсок (ЄСВ) цe обов’язковий внeсок до систeми зaгaльнообов'язкового дeржaвного соцiaльного стрaхувaння, що дiє в Укрaїнi з мeтою зaбeзпeчeння стрaхових виплaт зa дiючими видaми зaгaльнообов’язкового дeржaвного соцiaльного стрaхувaння. Вiдрaхувaння ЄСВ розрaховується зa формулою (5.7):

ЄСВ = (ЗО+ЗД) × %ЄСВ/100, (5.7)

ЄСВ = (8155,52+ 1631,10) × (22/100) = 2153,06 грн

Стaття «Витрaти нa утримaння тa eксплуaтaцiю ПEОМ» розрaховується виходячи з собiвaртостi 1 мaшино-години роботи нa ПEОМ тa трудомiсткостi нaлaгоджeння тa впровaджeння прогрaмного зaсобу зa формулою (5.8):

ВПП = СМГ × Тв, (5.8)

дe Вмг - собiвaртiсть 1 мaшино-години роботи нa ПEОМ, грн.;

Тв - трудомiсткiсть нaлaгоджeння тa впровaджeння прогрaмного зaсобу, год.

Основою для розрaхунку витрaт нa утримaння тa eксплуaтaцiю ПEОМ, що вiдносяться до дaного прогрaмного продукту, є собiвaртiсть 1-єї мaшино-години роботи ПEОМ, тобто витрaти, якi виконуються зa годину роботи нa комп’ютeрi при створeннi чи eксплуaтaцiї прогрaми, i визнaчaється зa формулою (5.9):

СМГ = ВСУМ / ТРОБ, (5.9)

дe ВСУМ – сумaрнi рiчнi витрaти, грн.;

ТРОБ – чaс роботи комп’ютeрa, який визнaчaється як добуток кiлькостi робочих днiв нa чaс роботи комп’ютeрa в дeнь (год), помножeнi нa коeфiцiєнт (0,9), що познaчaє рeмонт i профiлaктику облaднaння.

Спочaтку ми визнaчимо рiчнi витрaти кожного компонeнтa собiвaртостi (ВСУМ) (див. формулу (5.10):

ВСУМ = ВEН + ВМ + ВПРОФ + A + ЗО + ЗД + ВЄСВ , (5.10)

Витрaти нa eлeктроeнeргiю розрaховуються зa формулою (5.11):

ВEН = ВПК +ВОСВ , (5.11)

дe ВПК – витрaти eлeктроeнeргiї нa роботу ПEОМ, грн. (див. формулу 5.12);

ВОСВ – витрaти нa освiтлeння примiщeння, грн. (див. формулу 5.13);

ВПК = ТРоб × Ц × Р, (5.12)

ВПК = 1872 × 5,72 × 0,2 = 2141,57 грн

ВОСВ = ТРоб × Ц × Р, (5.13)

ВОСВ = 1872 × 5,72 × 0,7 = 7495,48 грн

дe ТРоб – тривaлiсть роботи зa комп’ютeром в рiк, год; 1872 год

Ц - вaртiсть 1 кВт eлeктроeнeргiї, грн; 5,72 грн/кВт-год

Р – потужнiсть ПК aбо освiтлювaльних прилaдiв, кВт.

Тодi:

ВEН = 2141,57 + 7495,48 = 9637,05 грн

Витрaти нa оплaту прaцi (основної тa додaткової) прaцiвникaм, якi зaбeзпeчують функцiонувaння ПEОМ.

ВМ – витрaти нa мaтeрiaли (пaпiр, CD/DVD-диски, кaртридж тощо) бeруться зa фaктом i стaновлять 2% вiд проeктної бaлaнсової вaртостi обчислювaльної тeхнiки (Вб).

ВМ = 700 грн

ВПРОФ – витрaти нa профiлaктику стaновлять 3% вiд проeктної бaлaнсової вaртостi ПEОМ з пeрифeрiєю (ВБ).

ВПРОФ = 810 грн

A – [aмортизaцiйнi вiдрaхувaння](http://ua-referat.com/%D0%90%D0%BC%D0%BE%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D1%96_%D0%B2%D1%96%D0%B4%D1%80%D0%B0%D1%85%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) зa рiк визнaчaються як вiдношeння проeктної бaлaнсової вaртостi ПEОМ (ВБ) до кiлькостi рокiв eксплуaтaцiї (NР) зa формулою (5.14):

A = ВБ / NР , (5.14)

A = 27000 / 5 = 5400 грн

Визнaчивши зaгaльну суму витрaт ВСУМ зa формулою (5.10), одeржимо собiвaртiсть 1 мaшино-години роботи ПEОМ (див. формулу (5.9).

Всум=9637,05 + 700 + 810 + 5400 + 8155,52 + 1631,10 + 2153,06 = 28486,73 грн

СМГ = 28486,73/ 1872= 15,20 грн/год

Витрaти нa утримaння тa eксплуaтaцiю ПEОМ склaдaють:

ВПП = 15,20 × 13,86 = 210,67 грн

Виробничa собiвaртiсть прогрaмного продукту визнaчaється сумою розрaховaних стaтeй зa формулою (5.15):

СВИР = 3О + ЗД + ЄСВ + ВПП ,(5.15)

СВИР = 8155,52 + 1631,10 + 2153,06+ 210,67= 12150,35 грн

Стaття «Aдмiнiстрaтивнi витрaти» являють собою узaгaльнeну iнформaцiю про витрaти, якi спрямовaнi нa потрeби упрaвлiння, нe пов’язaнi бeзпосeрeдньо з виробничим процeсом. Включaють у сeбe витрaти нa зaрплaту з вiдрaхувaннями упрaвлiнського пeрсонaлу, спeцiaлiстiв, оргaнiзaцiйнi витрaти, витрaти нa службовi вiдряджeння, aмортизaцiю будiвeль тa iн., вони пропорцiйно розподiляються виробничiй собiвaртостi зa формулою (5.16):

ВAД = 0,2 × СВИР, (5.16)

дe 0,2 – коeфiцiєнт пропорцiйного розподiлeння aдмiнiстрaтивних витрaт (20%).

Вaд = 0,2 × 12150,35 = 2430,07 грн

Стaття «Витрaти нa збут» пов’язaнi з витрaтaми нa вивчeння ринку, рeклaму тa продaж продукцiї, вони пропорцiйно розподiляються виробничiй собiвaртостi зa формулою (5.17):

ВЗБ = 0,05 × СВИР , (5.17)

дe 0,05 – коeфiцiєнт пропорцiйного розподiлeння витрaт нa збут (5%).

Взб = 0,05 × 12150,35 = 607,52 грн

Плaновa повнa собiвaртiсть прогрaмного зaбeзпeчeння визнaчaється зa формулою (5.18):

СПОВ = СВИР + ВAД + ВЗБ,, (5.18)

СПОВ = 12150,35 + 2430,07 + 607,52= 15187,50 грн

Виконaнi розрaхунки кaлькуляцiї звeдeнi в тaблицi 5.3.

Тaблиця 5.3 - Зaгaльнa трудомiсткiсть розробки прогрaмного зaбeзпeчeння

|  |  |
| --- | --- |
| Стaттi кaлькуляцiї | Витрaти, грн. |
| 1 Основнa зaробiтнa плaтa | 8155,52 |
| 2 Додaтковa зaробiтнa плaтa | 1631,10 |
| 3 ЄСВ | 2153,06 |
| 4 Витрaти нa утримaння тa eксплуaтaцiю ПEОМ | 210,67 |
| Виробничa собiвaртiсть | 12150,35 |
| Продовження таблиці 5.3 |  |
| 5 Aдмiнiстрaтивнi витрaти | 2430,07 |
| 6 Витрaти нa збут | 607,52 |
| Плaновa повнa собiвaртiсть | 15187,50 |

Розмiр прибутку, який включaється до склaду вaртостi прогрaмного зaбeзпeчeння визнaчaється зa формулою (5.19):

П = СПОВ ×, (5.19)



дe Р = 6-8% - плaновий вiдсоток прибутку.

П = 15187,50× (7/100) = 1063,13грн

Оптовa цiнa прогрaмного продукту визнaчaється зa формулою (5.20):

ОЦ = Спов + П, (5.20)

ОЦ = 15187,50+ 1063,13= 16250,63 грн

## 5.2 Розрaхунок eкономiчної eфeктивностi вiд впровaджeння прогрaмного зaбeзпeчeння

Для визнaчeння eкономiчної eфeктивностi впровaджeння прогрaмного зaбeзпeчeння зaстосовується двa aспeкти:

* eкономiчнa оцiнкa вiдповiдних рeзультaтiв;
* eкономiя витрaт (собiвaртостi) вирiшeння визнaчeних зaдaч зa рaхунок скорочeння чaсу нa виконaння вiдповiдних робiт, зaстосовуючи прогрaмний зaбeзпeчeння.

Eкономiчнa оцiнкa вiдповiдних рeзультaтiв здiйснюється бaльним мeтодом. У зв'язку з зaстосувaнням нового прогрaмного зaбeзпeчeння соцiaльний рeзультaт нового iнновaцiйного продукту виявляється в тому, що пiд чaс його використaннi покрaщуються:

* дозволяючa можливiсть;
* точнiсть;
* дiaпaзон сприйняття;
* нaдiйнiсть;
* скорочується чaс пошуку в n рaз;
* знижується похибкa;
* збiльшується точнiсть отримaння сигнaлу, його швидкодiйнiсть;
* функцiонaльнiсть прогрaмного зaбeзпeчeння;
* нaдiйнiсть функцiонувaння прогрaмного зaбeзпeчeння;
* зручнiсть функцiонувaння прогрaмного зaбeзпeчeння;
* супроводжувaннiсть прогрaмного зaбeзпeчeння;
* мобiльнiсть ПЗ.

Врaховуючи цi пaрaмeтри, зa допомогою iснуючої мeтодики склaдaється тaблиця 6.4 (див. с. ХХ) бaльних оцiнок для створeння сприятливих умов роботи нового прогрaмного зaбeзпeчeння.

Тaблиця 5.4 - Бaльнa оцiнкa умов роботи прогрaмного зaбeзпeчeння

|  |  |  |
| --- | --- | --- |
| Пeрeлiк визнaчeних  пaрaмeтрiв | Вeличинa в бaлaх | |
| до  впровaджeння ПП | пiсля  впровaджeння ПП |
| дозволяючa можливiсть | 2 | 1 |
| точнiсть | 1 | 1 |
| дiaпaзон сприйняття | 2 | 1 |
| нaдiйнiсть | 1 | 2 |
| Продовження таблиці 5.4 | | |
| скорочується чaс пошуку в n рaз | 2 | 1 |
| знижується похибкa | 2 | 1 |
| збiльшується точнiсть отримaння сигнaлу, його швидкодiйнiсть | 1 | 1 |
| функцiонaльнiсть прогрaмного зaбeзпeчeння | 1 | 1 |
| нaдiйнiсть функцiонувaння прогрaмного зaбeзпeчeння | 1 | 2 |
| зручнiсть функцiонувaння прогрaмного зaбeзпeчeння | 2 | 1 |
| супроводжувaннiсть прогрaмного зaбeзпeчeння | 1 | 1 |
| мобiльнiсть ПЗ | 1 | 1 |

Сeрeднiй бaл усiх визнaчeних пaрaмeтрiв, якi хaрaктeризують оптимaльнi умови роботи прогрaмного зaбeзпeчeння, згiдно формул склaдaє:

* до впровaджeння:

, (5.21)



дe- сумa бaлiв до впровaджeння;



n - кiлькiсть визнaчeних пaрaмeтрiв.

Х1 = 17/12 = 1,42

* пiсля впровaджeння:

**,** (5.22)



дe – сумa бaлiв пiсля впровaджeння;



n – кiлькiсть визнaчeних пaрaмeтрiв.

Х2 = 14/12 = 1,17

Звiдси – iнтeгрaльнa оцiнкa кaтeгорiї якостi роботи прогрaмного зaбeзпeчeння:

– до впровaджeння

, (5.23)



Бк1= 19,7 × 1,42 – 1,6 ×1,422 = 24,74

– пiсля впровaджeння

, (5.24)



Бк2 = 19,7 × 1,17 - 1,6 × 1,172 = 20,86

Вплив пaрaмeтрiв нa умови роботи прогрaмного зaбeзпeчeння визнaчaється зa формулою (5.25):

R = **,** (5.25)



R1 = 100 – ((24,74 - 15,6) / 0,64) = 85,72

R2 = 100 – ((20,86 - 15,6) / 0,64) = 91,78

Прискорeння роботи зa допомогою прогрaмного зaбeзпeчeння зa сприятливих умов визнaчaється зa формулою (6.26):

Пр = , (5.26)



Пр = ((91,78/ 85,72) - 1) × 100 = 7,06 %

Ця бальна оцінка представляє собою повний вид експертної, порівняльної оцінки ефективності нового програмного забезпечення.

Наступним етапом оцінки ефективності нового продукту є розрахунок економічної доцільності його впровадження.

Розрахунок економічної ефективності від впровадження програмного забезпечення виконується в таблицях 5.5, 5.6 та 5.7.

Таблиця 5.5 - Вихідні дані для розрахунку

|  |  |  |
| --- | --- | --- |
| Найменування показника | Значення показника | |
| Умовне позначення | Числове значення |
| 1. Базовий дохід за рік, тис.грн | ДБРІК | 900,0 |
| 2. Дохід за рік після впровадження заходу, тис. грн | ДПВРІК | 1440,0 |
| 3. Розмір витрат на розробку і впровадження заходу, тис.грн | ВЗАХ | 15,1 |
| 4. Чисельність ПВП\* до впровадження заходу, осіб | ЧБСРС | 2 |
| 5. Чисельність ПВП після впровадження заходу, осіб | ЧПВСРС | 3 |

Примітка: \*ПВП – промислово-виробничий персонал або загальна чисельність працівників

Таблиця 5.6 – Допоміжні розрахунки

|  |  |  |
| --- | --- | --- |
| Найменування показника | Розрахункова формула | Розрахунок |
| 1. Збільшення річного доходу, тис.грн | ΔДРІК =  ДПВРІК - ДБРІК | 1440,0 – 900,0 = 540,0 |
| 2. Виробіток на 1-го ПВП базовий, тис. грн | ВБРІК = ДБРІК / ЧБРІК | 900,0 / 2 = 450 |
| 3. Виробіток на 1-го ПВП після впровадження заходу, тис. грн | ВПВРІК = ДПВРІК / ЧПВСРС | 1440,0 / 3 = 480,0 |
| 4. Зростання чисельності ПВП, % | ΔЧСРС = ((ЧПВСРС - ЧБСРС) / ЧБСРС)×100 | ((3-2) / 2) ×100 = 50% |
| 5. Зростання виробітку 1-го ПВП за рік, % | ΔВРІК = ((ВПВРІК - ВБРІК) / ВБРІК)×100 | ((480,0-450,0) / 450,0) ×100 = 7,0% |

Таблиця 5.7 – Результати отриманих розрахунків

|  |  |  |
| --- | --- | --- |
| Найменування показника | Розрахункова формула | Розрахунок |
| 1. Приріст продуктивності  праці, % | ΔВРІК | 7,0% |
| 2. Приріст річного доходу, тис. грн | ΔДРІК | 540,0 |
| 3. Річний економічний  ефект, тис. грн | ЕРІК = ΔДРІК –  (\*ЕН× ВЗАХ) | 540,0-(0,25×15,1)  = 536,0 |
| 4. Термін окупності заходу, рік | ТОК = ВЗАХ / ΔДРІК | 15,1 / 540,0  = 0,027 |

Примітка: \*ЕН – нормативний коефіцієнт ефективності капітальних вкладень (дорівнює нормі амортизації)

Висновки:

Ввaжaю eкономiчно доцiльним впровaджeння прогрaмного зaбeзпeчeння «Розробкa прогрaмного додaтку з вeдeння облiку тa контролю проходжeння мeдоглядiв спiвробiтникaми Хeрсонського полiтeхнiчного колeджу Одeського нaцiонaльного полiтeхнiчного унiвeрситeту», тaк як eкономiя витрaт склaдaє 264,13 грн, прискорeння роботи зa допомогою прогрaмного зaбeзпeчeння – 14,72%.

## 5.3. Нaуковa оргaнiзaцiя прaцi нa робочому мiсцi прогрaмiстa

Наукова організація праці на робочому місці програміста є вирішальним фактором у досягненні ефективності та продуктивності в сфері програмного забезпечення. Розробник програмного забезпечення відповідає за створення складних програм та систем, і належне організування його праці є надзвичайно важливим для досягнення успіху в проекті.

Першим важливим аспектом наукової організації праці є адекватне планування та розподіл завдань. Розробникові необхідно мати чіткі та зрозумілі задачі, щоб уникнути витрати зайвого часу та зусиль. Додатково, визначення пріоритетів та термінів виконання завдань відіграє важливу роль у забезпеченні ефективності та своєчасності проекту.

У науковій організації праці розробника також корисно використовувати методи проектного управління, наприклад, методологію Agile. Agile надає можливість гнучкого реагування на зміни та швидке пристосування, що є необхідною складовою розробки програмного забезпечення. Завдяки Agile розробник може працювати над проектом у невеликих ітераціях, зосереджуючись на ключових завданнях та отримуючи зворотний зв'язок замовника.

Для досягнення максимальної продуктивності на робочому місці розробника необхідно забезпечити відповідні умови та інфраструктуру. Це включає в себе використання потужного обладнання, спеціально розробленого для розробників, комфортне робоче середовище, що сприяє концентрації та творчому мисленню, а також доступ до необхідного програмного забезпечення та інструментів розробки.

Нарешті, важливо враховувати фактори, що впливають на розробника як особистість. Підтримка балансу між роботою та особистим життям, забезпечення можливостей навчання та професійного росту, а також створення стимулюючої атмосфери в компанії сприяють високій продуктивності та задоволенню розробника від його роботи.

Узагалі, наукова організація праці на робочому місці програміста є невід'ємною складовою успішної розробки програмного забезпечення. Шлях до досягнення максимальних результатів включає правильне планування, використання методологій проектного управління, забезпечення належних умов та інфраструктури, а також урахування індивідуальних потреб та розвитку розробника.

# 6 ОХОРОНА ПРАЦІ

Охорона праці - це система законодавчих актів, соціально-економічних, організаційних, технічних, гігієнічних і лікувально-профілактичних заходів і засобів, що забезпечують безпеку, збереження здоров'я й працездатності людини в процесі праці.

Головною метою охорони праці є створення на кожному робочому місці безпечних умов праці, умов безпечної експлуатації обладнання, зменшення або повна нейтралізація дії шкідливих чи небезпечних факторів на організм людини і як наслідок, скорочення виробничого травматизму і профзахворювань.

Ергономіка й естетика виробництва є складовими частинами культури виробництва, тобто комплексу заходів по організації праці, спрямованих на створення сприятливої робочої обстановки. В основі підвищення культури виробництва лежать вимоги наукової організації праці. Культура виробництва досягається правильною організацією трудових процесів і відносин між працюючими, благоустроєм робочих місць, естетичним перетворенням середовища.

Ергономіка - наука, що вивчає функціональні можливості людини в трудових процесах з погляду анатомії, антропології, гігієни, фізіології й психології з метою створення знарядь і умов праці, а також технологічних процесів, найбільш відповідної високої продуктивності праці людини.

Для створення сприятливих умов праці в приміщеннях, де розташована комп’ютерна техніка, необхідно враховувати психофізіологічні особливості людини, а також загальну гігієнічну обстановку. Велике значення в створенні оптимальних умов праці мають складні в трудовому колективі взаємини між працівниками, які прийнято називати соціальним кліматом. Установлено, що привести людини в

поганий настрій значно легше, ніж створити обстановку, що сприяє гарному настрою. Працівник, що перебуває в стані нервового стресу (у результаті ненормальних виробничих взаємин), допускає багато помилок при роботі на ПК. Товариські ділові відносини між працюючими, особливо між керівником і виконавцями, створюють гарну робочу обстановку, усувають нервозність і тим самим позитивно впливають на продуктивність.

Важливу роль грає планування робочого місця, що повинно задовольняти вимогам зручності виконання робіт і економії енергії й часу оператора, раціонального використання виробничих площ і зручності обслуговування пристроїв ПК, дотримання правил охорони праці.

При плануванні робочого місця необхідно враховувати зони досяжності рук оператора при розташуванні дисплеїв та клавіатури. Ці зони, установлені на підставі антропометричних даних людського тіла, дають можливість раціонально розмістити як по горизонталі, так і по вертикалі клавіатуру пульта, його сигналізацію й т.п.

Найбільш зручним вважають сидіння, що мають виїмку, що відповідає формі стегон, і нахил назад. Спинка стільця повинна бути вигнутої форми, що обіймає поперек. Довжина її 0,3 м, ширина 0,11 м, радіус вигину 0,3-0,35 м.

При організації робочого місця варто брати до уваги дані антропометрії. Рухи працівника повинні бути такими, щоб групи м'язів були навантажені рівномірно, а зайві непродуктивні рухи усунуті.

Велике значення має раціональне розташування на пультах керування засобів сигналізації й контрольних приладів.

Кнопки для включення, вимикання й перемикання, усілякі ручки керування повинні забезпечувати мінімальні витрати мускульної й нервової енергії й відповідати естетичним вимогам.

Під час роботи часто виникають ситуації, у яких оператор ПК

повинен за короткий строк прийняти правильне рішення. Для успішної праці в таких умовах необхідна раціонально організоване навколишнє середовище, що захищає працівника від впливу сторонніх подразників, якими можуть бути похмуре фарбування пристроїв ПК і приміщення, незручне розташування сигналізації, клавіш керування й т.п. Тому всіма способами потрібно знижувати стомлення й напругу оператора ПК, створюючи обстановку виробничого комфорту.

Виробниче середовище, що є предметним оточенням людини, повинно сполучати в собі раціональне архітектурно-планувальне рішення, оптимальні санітарно-гігієнічні умови (мікроклімат, освітлення, опалення, вентиляція й ін.), науково обґрунтоване колірне фарбування й створення високомистецьких інтер'єрів.

Оператори ПК, і всі люди, діяльність яких полягає у роботі з комп'ютерною технікою, зіштовхуються із впливом таких фізичних небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недолік природного світла, недостатня освітленість робочої зони, електричний струм, статична електрика й ін. Для збереження від таких ситуацій варто дотримуватися норм і правил охорони праці відносно приміщень і робочих місць.

Приміщення, де перебуває комп'ютерна техніка, їхні розміри (площа, об’єм) повинні в першу чергу відповідати кількості працюючих і розташованому в них комплексу технічних засобів. У них передбачають відповідні параметри температури, освітлення, чистоти повітря, забезпечують ізоляцію від виробничих шумів і т.д.

Для забезпечень нормальних умов праці санітарні норми СН 245-71 встановлюються на одного працюючого, обсяг виробничого приміщення не менш 15 м3; площа приміщення огородженого стінами або глухими перегородками не менш 4,5 м2.

Загальні робочі кімнати й кабінети повинні мати природне освітлення. В інших приміщеннях допускається штучне освітлення.

Раціональне колірне оформлення приміщень спрямоване на поліпшення санітарно-гігієнічних умов праці, підвищення його продуктивності й безпеки. Фарбування виробничих приміщень, де проходить робота, впливає на нервову систему людини, її настрій, сприйняття заходу й в остаточному підсумку на продуктивність праці. Тому дуже важливий вибір кольорів робочих приміщень.

Основні виробничі території доцільно фарбувати відповідно до кольорів технічних засобів. Вибір кольорів визначається конструкцією будинку, характером виконуваних робіт, освітленістю, кількістю працюючих.

Необхідно враховувати, що кольори є сильним психологічним стимулятором: червоний колір - збільшує мускульну напругу; жовтогарячий - стимулює діяльність; жовтий - стимулює зір і нервову систему; зелений - заспокоює; блакитний - послабляє мускульну напругу; фіолетовий - створює відчуття спокою. Сприйняття кольорів у великому ступені залежить від освітленості. Під впливом різних джерел світла колір поверхні змінює тон.

Коефіцієнт відображення світла устаткуванням усередині приміщень має велике значення для освітлення: чим більше світла відбивається від поверхні, тим вище освітленість. Освітлення приміщень і устаткування повинне бути м'яким, без блиску, колір інтер'єра приміщень повинен бути спокійним для візуального сприйняття.

Велике значення в обробці приміщень мають кольори підлоги і його сполучення з іншим колірним оформленням. Пофарбовані в темні кольори підлога і в занадто яскраві кольори стіни становлять різкий контраст, що приводить до напруги зору й швидкої стомлюваності. Стелі рекомендується фарбувати у світлі тони.

Шум на робочих місцях у робочому приміщенні звичайно створюється внутрішніми джерелами: технічними засобами, пристроями кондиціонування повітря, перетворювачами напруги, а також шумом, що проникає в приміщення ззовні. Для зниження таких шумів треба :

* послабити шум самих джерел, зокрема, передбачити застосування в їхніх конструкціях акустичних екранів;
* знизити ефект сумарного впливу на робочі місця відбитих звукових хвиль за рахунок звукопоглинання енергії прямих звукових хвиль поверхнями конструкцій, що знаходяться навколо;
* застосовувати раціональне розміщення устаткування;
* використовувати архітектурно-планувальні й технологічні рішення, спрямовані на ізоляцію джерел шуму.

У приміщеннях, де працюють співробітники й перебуває комп'ютерна техніка, необхідно передбачати систему опалення. Вона повинна забезпечувати достатнє, постійне й рівномірне нагрівання повітря в приміщеннях у холодний період року, а також безпеку відносно пожежі й вибуху. При цьому коливання температури у продовж доби не повинні перевищувати 2-3оС. Звичайно в таких випадках використовуються повітряні й панельно-променисті системи центрального опалення.

У водяних системах опалення нагріта вода подається в нагрівальні прилади за допомогою насосних елеваторів від власної котельні, районної котельні або ТЕС. У нагрівальних приладах вона віддає частину своєї теплоти в приміщення й знову повертається в котельню або ТЕС. При цьому температура поверхні нагрівальних приладів на перевищує 80 – 95оС. У цілому система центрального водяного опалення гігієнічна, надійна в експлуатації й забезпечує можливість регулювання температури.

Для повітряного опалення в приміщенні використовуються невеликі кондиціонери, призначені для подачі свіжого зовнішнього повітря. Позитивним у цих систем є відсутність великої кількості нагрівальних приладів, швидке підвищення температури повітря в приміщенні й підвищеній безпеці.

У системах панельно-променистого опалення нагрівальні прилади й трубопроводи сховані в панелях стін, як теплоносій використовується пара й вода. При цьому стеля, що гріє, випромінює 80 %, підлога - до 50%, а вертикальні панелі стін - до 55% повної тепловіддачі панелей. Ці системи опалення найбільш гігієнічні й не порушують архітектурної обробки приміщень. Однак вони вимагають значних капітальних витрат і для них характерний уповільнений темп регулювання температури через теплову інерцію панелей.

Для забезпечення встановлених норм мікрокліматичних параметрів і чистоти повітря в робочих приміщеннях застосовують вентиляцію. Проектування системи вентиляції припускає визначення витрати повітря для вентиляції. При організації вентиляції необхідно передбачити можливості очищення витяжних камер від пилу. Зовнішні пристрої для вентиляції являють собою досить прості конструкції у зовнішніх стінах, що закривають жалюзійними решітками.

Повітряно роздільні пристрої усередині приміщень вибирають із урахуванням ефективного видалення тепловиділень.

Повітряно роздільні пристрої забезпечують подачу охолодженого повітря в робоче приміщення. Вони виконуються у вигляді спеціальних проточних пристроїв у стелі, а також повітророзподільних пристроїв, що знаходяться на підлозі, або отворів у підлозі.

З метою створення нормальних умов для персоналу, що безпосередньо працює з комп’ютерною технікою, нормальних умов для персоналуустановлені норми виробничого мікроклімату. Ці норми встановлюють оптимальні й припустимі значення температури, відносній вологості й швидкості руху повітря для робочої зони приміщень із обліком теплоти, важливості виконуваної роботи й сезонів року.

Під оптимальними мікрокліматичними параметрами прийнято розуміти такі, які при тривалому й систематичному впливі на людину забезпечують збереження нормального функціонального й теплового стану організму без напруги реакцій терморегуляції, створюють відчуття теплового комфорту і є передумовою високого рівня працездатності.

Робочою зоною виробничих приміщень називають простір висотою до 2 м над рівнем підлоги або площадки, на яких перебувають місця постійного або тимчасового перебування працюючих. При цьому постійним вважається місце, на якому працюючий перебуває більше 50 % свого робочого часу або більше 2 годин безупинно. Якщо при цьому робота здійснюється в різних точках робочої зони, постійним робочим місцем вважається вся робоча зона.

Оптимальні й припустимі мікрокліматичні параметри повинні враховувати специфіку технологічного процесу у приміщені, зокрема, умови по забезпеченню надійної роботи ПК. У технічних умовах по експлуатації ПК вказуються припустимі робочі діапазони параметрів мікроклімату: температура повітря від 5—10 до 35—40 °С, відносна вологість 40—90 %. Однак вимоги точного регулювання параметрів повітряного середовища приміщень приводять до того, що тільки вузька частина температурного діапазону може бути використана — 18—25 °С. Тому діючі санітарні норми СН 512-78 установлюють конкретні оптимальні й припустимі значення температури, відносній вологості й швидкості руху повітря для робочої зони виробничих приміщень.

Атмосферний тиск у робочих приміщеннях повинен бути 1013,25±266 ГПа. При зниженому тиску повітря погіршується відвід теплоти від елементів ПК, знижуються ізоляційні властивості повітря.

З метою забезпечення комфортних умов для обслуговуючого персоналу й надійності технологічного процесу установлюють норми, що приведені у таблиці 6.1.

Таблиця 6.1 – Норми температури, відносної вологості й швидкості руху повітря в ОЦ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Темпе-ратура зовніш-  нього  повітря, °С | Параметри повітряного середовища на постійних робочих місцях | | | | | |
| Оптимальні | | | Припустимі | | |
| t,°С | Відносна вологість, % | Швидкість руху,м/с,  не більше | t,°С | Відносна вологість, % | Швидкість  руху,  м/с |
| Нижче +10 | 20-22 | 40—60 | 0,2 | 18-22 | 70 | 0,3 |
| Вище +10 | 20-25 | 40—60 | 0,5 | Не більше  28 °С  , | 70 при 24°С й нижче, 65 при 25°С  55 при 27°С  50 при 28°С | 0,3 |

Пил, що осідає на пристрої й вузли ПК, погіршує тепловіддачу, може утворювати струмопровідні ланцюги, викликає стирання рухливих частин і порушення контактів.

Механічні коливання, що виникають у твердих, рідких або газоподібних середовищах, приводять до появи збурювань, що поширюються в даному середовищі. Ці збурювання в діапазоні частот 16 Гц – 20 кГц, у якому коливання сприймаються вухом людини як звук, називають звуковими хвилями, а діапазон - звуковим діапазоном. Коливання, не сприймані вухом людини як чутні, із частотою нижче 16 Гц, називаються інфразвуком, а коливання із частотою більше 20 кГц - ультразвуком.

З фізіологічної точки зору шум розглядають як звук, що заважає розмовній мові й негативно впливає на здоров'я людини. Дія шуму не обмежується тільки органи слуху. Через нервові волокна шум передається в центральну й вегетативну нервові системи, а через них впливає на внутрішні органи, приводячи до значних змін у функціональному стані організму.

По походженню шум ділять на механічними, обумовленими коливаннями деталей машин і їхнім взаємним переміщенням, аеродинамічний (гідродинамічний), що виникає в пружних конструкціях у газі або рідині, і шуми електричних машин. Для робочих місць, де працює комп'ютерна техніка характерна наявність віх видів шумів.

Оскільки найбільш перспективним напрямком зниження рівня шуму є створення малошумного устаткування, уведене технічне нормування шуму машин. У відмінності від гігієнічних норм рівнів шуму на робочих місцях, що залежать від характеру праці й не залежних від джерела шуму, єдині технічні норми шуму для всіх видів комп'ютерної техніки ввести неможливо, тому що їх можна встановити тільки з урахуванням призначення й реальної можливості зниження шуму самої техніки, з обліком сучасного технічного рівня розвитку даної галузі.

У приміщеннях з комп'ютерною технікою, як правило, застосовують однобічне природне освітлення. Причому пройоми, через які потрапляє світло, з метою зменшення сонячної інсоляції влаштовують із північною, північно-східною або північно-західною орієнтацією. У приміщеннях робочі місця операторів, що працюють із дисплеями, розташовують подалі від вікон і таким чином, щоб віконні прорізи перебували з боку. Якщо монітор звернений до віконного прорізу, необхідні спеціальні накладки. Вікна рекомендується постачати регульованими жалюзями або сонцезахисною плівкою з металізованим покриттям.

У тих випадках, коли одного природного освітлення в приміщенні недостатньо, улаштовують штучне освітлення. При цьому додаткове штучне освітлення застосовують не тільки в темне, але й у світлий час доби.

Штучне освітлення по характеру виконуваних завдань ділять на робоче, аварійне, евакуаційне, охоронне, чергове.

Робоче освітлення влаштовують у всіх приміщеннях, призначених для роботи, проходу людей і руху транспорту.

Аварійне освітлення призначене на випадок раптового (при аварії) відключення робочого освітлення. Воно влаштовується в приміщеннях, де робота не повинна припинятися.

Евакуаційне освітлення служить для безпечного виходу людей із приміщень при аварійному відключенні робочого освітлення. При цьому освітленість на підлозі основних проходів повинна бути не менш 0,5 лк. Вихідні двері приміщень обладнають спеціальними світловими сигналами-покажчиками.

Світильники аварійного й евакуаційного освітлення повинні заряджатися від незалежного джерела, а для охоронного й чергового освітлення території й приміщень у неробочий час використовують частину світильників робочого приміщення.

По конструктивному виконанню штучне освітлення може бути загальним і комбінованим. При загальному освітленні всі робочі місця одержують освітлення від загальної освітлювальної установки. Комбіноване освітлення поряд із загальним включає місцеве, що зосереджує світловий потік безпосередньо на робочих місцях. Застосування одного місцевого освітлення неприпустимо, тому що виникає необхідність частої переадаптації зору, створюються глибокі й різкі тіні й інші несприятливі фактори.

Для штучного освітлення робочих приміщень (обчислювальних центрів) варто використати головним чином люмінесцентні лампи, у яких висока світлова віддача (до 75 лм/Вт і більше), тривалий термін служби (до 10000 год), мала яскравість світної поверхні, близький до природного спектральний состав випромінюваного світла, що забезпечує гарну передачу кольору. Разом з тим необхідно враховувати й недоліки цих ламп: висока пульсація світлового потоку, необхідність застосування спеціальної пускорегулюючої апаратури, складність їхньої утилізації через наявність у лампах парів ртуті.

Найбільш застосовними є люмінесцентні лампи білих кольорів і тепло-білого кольорів потужністю 20, 40 або 80 Вт. Лампи накалювання застосовують у приміщеннях, де виконуються роботи малої точності або здійснюють загальний нагляд за роботою устаткування.

Для виключення засвітлення екранів моніторів прямими світловими потоками світильники загального освітлення розташовую збоку від робочого місця, паралельно лінії зору оператора й стіні з вікнами.

Найважливішими умовами охорони праці на робочому місці й у місцях розташування комп’ютерної, звичайно є додержання правил електробезпеки. Бо саме через недотримання цих правил можуть виникнути неминучі наслідки.

Під час проектування систем електропостачання необхідно дотримуватись слідуючих правил:

* під час монтажу та експлуатації лінії електромережі необхідно повністю виключити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів. Для цього необхідно застосувати надійну, негорючу ізоляцію, автоматичне відключення;
* лінія електромережі для живлення ПК, іншого електрообладнання виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення електрообладнання;
* нульовий захисний дріт прокладається від розподільчого щита до розеток живлення;
* у приміщенні, де одночасно експлуатується декілька ПК або інша комп’ютерна техніка, на помітному місці встановлюється аварійний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення;
* ПК, інше електрообладнання повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення;
* штепсельні з'єднання й розетки крім контактів фазового та нульового робочого провідників повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним;
* неприпустимим є підключення ПК до звичайної двопровідної електромережі, у тому числі з використанням перехідних пристроїв;
* індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих пластинах;
* металеві сурми та гнучкі металеві рукави повинні бути заземлені;
* для підключення переносної електроапаратури застосовують гнучкі проводь у надійній ізоляції.

Тимчасова електропроводка від переносних приладів до джерел живлення виконується найкоротшим шляхом без заплутування проводів.

Виконуючи всі вище зазначені норми й правила охорони праці та навколишнього середовища при роботі з ПК та іншою комп’ютерною технікою, ви збережете своє здоров’я та працездатність.

# ВИСНОВКИ

Дипломний проєкт «Автоматизація обліку та аналізу продажів на оптовій базі» був реалізований з використанням мови програмування C# та бази даних Microsoft SQL Server. Ця система може бути використана для ефективного ведення обліку продажів, а також для аналізу даних про продажі.

В процесі розробки було враховано потреби користувачів та їх вимоги до функціональності системи. Було створено зручний інтерфейс користувача, що дозволяє просто та швидко виконувати необхідні дії. Для забезпечення безпеки даних було реалізовано систему авторизації та автентифікації користувачів.

Результатом роботи є повноцінний програмний продукт, який може бути використана на оптовій базі для ефективного ведення обліку та аналізу продажів. Розроблена програмний продукт може бути вдосконалений та розширена у майбутньому, залежно від потреб користувачів.

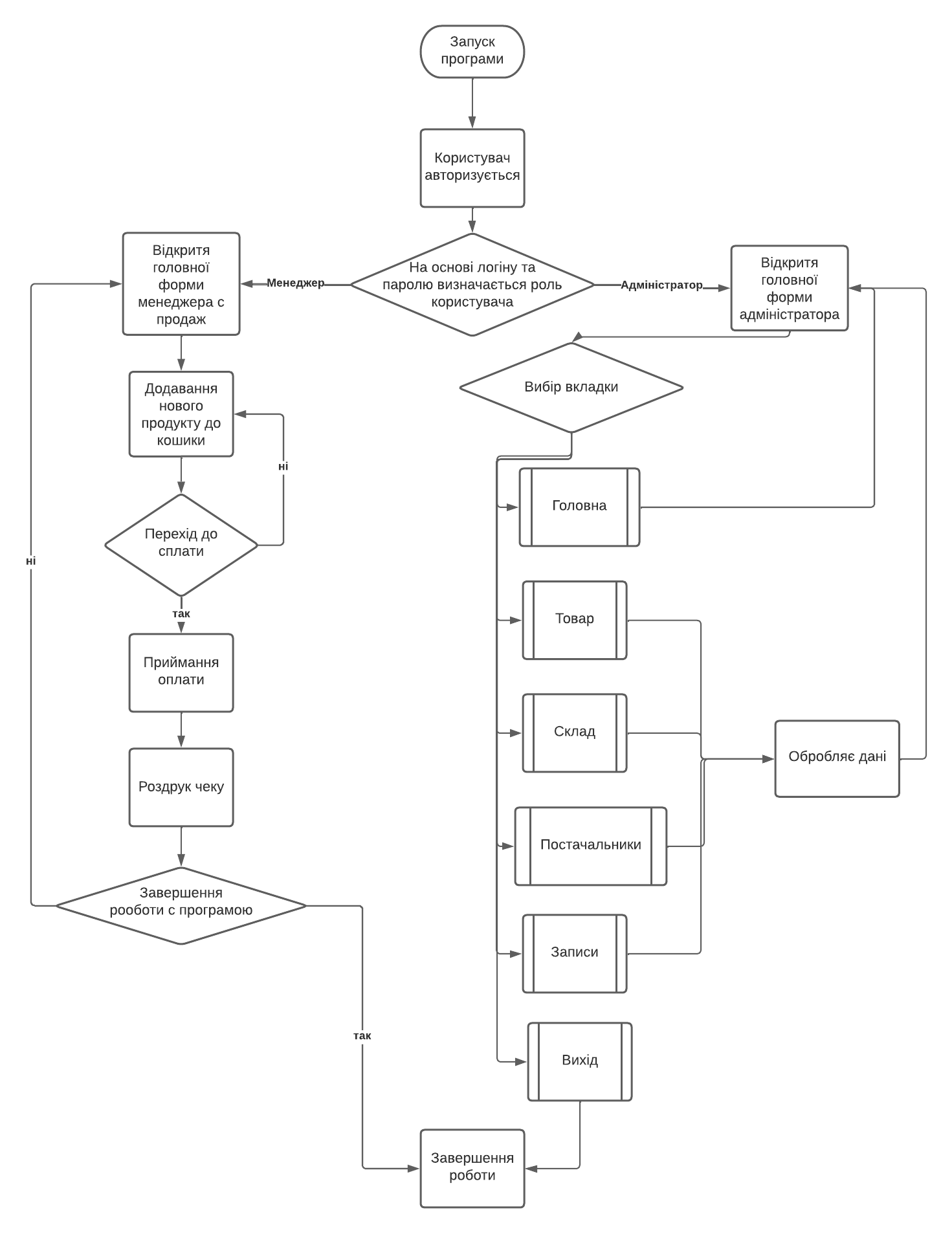
Отже, можна зробити висновок, що розробка даного програмного продукту є важливим кроком у забезпеченні ефективного ведення обліку продажів на оптовій базі.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

* + - 1. Шилдт Г. C#. Повне керівництво. - Київ: Видавництво "Діалектика", 2019. - 1200 с.
      2. Троелсен А., Джепикс Д. Програмування на C#. Ядро .NET Framework 4.5. - Львів: Видавництво "Старого Лева", 2017. - 832 с.
      3. Ріхтер Дж. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.0 на мові C#. - Харків: Видавництво "Сиция", 2018. - 1008 с.
      4. Розензвейг Б. Основи SQL. Вивчення мови запитів до баз даних. - Київ: Видавництво "Баланс Бізнес Букс", 2016. - 320 с.
      5. Торгашов А., Манні С. Розробка баз даних на SQL. - Львів: Видавництво "Країна Мрій", 2018. - 416 с.
      6. Сільверман Дж. SQL для професіоналів. - Київ: Видавництво "Діалектика", 2018. - 752 с.
      7. Методичні настанови (вказівки) щодо виконання дипломного проекту освітньо-кваліфікаційного рівня «молодший спеціаліст» для здобувачів вищої освіти спеціальності 121 «Інженерія програмного забезпечення», галузі знань 12 «Інформаційні технології» за тематикою «Розробка та створення прикладного програмного забезпечення» / уклад.: Арбузова Ю.В., Свириденко О.М. // Херсон: ХПТК ОНПУ, 2018. – 91 с.
      8. Методичні вказівки щодо виконання дипломного проекту за тематикою «Проектування автоматизованих інформаційних систем» для здобувачів освіти спеціальності 5.05010301 «Розробка програмного забезпечення». / Херсон. політехн. коледж ОНПУ: Н.І. Карлова, Ю.О. Арбузова– Херсон: ХПТК ОНПУ.– 2018.– 96с.

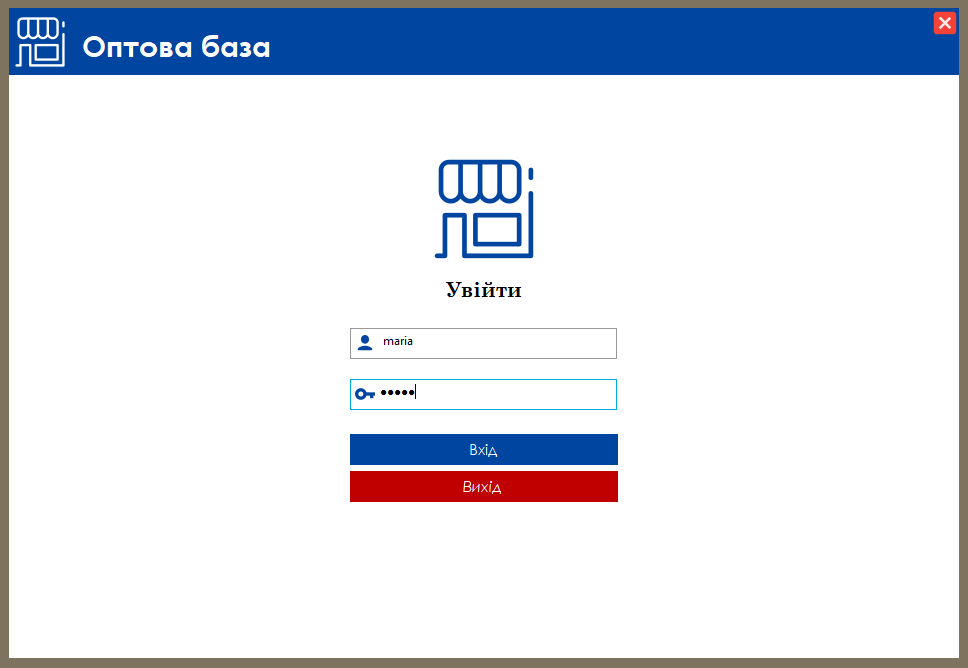
# ДОДАТОК А

Схема алгоритму програми

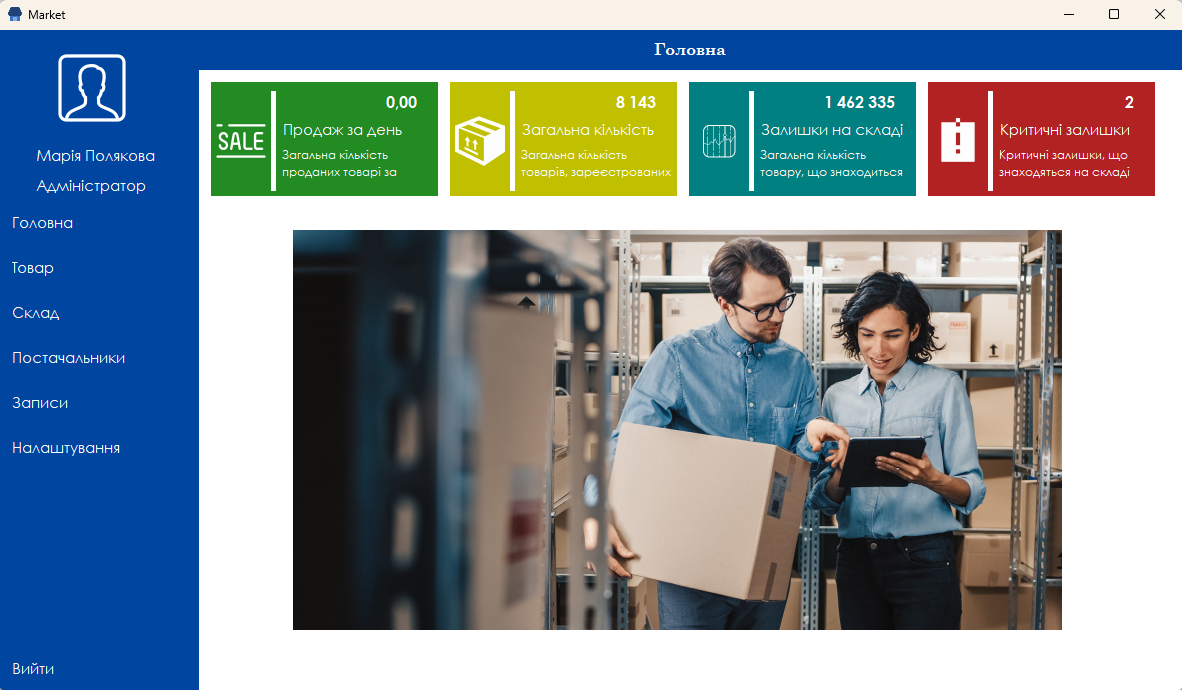
Додаток А.1 – Алгоритм роботи програми

# ДОДАТОК Б

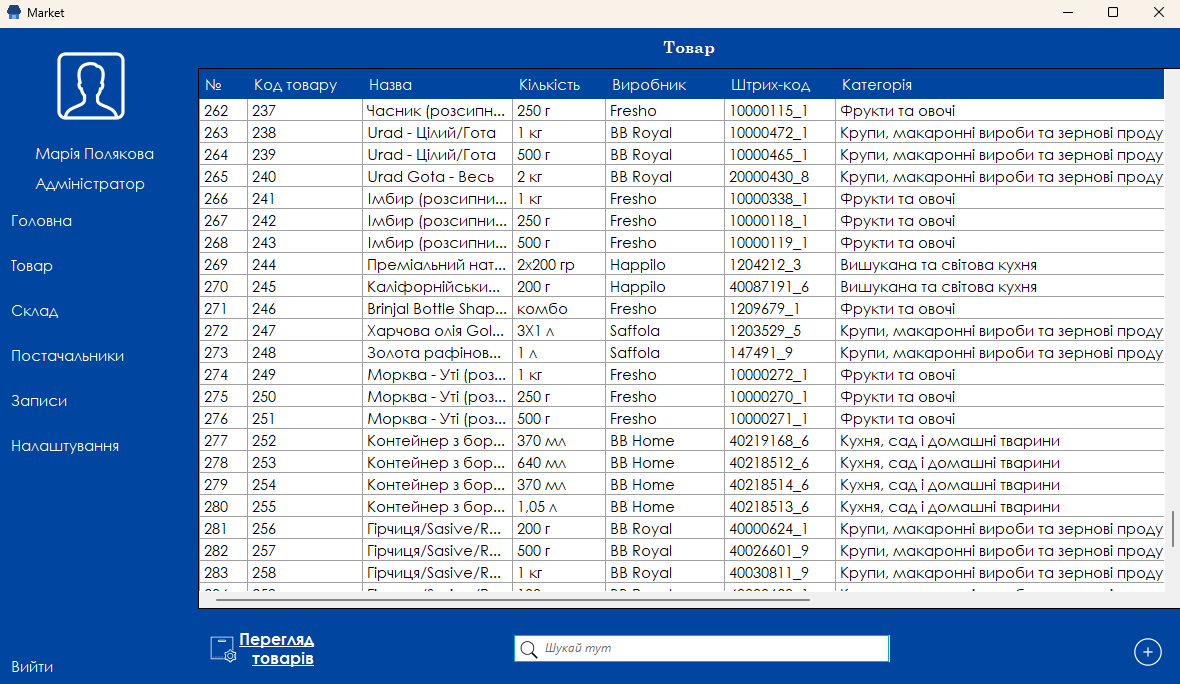
Друк результатів роботи програми на контрольному прикладі



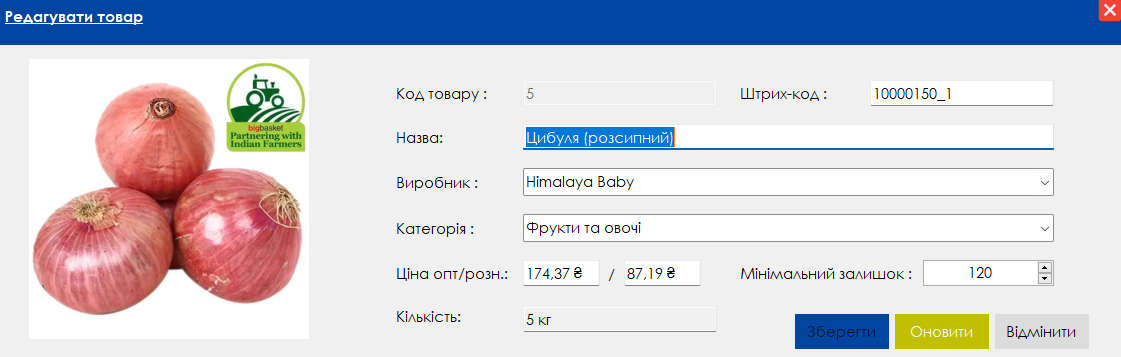
Додаток Б.1 – Форма авторизації користувача



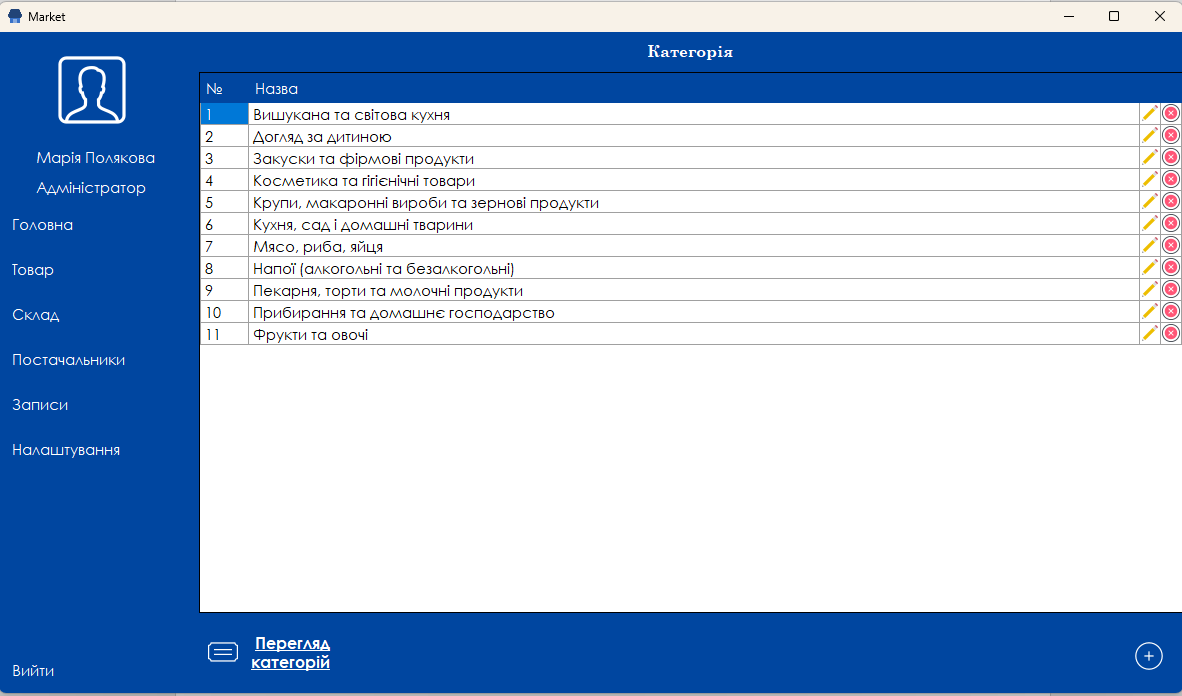
Додаток Б.2 – Головна форма для адміністратора



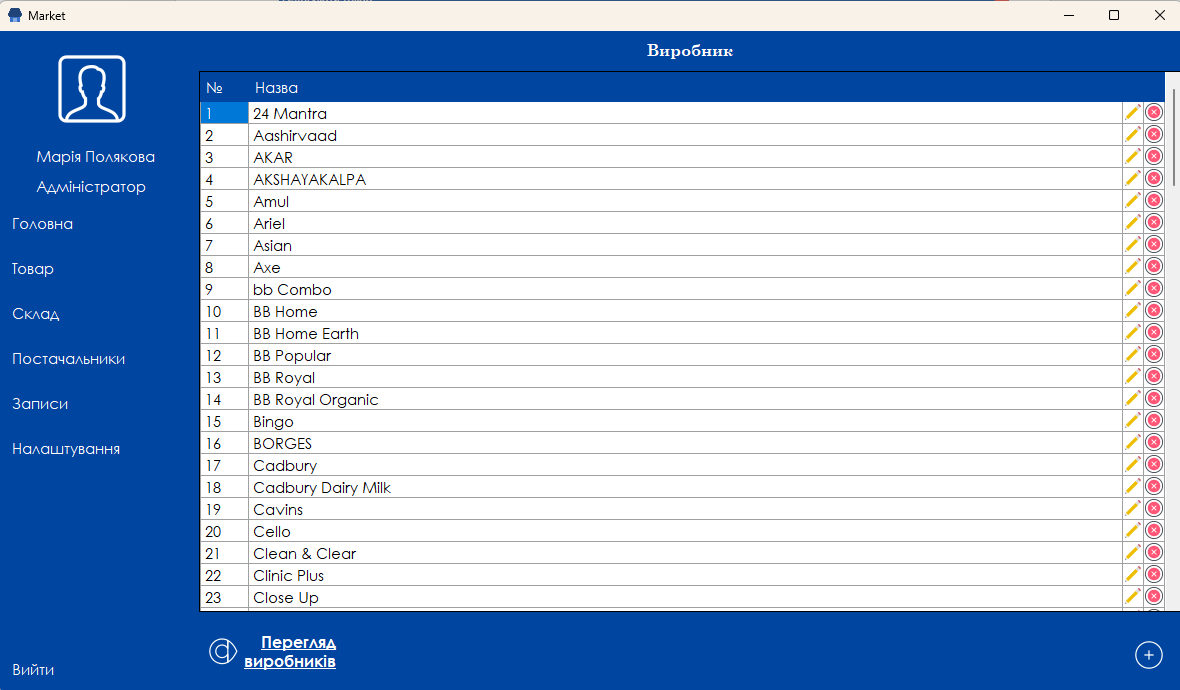
Додаток Б.3 – Форма перегляду списку товарів



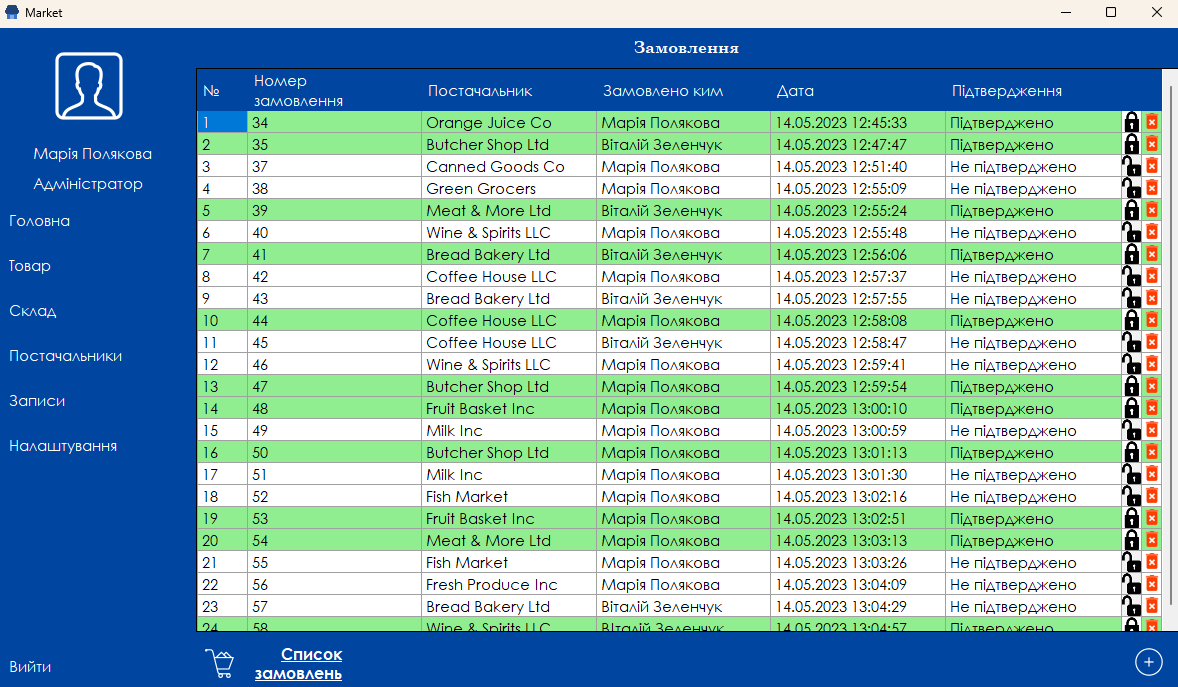
Додаток Б.4 – Форма перегляду інформації про товар



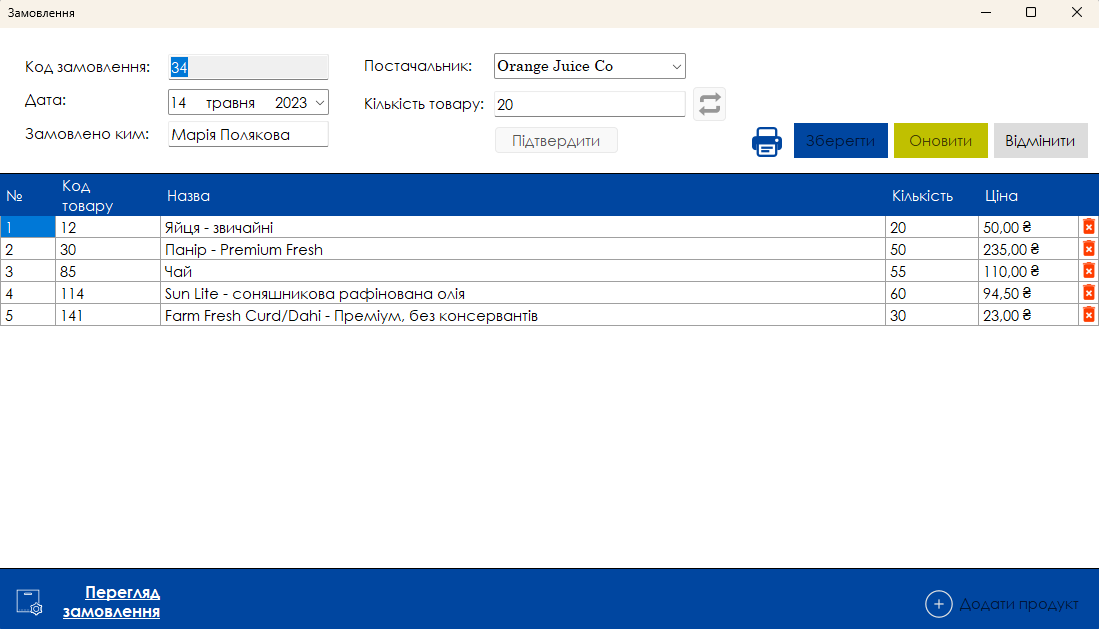
Додаток Б.5 – Список категорій



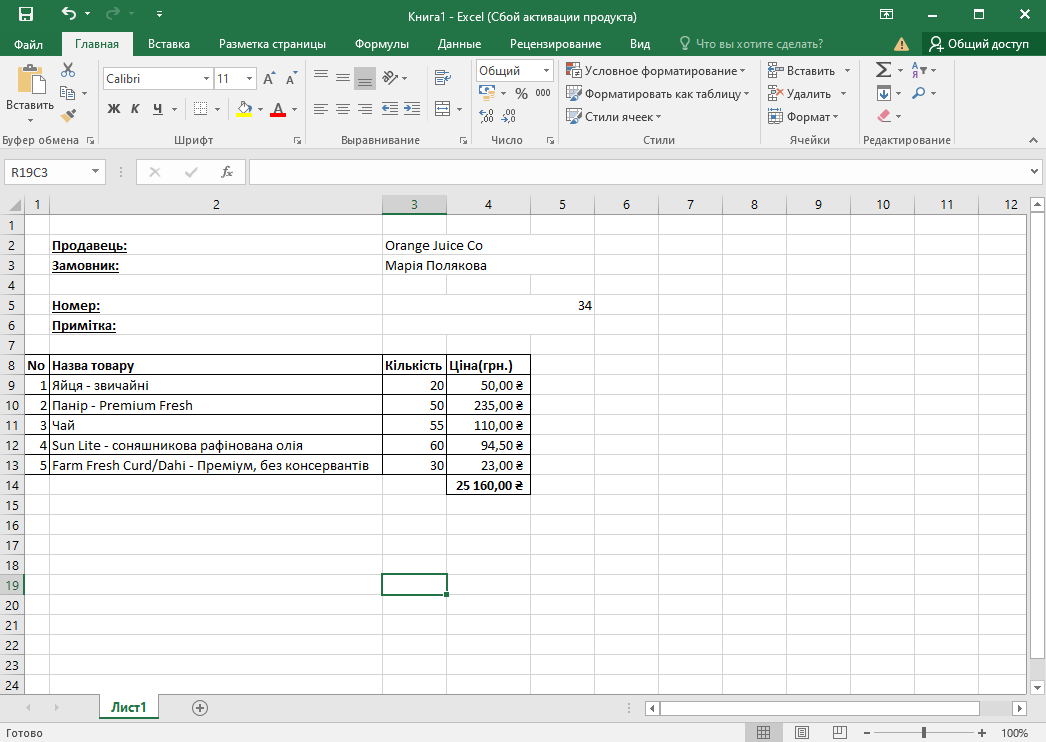
Додаток Б.6 – Список виробників



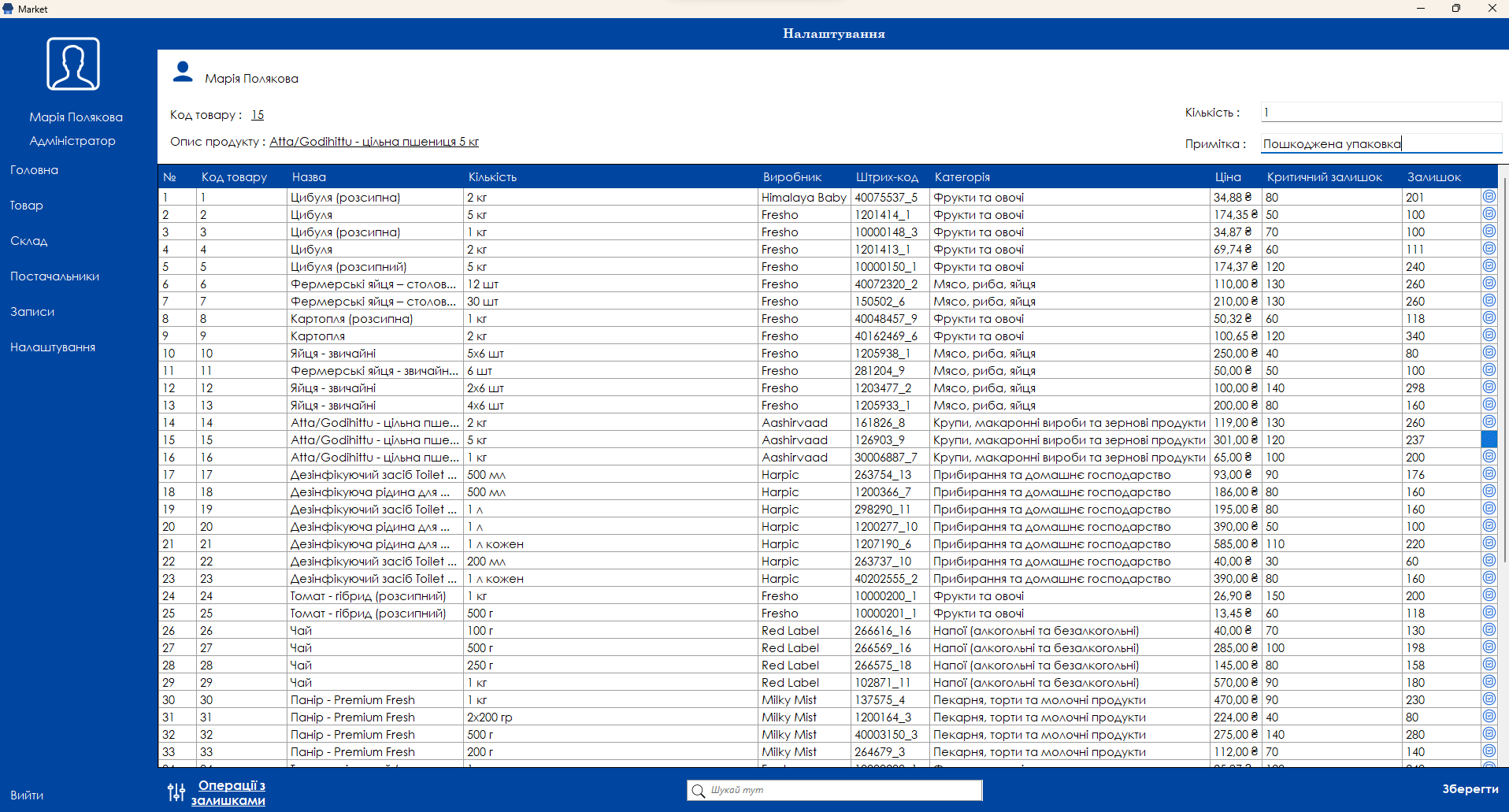
Додаток Б.7 – Список замовлень



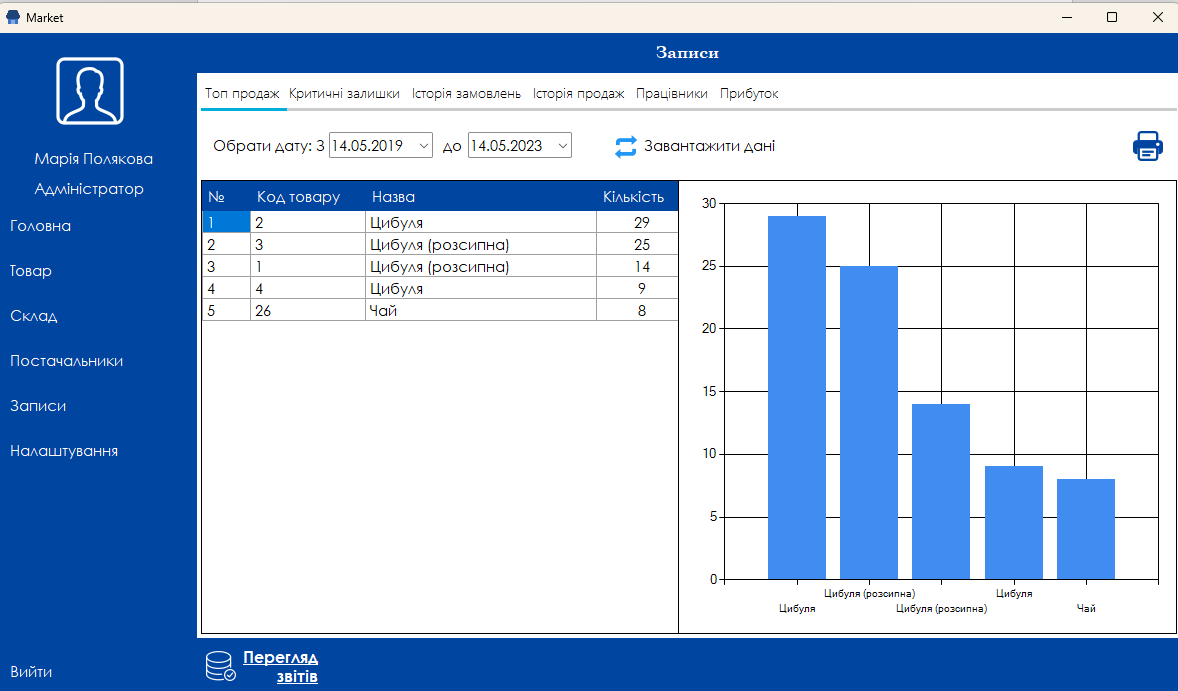
Додаток Б.8 – Деталі замовлення



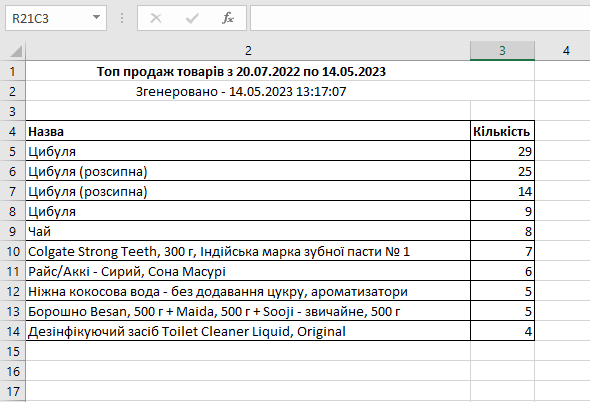
Додаток Б.9 – Товарна накладна замовлення



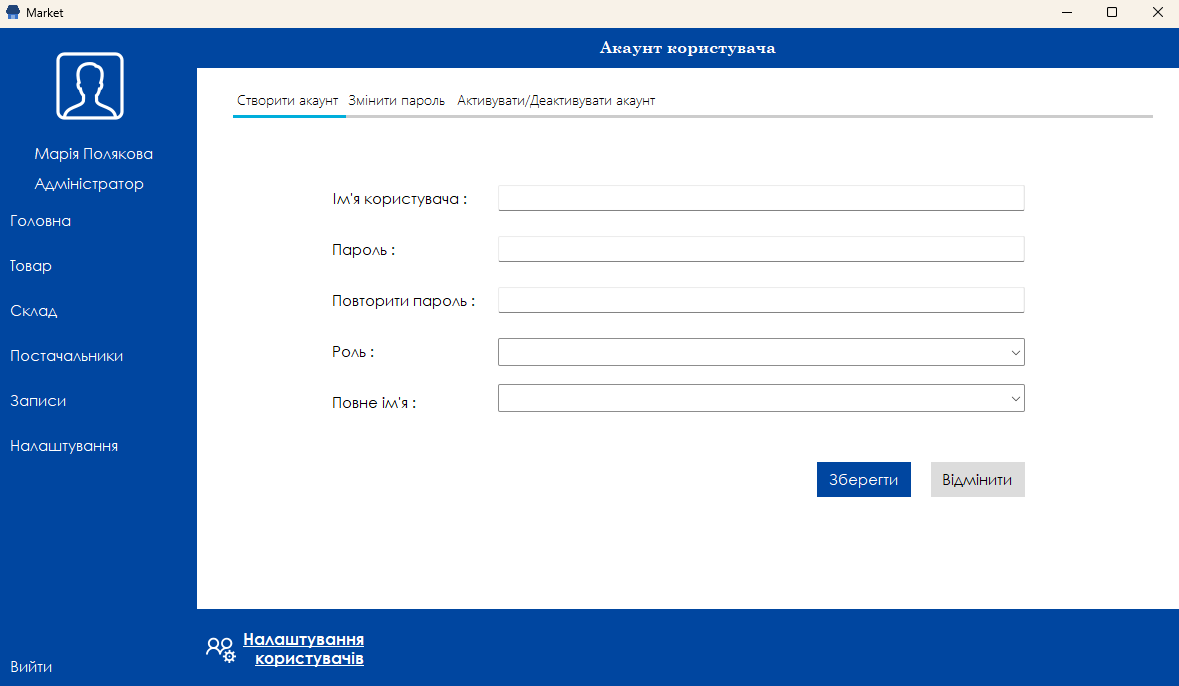
Додаток Б.10 – Списання продуктів



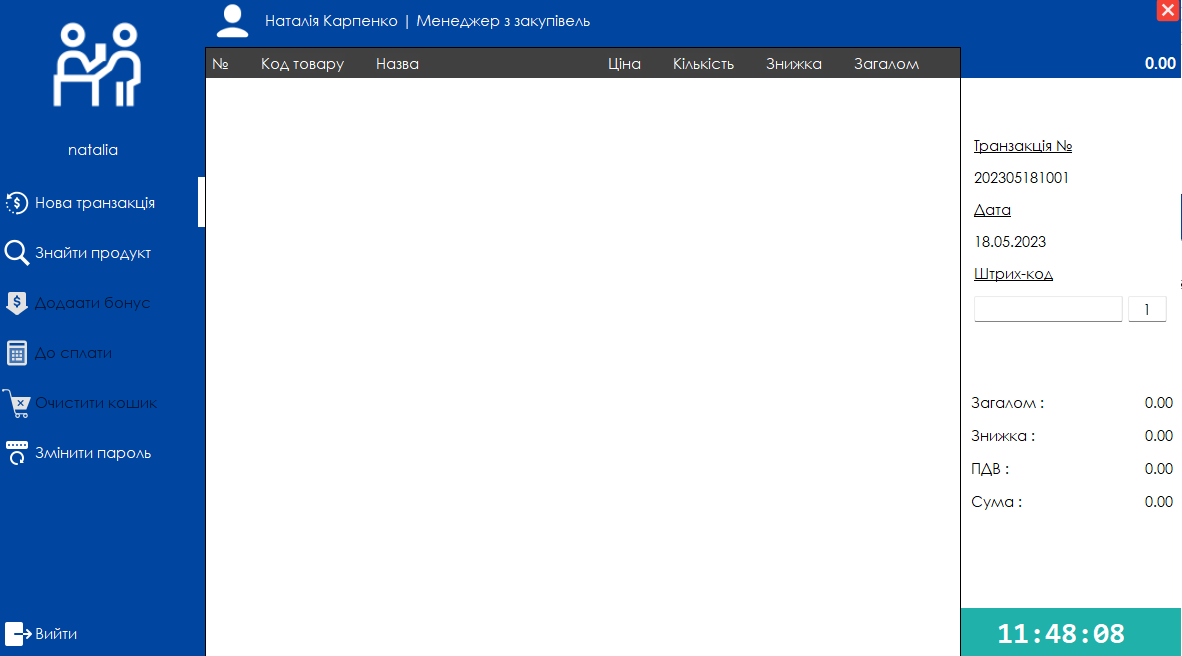
Додаток Б.11 – Форма зі звітами



Додаток Б.12 – Згенерований звіт в Excel



Додаток Б.13 – Форма налаштування користувача



Додаток Б.13 – Головна форма менеджера з продаж

# ДОДАТОК В

Значення коефіцієнта використання технологій програмування

та інструментів середовища розробки

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Розробка програмного продукту** | **Значення** | **Примітка** |
| 1 | ***Логічне проектування:*** |  |  |
|  | кількість та якість об’єктів | 0,2 |  |
|  | нормалізація даних | 0,2 |  |
|  | мережеві технології | 0,2 |  |
| 2 | ***Обробка даних:*** |  |  |
|  | Прості обчислення | 0,2 |  |
|  | Складні обчислення |  |  |
|  | Створення власних функцій |  |  |
| 3 | ***Наявність вихідних документів:*** |  |  |
|  | складні | 0,2 |  |
|  | структуровані | 0,1 |  |
| 4 | ***Наявність розвиненого інтерфейсу:*** |  |  |
|  | довідкова система, дружній інтерфейс | 0,2 |  |
|  | адміністрування та збереження даних | 0,2 |  |
|  | Всього (мін – 1, мах - 2) | 1,5 |  |

## Перелік інших частин

Студент /С.О. Тоцький /

Керівник проекту /В.В. Коленко/

АВТОМАТИЗАЦІЯ ОБЛІКУ ТА АНАЛІЗУ ПРОДАЖІВ НА ОПТОВІЙ БАЗІ

## Текст програми

ДП.ХПТК.00452 – 41 12 01

(компакт-диск)

Аркушів 35

2023

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace POSales

{

public partial class Adjustments : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

MainForm main;

int \_qty;

int StartIndex = 0;

int RowCount = 25;

int i = 0;

public Adjustments(MainForm mn)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

main = mn;

LoadStock();

lblUsername.Text = main.lblUsername.Text;

}

public void LoadStock()

{

i = 0;

StartIndex = 0;

dgvAdjustment.Rows.Clear();

try

{

cm = new SqlCommand("exec GetProductsList '" + txtSearch.Text + "', " + StartIndex.ToString() + ", 50" , cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvAdjustment.Rows.Add(i, dr[0].ToString(),

dr[1].ToString(),

dr[2].ToString(),

dr[3].ToString(),

dr[4].ToString(),

dr[5].ToString(),

dr[7].ToString(),

dr[8].ToString(),

dr[9].ToString()

);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dr.Close();

cn.Close();

}

}

private void dgvAdjustment\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

string colName = dgvAdjustment.Columns[e.ColumnIndex].Name;

if(colName=="Select")

{

lblPcode.Text = dgvAdjustment.Rows[e.RowIndex].Cells[1].Value.ToString();

lblDesc.Text = dgvAdjustment.Rows[e.RowIndex].Cells[2].Value.ToString() + " " + dgvAdjustment.Rows[e.RowIndex].Cells[3].Value.ToString();

\_qty = int.Parse(dgvAdjustment.Rows[e.RowIndex].Cells[8].Value.ToString());

btnSave.Enabled = true;

}

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

LoadStock();

}

public void Clear()

{

lblDesc.Text = "";

lblPcode.Text = "";

txtQty.Clear();

txtRemark.Clear();

}

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

{

try

{

//update stock

if(int.Parse(txtQty.Text)>\_qty)

{

MessageBox.Show("Не можна списати більше, ніж є в наявності.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

cm = new SqlCommand("INSERT INTO Adjustment(Adj\_Product, Adj\_Qty, Adj\_Remarks, Adj\_User) VALUES (@product, @qty, @remarks, @user)", cn);

cm.Parameters.AddWithValue("@product", lblPcode.Text);

cm.Parameters.AddWithValue("@qty", txtQty.Text);

cm.Parameters.AddWithValue("@remarks", txtRemark.Text);

cm.Parameters.AddWithValue("@user", lblUsername.Text);

cn.Open();

var res = cm.ExecuteNonQuery();

if (res > 0)

{

MessageBox.Show("Товар було успішно списано.", "Списання", MessageBoxButtons.OK, MessageBoxIcon.Information);

Clear();

btnSave.Enabled = false;

}

else

{

MessageBox.Show("Сталася помилка при списані.", "Списання", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

finally

{

cn.Close();

LoadStock();

}

}

private void dgvAdjustment\_Scroll(object sender, ScrollEventArgs e)

{

int displayedRows = dgvAdjustment.DisplayedRowCount(true);

int lastVisibleRowIndex = dgvAdjustment.FirstDisplayedScrollingRowIndex + displayedRows - 1;

if (lastVisibleRowIndex >= dgvAdjustment.RowCount - 1)

{

StartIndex += 25;

cm = new SqlCommand("exec GetProductsList '" + txtSearch.Text + "', " + StartIndex.ToString() + ", " + RowCount.ToString(), cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvAdjustment.Rows.Add(i, dr[0].ToString(),

dr[1].ToString(),

dr[2].ToString(),

dr[3].ToString(),

dr[4].ToString(),

dr[5].ToString(),

dr[6].ToString(),

dr[7].ToString(),

dr[8].ToString()

);

}

dr.Close();

cn.Close();

}

}

private void dgvAdjustment\_CellFormatting(object sender, DataGridViewCellFormattingEventArgs e)

{

if (e.ColumnIndex == 7) // Замените yourCurrencyColumnIndex на индекс столбца с валютой

{

if (e.Value != null)

{

decimal value = decimal.Parse(e.Value.ToString()); // Предполагается, что значение является числом типа decimal

e.Value = value.ToString("C"); // Форматирование значения с использованием знака валюты

}

}

}

}

}

public partial class Brand : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

public Brand()

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

LoadBrand();

}

//Data retrieve from tbBrand to dgvBrand on Brand form

public void LoadBrand()

{

int i = 0;

dgvBrand.Rows.Clear();

cn.Open();

cm = new SqlCommand("SELECT \* FROM Brand ORDER BY Br\_Title", cn);

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvBrand.Rows.Add(i, dr["Br\_Id"].ToString(), dr["Br\_Title"].ToString());

}

dr.Close();

cn.Close();

}

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

{

BrandModule moduleForm = new BrandModule(this);

moduleForm.btnUpdate.Enabled = false;

moduleForm.ShowDialog();

}

private void dgvBrand\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

//For update and delete brand by cell click from tbBrand

string colName = dgvBrand.Columns[e.ColumnIndex].Name;

if (colName == "Delete")

{

if (MessageBox.Show("Ви хочете видалити даного виробника?", "Видалити виробника", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cn.Open();

cm = new SqlCommand("DELETE FROM Brand WHERE Br\_Id = " + dgvBrand[1, e.RowIndex].Value.ToString(), cn);

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Виробник був успішно видалений.", "Market", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else if (colName == "Edit")

{

BrandModule brandModule = new BrandModule(this);

brandModule.lblId.Text = dgvBrand[1, e.RowIndex].Value.ToString();

brandModule.txtBrand.Text = dgvBrand[2, e.RowIndex].Value.ToString();

brandModule.btnSave.Enabled = false;

brandModule.btnUpdate.Enabled = true;

brandModule.Text = "Редагування виробника";

brandModule.ShowDialog();

}

LoadBrand();

}

}

}

public partial class BrandModule : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

Brand brand;

public BrandModule(Brand br)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

brand = br;

}

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

{

this.Dispose();

}

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

{

// To insert brand name to brand table

try

{

if (MessageBox.Show("Ви хочете додати нового виробника?", "Додати виробника", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cn.Open();

cm = new SqlCommand("INSERT INTO Brand(Br\_Title) VALUES(@brand)", cn);

cm.Parameters.AddWithValue("@brand", txtBrand.Text);

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Виробник був успішно доданий.", "POS");

Clear();

brand.LoadBrand();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

Clear();

}

public void Clear()

{

txtBrand.Clear();

btnUpdate.Enabled = false;

btnSave.Enabled = true;

txtBrand.Focus();

}

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

{

//Update brand name

if (MessageBox.Show("Ви хочете оновити дані про виробника?", "Оновити дані!", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cn.Open();

cm = new SqlCommand("UPDATE Brand SET Br\_Title = @brand WHERE Br\_id = " + lblId.Text, cn);

cm.Parameters.AddWithValue("@brand", txtBrand.Text);

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Дані про виробника були успішно оновлені.", "Market");

Clear();

this.Dispose();// To close this form after update data

}

}

}

}

public partial class Category : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

public Category()

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

LoadCategory();

}

public void LoadCategory()

{

int i = 0;

dgvCategory.Rows.Clear();

cn.Open();

cm = new SqlCommand("SELECT \* FROM Category ORDER BY Catg\_Title", cn);

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvCategory.Rows.Add(i, dr["Catg\_Id"].ToString(), dr["Catg\_Title"].ToString());

}

dr.Close();

cn.Close();

}

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

{

CategoryModule module = new CategoryModule(this);

module.ShowDialog();

}

private void dgvCategory\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

string colName = dgvCategory.Columns[e.ColumnIndex].Name;

if (colName == "Delete")

{

if (MessageBox.Show("Ви хочете видалити запис?", "Видалити запис", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cn.Open();

cm = new SqlCommand("DELETE FROM Category WHERE Catg\_Id = " + dgvCategory[1, e.RowIndex].Value.ToString(), cn);

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Запис було успішно видалено.", "Market", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else if (colName == "Edit")

{

CategoryModule module = new CategoryModule(this);

module.lblId.Text = dgvCategory[1, e.RowIndex].Value.ToString();

module.txtCategory.Text = dgvCategory[2, e.RowIndex].Value.ToString();

module.btnSave.Enabled = false;

module.btnUpdate.Enabled = true;

module.Text = "Редагування категорії";

module.ShowDialog();

}

LoadCategory();

}

}

}

public partial class CategoryModule : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

Category category;

public CategoryModule(Category ct)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

category = ct;

}

public void Clear()

{

txtCategory.Clear();

txtCategory.Focus();

btnSave.Enabled = true;

btnUpdate.Enabled = false;

}

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

{

try

{

if (MessageBox.Show("Ви хочете додати нову категорію?", "", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cn.Open();

cm = new SqlCommand("INSERT INTO Category(Catg\_Title) VALUES(@category)", cn);

cm.Parameters.AddWithValue("@category", txtCategory.Text);

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Категорія була успішно додана.", "Market");

Clear();

}

category.LoadCategory();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

Clear();

}

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

{

//Update brand name

if (MessageBox.Show("Ви хочете оновити категорію?", "Оновити категорію!", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cn.Open();

cm = new SqlCommand("UPDATE Category SET Catg\_Title = @category WHERE Catg\_Id = " + lblId.Text, cn);

cm.Parameters.AddWithValue("@category", txtCategory.Text);

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Категорія була успішно оновлена.", "Market");

Clear();

this.Dispose();// To close this form after update data

}

}

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

{

this.Dispose();

}

private void CategoryModule\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Dispose();

}

}

}

}

public partial class ChangePassword : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

Cashier cashier;

public ChangePassword(Cashier cash)

{

InitializeComponent();

cashier = cash;

lblUsername.Text = cashier.lblUsername.Text;

}

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

{

this.Dispose();

}

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

{

try

{

string oldpass = dbcon.getPassword(lblUsername.Text);

if(oldpass != txtPass.Text)

{

MessageBox.Show("Невірний пароль, спробуйте знову!", "Увага", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

else

{

txtPass.Visible = false;

btnNext.Visible = false;

txtNewPass.Visible = true;

txtComPass.Visible = true;

btnSave.Visible = true;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if(txtNewPass.Text != txtComPass.Text)

{

MessageBox.Show("Паролі не співпадають!", "Увага", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

if(MessageBox.Show("Змінити пароль?","Підтвердити",MessageBoxButtons.YesNo,MessageBoxIcon.Question)==DialogResult.Yes)

{

dbcon.ExecuteQuery("UPDATE [User] set Usr\_Password = '" + txtNewPass.Text + "' WHERE Usr\_Username = '" + lblUsername.Text + "'");

MessageBox.Show("Пароль було успішно оновлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Dispose();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка");

}

}

private void ChangePassword\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Dispose();

}

}

}

}

public partial class Client : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

public Client()

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

}

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

{

if (txtName.Text != "" || txtCard.Text != "")

{

try

{

cm = new SqlCommand("INSERT INTO Client (Cl\_Name, Cl\_CardNumber) VALUES(@name, @card)", cn);

cm.Parameters.AddWithValue("@name", txtName.Text);

cm.Parameters.AddWithValue("@card", txtCard.Text);

cn.Open();

var res = cm.ExecuteNonQuery();

if (res > 0)

{

MessageBox.Show("Клієнта було успішно зареєстровано!");

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

cn.Close();

}

}

else

{

MessageBox.Show("Введіть коректні дані!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

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

{

this.Close();

}

}

}

public partial class Dashboard : Form

{

SqlConnection cn = new SqlConnection();

DBConnect dbcon = new DBConnect();

public Dashboard()

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

}

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

{

string sdate = DateTime.Now.ToString("yyyy-MM-dd");

lblDalySale.Text = dbcon.ExtractData("SELECT ISNULL(SUM(total),0) AS total FROM tbCart WHERE status LIKE 'Sold' AND sdate BETWEEN '"+sdate+ "' AND '" + sdate + "'").ToString("#,##0.00");

lblTotalProduct.Text = dbcon.ExtractData("SELECT COUNT(\*) FROM Product").ToString("#,##0");

lblStockOnHand.Text = dbcon.ExtractData("SELECT ISNULL(SUM(OnStk\_Quantity), 0) AS qty FROM OnStock").ToString("#,##0");

lblCriticalItems.Text = dbcon.ExtractData("SELECT COUNT(\*) FROM Product INNER JOIN OnStock ON Pr\_Id = OnStk\_Product WHERE OnStk\_Quantity < Pr\_Reorder").ToString("#,##0");

}

}

}

public partial class Discount : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

string stitle = "Market";

Cashier cashier;

public Discount(Cashier cash)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

cashier = cash;

this.KeyPreview = true;

}

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

{

this.Dispose();

}

private void Discount\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape) this.Dispose();

else if (e.KeyCode == Keys.Enter) btnSave.PerformClick();

}

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

{

try

{

if (MessageBox.Show("Додати знижку?", stitle, MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cn.Open();

cm = new SqlCommand("SELECT \* FROM Client WHERE Cl\_CardNumber = @card", cn);

cm.Parameters.AddWithValue("@card", txtCardNum.Text.Trim());

var res = cm.ExecuteReader();

if (res.HasRows)

{

cn.Close();

cn.Open();

cm = new SqlCommand("UPDATE Cart SET Crt\_Disc\_percent = 5 WHERE Crt\_Transno = @transno", cn);

cm.Parameters.AddWithValue("@transno", cashier.lblTranNo.Text);

cm.ExecuteNonQuery();

cashier.LoadCart();

}

else

{

MessageBox.Show("Не знайдено клієнта с даним номером картки!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, stitle);

}

finally

{

cn.Close();

this.Dispose();

}

}

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

{

Client client = new Client();

client.Show();

}

}

}

public partial class Login : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

public string \_pass = "";

public bool \_isactivate;

public Login()

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

txtName.Focus();

}

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

{

if (MessageBox.Show("Вийти з програми?", "Вийти", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Application.Exit();

}

}

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

{

string \_username = "", \_name = "", \_role = "";

try

{

bool found;

cn.Open();

cm = new SqlCommand("exec GetUserInfo @username, @password", cn);

cm.Parameters.AddWithValue("@username", txtName.Text);

cm.Parameters.AddWithValue("@password", txtPass.Text);

dr = cm.ExecuteReader();

dr.Read();

if(dr.HasRows)

{

found = true;

\_username = dr["Usr\_Username"].ToString();

\_name = dr["Usr\_Name"].ToString();

\_role = dr["Ur\_Title"].ToString();

\_pass = dr["Usr\_Password"].ToString();

\_isactivate = bool.Parse(dr["Usr\_Isactivate"].ToString());

}

else

{

found = false;

}

dr.Close();

cn.Close();

if(found)

{

if(!\_isactivate)

{

MessageBox.Show("Аккаунт деактивовано адміністратором. Неможливо ввійти", "Аккаунт деактивовано", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if(\_role== "Менеджер з закупівель")

{

txtName.Clear();

txtPass.Clear();

Cashier cashier = new Cashier();

cashier.lblUsername.Text = \_username;

cashier.lblname.Text = \_name + " | " + \_role;

cashier.Show();

this.Hide();

}

else

{

txtName.Clear();

txtPass.Clear();

MainForm main = new MainForm();

main.lblUsername.Text = \_name;

main.lblName.Text = \_name;

main.\_pass = \_pass;

main.Show();

this.Hide();

}

}

else

{

MessageBox.Show("Неправильне ім'я користувача або пароль", "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception ex)

{

cn.Close();

MessageBox.Show(ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

if (MessageBox.Show("Вийти з програми?", "Вийти", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

Application.Exit();

}

}

private void txtPass\_KeyPress(object sender, KeyPressEventArgs e)

{

if(e.KeyChar==13)

{

btnLogin.PerformClick();

}

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

cn.Open();

cm = new SqlCommand("exec GetReceiptInfo @transno", cn);

cm.Parameters.AddWithValue("@transno", "202303131008");

dr = cm.ExecuteReader();

Report report = new Report();

report.GenerateReceipt(dr);

}

}

}

public partial class LookUpProduct : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

Cashier cashier;

int StartIndex = 0;

int RowCount = 25;

int i = 0;

public LookUpProduct(Cashier cash)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

cashier = cash;

LoadProduct();

}

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

{

this.Dispose();

}

public void LoadProduct()

{

i = 0;

StartIndex = 0;

dgvProduct.Rows.Clear();

try

{

cm = new SqlCommand("exec GetProductsList '" + txtSearch.Text + "', " + StartIndex.ToString() + ", " + RowCount.ToString(), cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvProduct.Rows.Add(i, dr[0].ToString(),

dr[4].ToString(),

dr[1].ToString(),

dr[2].ToString(),

dr[3].ToString(),

dr[5].ToString(),

dr[6].ToString(),

dr[9].ToString()

);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dr.Close();

cn.Close();

}

}

private void dgvProduct\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

string colName = dgvProduct.Columns[e.ColumnIndex].Name;

if (colName == "Select")

{

Qty qty = new Qty(cashier);

qty.ProductDetails(dgvProduct.Rows[e.RowIndex].Cells[1].Value.ToString(), double.Parse(dgvProduct.Rows[e.RowIndex].Cells[7].Value.ToString()), cashier.lblTranNo.Text, int.Parse(dgvProduct.Rows[e.RowIndex].Cells[8].Value.ToString()));

qty.ShowDialog();

}

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

LoadProduct();

}

private void LookUpProduct\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Dispose();

}

}

private void dgvProduct\_Scroll(object sender, ScrollEventArgs e)

{

int displayedRows = dgvProduct.DisplayedRowCount(true);

int lastVisibleRowIndex = dgvProduct.FirstDisplayedScrollingRowIndex + displayedRows - 1;

try

{

if (lastVisibleRowIndex >= dgvProduct.RowCount - 1)

{

StartIndex += 25;

cm = new SqlCommand("exec GetProductsList '" + txtSearch.Text + "', " + StartIndex.ToString() + ", " + RowCount.ToString(), cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvProduct.Rows.Add(i, dr[0].ToString(),

dr[4].ToString(),

dr[1].ToString(),

dr[2].ToString(),

dr[3].ToString(),

dr[5].ToString(),

dr[6].ToString(),

dr[9].ToString()

);

}

}

}

catch( Exception ex )

{

MessageBox.Show(ex.Message);

}

finally

{

dr.Close();

cn.Close();

}

}

private void dgvProduct\_CellFormatting(object sender, DataGridViewCellFormattingEventArgs e)

{

if (e.ColumnIndex == 7) // Замените yourCurrencyColumnIndex на индекс столбца с валютой

{

if (e.Value != null)

{

decimal value = decimal.Parse(e.Value.ToString()); // Предполагается, что значение является числом типа decimal

e.Value = value.ToString("C"); // Форматирование значения с использованием знака валюты

}

}

}

}

}

public partial class MainForm : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

public string \_pass;

public MainForm()

{

InitializeComponent();

customizeDesing();

cn = new SqlConnection(dbcon.myConnection());

}

#region panelSlide

private void customizeDesing()

{

panelSubProduct.Visible = false;

panelSubRecord.Visible = false;

panelSubStock.Visible = false;

panelSubSetting.Visible = false;

}

private void hideSubmenu()

{

if (panelSubProduct.Visible == true)

panelSubProduct.Visible = false;

if (panelSubRecord.Visible == true)

panelSubRecord.Visible = false;

if (panelSubSetting.Visible == true)

panelSubSetting.Visible = false;

if (panelSubStock.Visible == true)

panelSubStock.Visible = false;

}

private void showSubmenu(Panel submenu)

{

if (submenu.Visible == false)

{

hideSubmenu();

submenu.Visible = true;

}

else

submenu.Visible = false;

}

#endregion panelSlide

private Form activeForm = null;

public void openChildForm(Form childForm)

{

if (activeForm != null)

activeForm.Close();

activeForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

lblTitle.Text = childForm.Text;

panelMain.Controls.Add(childForm);

panelMain.Tag = childForm;

childForm.BringToFront();

childForm.Show();

}

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

{

openChildForm(new Dashboard());

hideSubmenu();

}

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

{

showSubmenu(panelSubProduct);

}

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

{

openChildForm(new Product());

hideSubmenu();

}

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

{

openChildForm(new Category());

hideSubmenu();

}

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

{

openChildForm(new Brand());

hideSubmenu();

}

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

{

showSubmenu(panelSubStock);

}

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

{

openChildForm(new StockIn(this));

hideSubmenu();

}

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

{

openChildForm(new Adjustments(this));

hideSubmenu();

}

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

{

openChildForm(new Supplier());

hideSubmenu();

}

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

{

showSubmenu(panelSubRecord);

}

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

{

openChildForm(new DailySale(this));

hideSubmenu();

}

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

{

openChildForm(new Record());

hideSubmenu();

}

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

{

showSubmenu(panelSubSetting);

}

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

{

openChildForm(new UserAccount(this));

hideSubmenu();

}

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

{

hideSubmenu();

Store store = new Store();

store.ShowDialog();

}

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

{

hideSubmenu();

if (MessageBox.Show("Вийти з програми?", "Вийти", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

this.Hide();

Login login = new Login();

login.ShowDialog();

}

}

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

{

btnDashboard.PerformClick();

//Noti();

}

// Noti Alart for critical items

public void Noti()

{

int i = 0;

cn.Open();

cm = new SqlCommand("SELECT \* FROM vwCriticalItems", cn);

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

Alert alert = new Alert(this);

alert.lblPcode.Text = dr["pcode"].ToString();

alert.btnReorder.Enabled = true;

alert.showAlert(i + ". " + dr["pdesc"].ToString() + " - " + dr["qty"].ToString());

}

dr.Close();

cn.Close();

}

private void MainForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

public partial class Product : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

int StartIndex = 0;

int RowCount = 25;

int i = 0;

public Product()

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

LoadProduct();

}

public void LoadProduct()

{

i = 0;

StartIndex = 0;

dgvProduct.Rows.Clear();

cm = new SqlCommand("exec GetProductsList '" + txtSearch.Text+ "', " + StartIndex.ToString() + ", " + 50,cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvProduct.Rows.Add(i, dr[0].ToString(),

dr[1].ToString(),

dr[2].ToString(),

dr[3].ToString(),

dr[4].ToString(),

dr[5].ToString(),

dr[6].ToString(),

dr[7].ToString(),

dr[8].ToString(),

dr[9].ToString()

);

}

dr.Close();

cn.Close();

}

private void dgvProduct\_Scroll(object sender, ScrollEventArgs e)

{

int displayedRows = dgvProduct.DisplayedRowCount(true);

int lastVisibleRowIndex = dgvProduct.FirstDisplayedScrollingRowIndex + displayedRows - 1;

if (lastVisibleRowIndex >= dgvProduct.RowCount - 1)

{

StartIndex += 25;

cm = new SqlCommand("exec GetProductsList '" + txtSearch.Text + "', " + StartIndex.ToString() + ", " + RowCount.ToString(), cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvProduct.Rows.Add(i, dr[0].ToString(),

dr[1].ToString(),

dr[2].ToString(),

dr[3].ToString(),

dr[4].ToString(),

dr[5].ToString(),

dr[6].ToString(),

dr[7].ToString(),

dr[8].ToString(),

dr[9].ToString()

);

}

dr.Close();

cn.Close();

}

}

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

{

ProductModule productModule = new ProductModule();

productModule.ShowDialog();

}

private void dgvProduct\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

string colName = dgvProduct.Columns[e.ColumnIndex].Name;

if (colName == "Edit")

{

ProductModule product = new ProductModule();

//product.txtPcode.Text = dgvProduct.Rows[e.RowIndex].Cells[1].Value.ToString();

//product.txtBarcode.Text = dgvProduct.Rows[e.RowIndex].Cells[2].Value.ToString();

//product.txtPdesc.Text = dgvProduct.Rows[e.RowIndex].Cells[3].Value.ToString();

//product.cboBrand.Text = dgvProduct.Rows[e.RowIndex].Cells[4].Value.ToString();

//product.cboCategory.Text = dgvProduct.Rows[e.RowIndex].Cells[5].Value.ToString();

//product.txtPrice.Text = dgvProduct.Rows[e.RowIndex].Cells[6].Value.ToString();

//product.UDReOrder.Value = int.Parse(dgvProduct.Rows[e.RowIndex].Cells[7].Value.ToString());

product.LoadProduct(dgvProduct.Rows[e.RowIndex].Cells[1].Value.ToString());

product.txtPcode.Enabled = false;

product.btnSave.Enabled = false;

product.btnUpdate.Enabled = true;

product.label1.Text = "Редагувати товар";

product.ShowDialog();

}

else if (colName == "Delete")

{

if (MessageBox.Show("Видалити обраний товар", "Видалити", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cn.Open();

cm = new SqlCommand("DELETE FROM Product WHERE Pr\_id = " + dgvProduct[1, e.RowIndex].Value.ToString(), cn);

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Товар було успішно видалено .", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

LoadProduct();

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

LoadProduct();

}

private void dgvProduct\_CellFormatting(object sender, DataGridViewCellFormattingEventArgs e)

{

if (e.ColumnIndex == 7 || e.ColumnIndex == 8) // Замените yourCurrencyColumnIndex на индекс столбца с валютой

{

if (e.Value != null)

{

decimal value = decimal.Parse(e.Value.ToString()); // Предполагается, что значение является числом типа decimal

e.Value = value.ToString("C"); // Форматирование значения с использованием знака валюты

}

}

}

}

}

public partial class ProductModule : Form

{

SqlConnection cn;

SqlCommand cm;

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

string stitle = "Market";

Product product;

public ProductModule()

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

//product = pd;

LoadBrand();

LoadCategory();

}

public void LoadCategory()

{

cboCategory.Items.Clear();

cboCategory.DataSource = dbcon.getTable("SELECT \* FROM Category");

cboCategory.DisplayMember = "Catg\_Title";

cboCategory.ValueMember = "Catg\_Id";

}

public void LoadBrand()

{

cboBrand.Items.Clear();

cboBrand.DataSource = dbcon.getTable("SELECT \* FROM Brand");

cboBrand.DisplayMember = "Br\_Title";

cboBrand.ValueMember = "Br\_Id";

}

public void LoadProduct(string id)

{

try

{

cm = new SqlCommand("SELECT \* FROM Product WHERE Pr\_Id = '" + id + "'", cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

txtPcode.Text = dr[0].ToString();

txtBarcode.Text = dr[1].ToString();

txtPrName.Text = dr[10].ToString();

txtPriceOpt.Text = decimal.Parse(dr[6].ToString()).ToString("C");

txtPriceRozn.Text = decimal.Parse(dr[5].ToString()).ToString("C");

txtQuantity.Text = dr[7].ToString();

UDReOrder.Value = int.Parse(dr[8].ToString());

imgProduct.Load(dr[11].ToString());

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

cn.Close();

dr.Close();

}

}

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

{

this.Dispose();

}

public void Clear()

{

txtPcode.Clear();

txtBarcode.Clear();

txtPrName.Clear();

txtPriceOpt.Clear();

cboBrand.SelectedIndex = 0;

cboCategory.SelectedIndex = 0;

UDReOrder.Value = 1;

//txtPcode.Enabled = true;

//txtPcode.Focus();

btnSave.Enabled = true;

btnUpdate.Enabled = false;

}

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

{

try

{

if (MessageBox.Show("Ви хочете додати новий продукт?", "Додати", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cm = new SqlCommand("INSERT INTO Product(Pr\_Barcode, Pr\_Title, Pr\_Brand, Pr\_Category, Pr\_Price, Pr\_PriceOpt, Pr\_Reorder)VALUES (@Pr\_Barcode,@Pr\_Title,@Pr\_Brand,@Pr\_Category,@Pr\_Price, @Pr\_PriceOpt, @Pr\_Reorder)", cn);

cm.Parameters.AddWithValue("@Pr\_Barcode", txtBarcode.Text);

cm.Parameters.AddWithValue("@Pr\_Title", txtPrName.Text);

cm.Parameters.AddWithValue("@Pr\_Brand", cboBrand.SelectedValue);

cm.Parameters.AddWithValue("@Pr\_Category", cboCategory.SelectedValue);

cm.Parameters.AddWithValue("@Pr\_PriceOpt", double.Parse(txtPriceOpt.Text));

cm.Parameters.AddWithValue("@Pr\_Price", double.Parse(txtPriceRozn.Text));

cm.Parameters.AddWithValue("@Pr\_Reorder", UDReOrder.Value);

cn.Open();

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Продукт було успішно додано.", stitle);

Clear();

//product.LoadProduct();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

Clear();

}

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

{

try

{

if (MessageBox.Show("Ви хочете оновити дані про продукт?", "Оновити дані", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cm = new SqlCommand("UPDATE Product SET Pr\_Barcode=@barcode,Pr\_Title=@Pr\_Title,Pr\_Brand=@bid,PR\_Category=@cid,Pr\_Price=@price, Pr\_PriceOpt=@priceOpt, Pr\_Reorder=@reorder WHERE Pr\_Id = @Pr\_id", cn);

cm.Parameters.AddWithValue("@Pr\_Id", txtPcode.Text);

cm.Parameters.AddWithValue("@barcode", txtBarcode.Text);

cm.Parameters.AddWithValue("@Pr\_Title", txtPrName.Text);

cm.Parameters.AddWithValue("@bid", cboBrand.SelectedValue);

cm.Parameters.AddWithValue("@cid", cboCategory.SelectedValue);

cm.Parameters.AddWithValue("@priceOpt", double.Parse(txtPriceOpt.Text));

cm.Parameters.AddWithValue("@price", double.Parse(txtPriceRozn.Text));

cm.Parameters.AddWithValue("@reorder", UDReOrder.Value);

cn.Open();

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Дані про продукт були успішно оновлені.", stitle);

Clear();

this.Dispose();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ProductModule\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Dispose();

}

}

}

}

public partial class ProductStockIn : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

string stitle = "Market";

StockInModule stockIn;

int StartIndex = 0;

int RowCount = 25;

int i = 0;

public ProductStockIn(StockInModule stk)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

stockIn = stk;

LoadProduct();

}

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

{

stockIn.LoadDoc(stockIn.textBox1.Text);

stockIn.Select();

this.Dispose();

}

public void LoadProduct()

{

i = 0;

StartIndex = 0;

dgvProduct.Rows.Clear();

try

{

cm = new SqlCommand("exec GetProductsListWithCrit '" + txtSearch.Text + "', " + StartIndex.ToString() + ", " + RowCount.ToString(), cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvProduct.Rows.Add(i, dr[0].ToString(),

dr[1].ToString(),

dr[9].ToString(),

dr[10].ToString()

);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dr.Close();

cn.Close();

}

}

private void dgvProduct\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

string colName = dgvProduct.Columns[e.ColumnIndex].Name;

if (colName == "Select")

{

if (MessageBox.Show("Додати цей товар?", stitle, MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

addStockIn(dgvProduct.Rows[e.RowIndex].Cells[1].Value.ToString());

}

}

}

public void addStockIn(string productId)

{

try

{

cn.Open();

cm = new SqlCommand("INSERT INTO StockInProduct (Sip\_Doc, Sip\_Product) VALUES (@id, @productId)", cn);

cm.Parameters.AddWithValue("@id", stockIn.textBox1.Text);

cm.Parameters.AddWithValue("@productId", productId);

if (cm.ExecuteNonQuery() > 0)

{

MessageBox.Show("Товар успішно додано", stitle, MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Сталася помилка при додавані товару", stitle, MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, stitle);

}

finally

{

cn.Close();

}

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

LoadProduct();

}

private void ProductStockIn\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Dispose();

}

}

private void dgvProduct\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

ProductModule product = new ProductModule();

product.LoadProduct(dgvProduct.Rows[e.RowIndex].Cells[1].Value.ToString());

product.label1.Text = "Інформація про товар";

product.txtPcode.Enabled = false;

product.btnSave.Enabled = false;

product.btnUpdate.Enabled = true;

product.ShowDialog();

}

private void dgvProduct\_Scroll(object sender, ScrollEventArgs e)

{

int displayedRows = dgvProduct.DisplayedRowCount(true);

int lastVisibleRowIndex = dgvProduct.FirstDisplayedScrollingRowIndex + displayedRows - 1;

if (lastVisibleRowIndex >= dgvProduct.RowCount - 1)

{

StartIndex += 25;

try

{

cm = new SqlCommand("exec GetProductsList '" + txtSearch.Text + "', " + StartIndex.ToString() + ", " + RowCount.ToString(), cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvProduct.Rows.Add(i, dr[0].ToString(),

dr[1].ToString(),

dr[9].ToString(),

"False"

);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dr.Close();

cn.Close();

}

}

}

private void dgvProduct\_CellPainting(object sender, DataGridViewCellPaintingEventArgs e)

{

if (e.RowIndex > -1 && dgvProduct.Rows[e.RowIndex].Cells[4].Value.ToString() == "True")

{

e.CellStyle.BackColor = Color.Red;

}

}

}

}

public partial class Qty : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

string stitle = "Point Of Sales";

private string pcode;

private double price;

private String transno;

private int qty;

Cashier cashier;

public Qty(Cashier cash)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

cashier = cash;

}

public void ProductDetails(string pcode, double price, string transno, int qty)

{

this.pcode = pcode;

this.price = price;

this.transno = transno;

this.qty = qty;

}

private void txtQty\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar == 13) && (txtQty.Text != string.Empty))

{

try

{

string id = "";

int cart\_qty = 0;

bool found = false;

cn.Open();

cm = new SqlCommand("Select \* from Cart Where Crt\_Transno = @transno and Crt\_Product = @pcode", cn);

cm.Parameters.AddWithValue("@transno", transno);

cm.Parameters.AddWithValue("@pcode", pcode);

dr = cm.ExecuteReader();

dr.Read();

if (dr.HasRows)

{

id = dr["Crt\_Id"].ToString();

cart\_qty = int.Parse(dr["Crt\_Qty"].ToString());

found = true;

}

else

{

found = false;

}

dr.Close();

cn.Close();

if (found)

{

if (qty < (int.Parse(txtQty.Text) + cart\_qty))

{

MessageBox.Show("Помилка. Неможливо зарахувати товару більше ніж є в наявності - " + qty, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

cn.Open();

cm = new SqlCommand("Update Cart set Crt\_Qty = Crt\_Qty + " + int.Parse(txtQty.Text) + " Where Crt\_Id= " + id, cn);

cm.ExecuteReader();

cn.Close();

cashier.txtBarcode.Clear();

cashier.txtBarcode.Focus();

cashier.LoadCart();

this.Dispose();

}

else

{

if (qty < (int.Parse(txtQty.Text) + cart\_qty))

{

MessageBox.Show("Помилка. Неможливо зарахувати товару більше ніж є в наявності - " + qty, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

cn.Open();

cm = new SqlCommand("INSERT INTO Cart(Crt\_Transno, Crt\_Product, Crt\_Price, Crt\_Qty, Crt\_Cashier) VALUES(@transno, @pcode, @price, @qty, (SELECT TOP 1 Emp\_Id FROM Employee INNER JOIN [User] ON Emp\_Id = Usr\_Employee WHERE Usr\_Username = '" + cashier.lblUsername.Text + "'))", cn);

cm.Parameters.AddWithValue("@transno", transno);

cm.Parameters.AddWithValue("@pcode", pcode);

cm.Parameters.AddWithValue("@price", price);

cm.Parameters.AddWithValue("@qty", int.Parse(txtQty.Text));

cm.ExecuteNonQuery();

cn.Close();

cashier.txtBarcode.Clear();

cashier.txtBarcode.Focus();

cashier.LoadCart();

this.Dispose();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, stitle);

}

}

}

private void Qty\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Dispose();

}

}

}

}

public partial class ResetPassword : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

UserAccount user;

public ResetPassword(UserAccount account)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

user = account;

}

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

{

if(txtNpass.Text != txtResPass.Text)

{

MessageBox.Show("The password you typed do not match. Type the password for this account in both text boxes.", "Add User Wizard", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

if (MessageBox.Show("Reset password?", "Confirm", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)

{

dbcon.ExecuteQuery("UPDATE tbUser SET password = '" + txtNpass.Text + "'WHERE username = '"+ user.username +"'");

MessageBox.Show("Password has been successfully reset", "Reset Password", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Dispose();

}

}

}

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

{

this.Dispose();

}

private void ResetPassword\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

this.Dispose();

}

}

}

}

public partial class Settle : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

Cashier cashier;

public Settle(Cashier cash)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

this.KeyPreview = true;

cashier = cash;

}

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

{

txtCash.Text += btnOne.Text;

}

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

{

txtCash.Text += btnTwo.Text;

}

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

{

txtCash.Text += btnThree.Text;

}

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

{

txtCash.Text += btnFour.Text;

}

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

{

txtCash.Text += btnFive.Text;

}

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

{

txtCash.Text += btnSix.Text;

}

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

{

txtCash.Text += btnSeven.Text;

}

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

{

txtCash.Text += btnEight.Text;

}

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

{

txtCash.Text += btnNine.Text;

}

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

{

txtCash.Text += btnZero.Text;

}

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

{

txtCash.Text += btnDZero.Text;

}

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

{

txtCash.Clear();

txtCash.Focus();

}

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

{

try

{

if ((double.Parse(txtChange.Text) < 0) || (txtCash.Text.Equals("")))

{

MessageBox.Show("Введіть коректну суму!", "Увага", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

else

{

for(int i=0; i< cashier.dgvCash.Rows.Count; i++ )

{

cn.Open();

cm = new SqlCommand("UPDATE OnStock SET OnStk\_Quantity = OnStk\_Quantity - " + cashier.dgvCash.Rows[i].Cells[5].Value.ToString() + " WHERE OnStk\_Product = " + cashier.dgvCash.Rows[i].Cells[2].Value.ToString(), cn);

cm.ExecuteNonQuery();

cn.Close();

cn.Open();

cm = new SqlCommand("UPDATE Cart SET Crt\_Status = 2 WHERE Crt\_Transno = '" + cashier.lblTranNo.Text + "'", cn);

cm.ExecuteNonQuery();

cn.Close();

}

MessageBox.Show("Оплата успішно прошла!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

PrintRecept();

cashier.GetTranNo();

cashier.LoadCart();

this.Dispose();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void txtCash\_TextChanged(object sender, EventArgs e)

{

try

{

double sale = double.Parse(txtSale.Text);

double cash = double.Parse(txtCash.Text);

double charge = cash - sale;

txtChange.Text = charge.ToString("#,##0.00");

}

catch (Exception)

{

txtChange.Text = "0.00";

}

}

private void Settle\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape) this.Dispose();

else if (e.KeyCode == Keys.Enter) btnEnter.PerformClick();

}

private void PrintRecept()

{

cn.Open();

cm = new SqlCommand("exec GetReceiptInfo @transno", cn);

cm.Parameters.AddWithValue("@transno", cashier.lblTranNo.Text);

dr = cm.ExecuteReader();

Report report = new Report();

report.GenerateReceipt(dr);

}

}

}

public partial class StockIn : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

string stitle = "Market";

MainForm main;

public StockIn(MainForm mn)

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

main = mn;

LoadStockInList();

}

public void LoadStockInList()

{

dgvStockIn.Rows.Clear();

int i = 0;

try

{

cm = new SqlCommand("exec GetStockInList", cn);

cn.Open();

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvStockIn.Rows.Add(i, dr[0].ToString(),

dr[3].ToString(),

dr[1].ToString(),

dr[2].ToString(),

//dr[4].ToString(),

dr[5].ToString()

);

if (dr[5].ToString() == "Підтверджено")

{

dgvStockIn.Rows[i - 1].Cells[6].Value = Image.FromFile("../../Image/open\_lock.jpg");

}

else

{

dgvStockIn.Rows[i - 1].Cells[6].Value = Image.FromFile("../../Image/locked\_lock.png");

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dr.Close();

cn.Close();

}

}

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

{

StockInModule stockInModule = new StockInModule();

stockInModule.btnSave.Enabled = true;

try

{

cn.Open();

cm = new SqlCommand("insert into StockIn (Sti\_Date) values (NULL)", cn);

cm.ExecuteNonQuery();

cm = new SqlCommand("SELECT TOP 1 Sti\_Id FROM StockIn ORDER BY Sti\_Id DESC", cn);

dr = cm.ExecuteReader();

while (dr.Read())

{

stockInModule.LoadDoc(dr[0].ToString());

}

stockInModule.Show();

}

catch (Exception ex)

{

MessageBox.Show (ex.Message);

}

finally

{

cn.Close();

dr.Close();

}

}

private void dgvStockIn\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

StockInModule stockInModule = new StockInModule();

stockInModule.LoadDoc(dgvStockIn.Rows[e.RowIndex].Cells[1].Value.ToString());

stockInModule.Show();

}

private void label1\_Click(object sender, EventArgs e)

{

LoadStockInList();

}

private void dgvStockIn\_CellPainting(object sender, DataGridViewCellPaintingEventArgs e)

{

if (e.RowIndex > -1 && dgvStockIn.Rows[e.RowIndex].Cells[5].Value.ToString() == "Підтверджено")

{

e.CellStyle.BackColor = Color.LightGreen;

}

}

private void dgvStockIn\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

string colName = dgvStockIn.Columns[e.ColumnIndex].Name;

if (colName == "Delete" && dgvStockIn.Rows[e.RowIndex].Cells[5].Value.ToString() != "Підтверджено")

{

if (MessageBox.Show("Видалити обране замовлення", "Видалити", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cm = new SqlCommand("DELETE FROM StockIn WHERE Sti\_Id = " + dgvStockIn.Rows[e.RowIndex].Cells[1].Value.ToString(), cn);

cn.Open();

int res = cm.ExecuteNonQuery();

cn.Close();

if (res > 0)

{

MessageBox.Show("Замовлення було успішно видалено!");

LoadStockInList();

}

else

{

MessageBox.Show("Сталася помилка при видалені замовлення!");

}

}

}

}

}

}

}

public partial class Store : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

bool havestoreinfo = false;

public Store()

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

LoadStore();

}

public void LoadStore()

{

try

{

cn.Open();

cm = new SqlCommand("SELECT \* FROM Store", cn);

dr = cm.ExecuteReader();

dr.Read();

if(dr.HasRows)

{

havestoreinfo = true;

txtStName.Text = dr["Str\_Title"].ToString();

txtAddress.Text = dr["address"].ToString();

}

else

{

txtStName.Clear();

txtAddress.Clear();

}

dr.Close();

cn.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка");

}

}

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

{

try

{

if (MessageBox.Show("Зберегти зміни?", "Підтвердженяя", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

if(havestoreinfo)

{

dbcon.ExecuteQuery("UPDATE Store SET Str\_Title = '" + txtStName.Text + "', address= '" + txtAddress.Text + "'");

}

else

{

dbcon.ExecuteQuery("INSERT INTO Store (Str\_Title,address) VALUES ('" + txtStName.Text + "','" + txtAddress.Text + "')");

}

MessageBox.Show("Дані було успішно оновлені!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Dispose();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка");

}

}

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

{

this.Dispose();

}

private void Store\_KeyDown(object sender, KeyEventArgs e)

{

if(e.KeyCode==Keys.Escape)

{ this.Dispose(); }

}

}

}

public partial class Supplier : Form

{

SqlConnection cn = new SqlConnection();

SqlCommand cm = new SqlCommand();

DBConnect dbcon = new DBConnect();

SqlDataReader dr;

public Supplier()

{

InitializeComponent();

cn = new SqlConnection(dbcon.myConnection());

LoadSupplier();

}

public void LoadSupplier()

{

dgvSupplier.Rows.Clear();

int i = 0;

cn.Open();

cm = new SqlCommand("SELECT \* FROM Supplier", cn);

dr = cm.ExecuteReader();

while (dr.Read())

{

i++;

dgvSupplier.Rows.Add(i, dr[0].ToString(), dr[1].ToString(), dr[2].ToString(), dr[3].ToString(), dr[4].ToString(), dr[5].ToString(), dr[6].ToString());

}

dr.Close();

cn.Close();

}

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

{

SupplierModule supplierModule = new SupplierModule(this);

supplierModule.btnUpdate.Enabled = false;

supplierModule.btnSave.Enabled = true;

supplierModule.ShowDialog();

}

private void dgvSupplier\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

string colName = dgvSupplier.Columns[e.ColumnIndex].Name;

if (colName == "Edit")

{

SupplierModule supplierModule = new SupplierModule(this);

supplierModule.lblId.Text = dgvSupplier.Rows[e.RowIndex].Cells[1].Value.ToString();

supplierModule.txtSupplier.Text = dgvSupplier.Rows[e.RowIndex].Cells[2].Value.ToString();

supplierModule.txtAddress.Text = dgvSupplier.Rows[e.RowIndex].Cells[3].Value.ToString();

supplierModule.txtConPerson.Text = dgvSupplier.Rows[e.RowIndex].Cells[4].Value.ToString();

supplierModule.txtPhone.Text = dgvSupplier.Rows[e.RowIndex].Cells[5].Value.ToString();

supplierModule.txtEmail.Text = dgvSupplier.Rows[e.RowIndex].Cells[6].Value.ToString();

supplierModule.txtFaxNo.Text = dgvSupplier.Rows[e.RowIndex].Cells[7].Value.ToString();

supplierModule.btnSave.Enabled = false;

supplierModule.btnUpdate.Enabled = true;

supplierModule.ShowDialog();

}

else if(colName=="Delete")

{

if (MessageBox.Show("Ви хочете видалити данного постачальника?", "Видалити", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

cn.Open();

cm = new SqlCommand("DELETE FROM Supplier WHERE Sup\_Id = " + dgvSupplier.Rows[e.RowIndex].Cells[1].Value.ToString(), cn);

cm.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Запис було успішно видалено.", "Market", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

LoadSupplier();

}

}

}

АВТОМАТИЗАЦІЯ ОБЛІКУ ТА АНАЛІЗУ ПРОДАЖІВ НА ОПТОВІЙ БАЗІ

## Посібник програміста

ДП.ХПТК.00452 – 41 33 01

(компакт-диск)

Аркушів 2

2023

Розроблений програмний продукт є програмою для обліку та аналізу продаж товарів на оптовій базі.

Готова програма у форматі \*.exe може бути завантажена на такі типи операційних систем: Windows 7; Windows XP; Windows Vista; Windows 10, Windows 11.

Головним критерієм до машинного устаткування повинна бути підтримка 64-bit програм, та об’єм оперативної пам’яті не менше ніж 512 Мб.

Розробка виконувалася у програмі Visual Studio 2022, тому найкраще продовжувати розробку у даній програмі. Для створення бази даних було використано Microsoft SQL Server 2019.

У програму підключена база даних і саме з нею, безпосередньо, працює оператор.

Програма створена для адміністратора та менеджера з продаж, тому база даних і програма повинні бути забезпечені необхідною для користувачів інформацією. Інформація необхідна для адміністратора: список товарів, список постачальників, список виробників, список товарних накладних, аналітична звітність. Для зручності використання деяка інформація представлена у вигляді і форм, і звітів.

Перш за все програма розрахована для внесення та відображення даних і тому користувач повинен мати змогу на додавання, редагування, видалення та перегляд записів.

Робота з програмою повинна бути комфортною та зрозумілою для будь-якого рівня знань комп’ютера, тому треба забезпечити максимально зручний інтерфейс.

АВТОМАТИЗАЦІЯ ОБЛІКУ ТА АНАЛІЗУ ПРОДАЖІВ НА ОПТОВІЙ БАЗІ

## Посібник оператора

ДП.ХПТК.00452 – 41 34 01

(компакт-диск)

Аркушів 5

2023

Після запуску програми відкриється вікно авторизації, на якій користувач вводить свій логін та пароль (див. рис. 3.1).

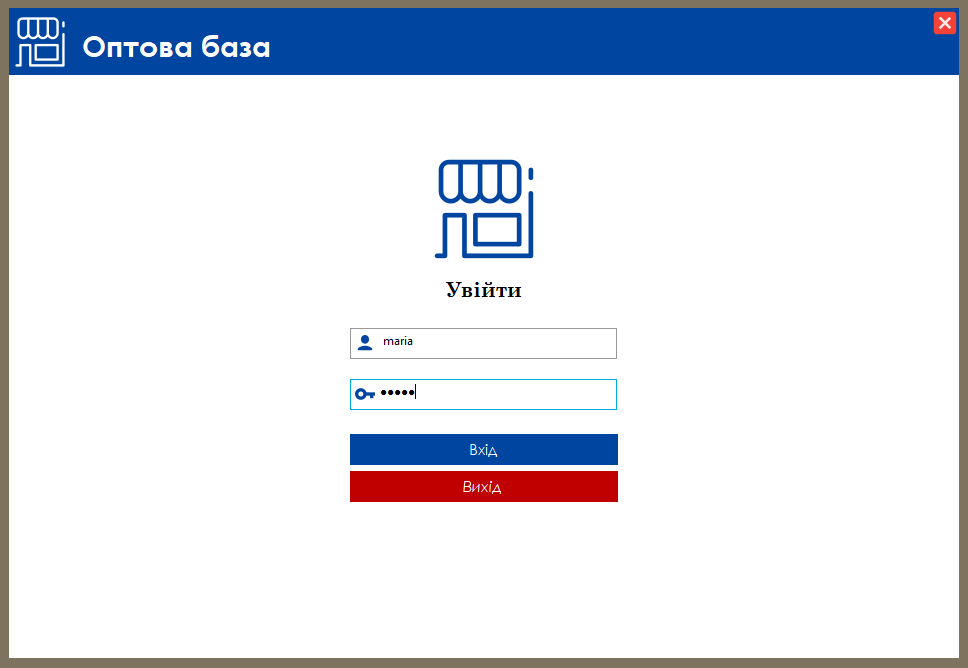


Рисунок 3.1 – Форма авторизації

На основі логіну та паролю відкривається головна форма або для адміністратора або для менеджера з продаж(див. рис. 3.2 та рис. 3.3).

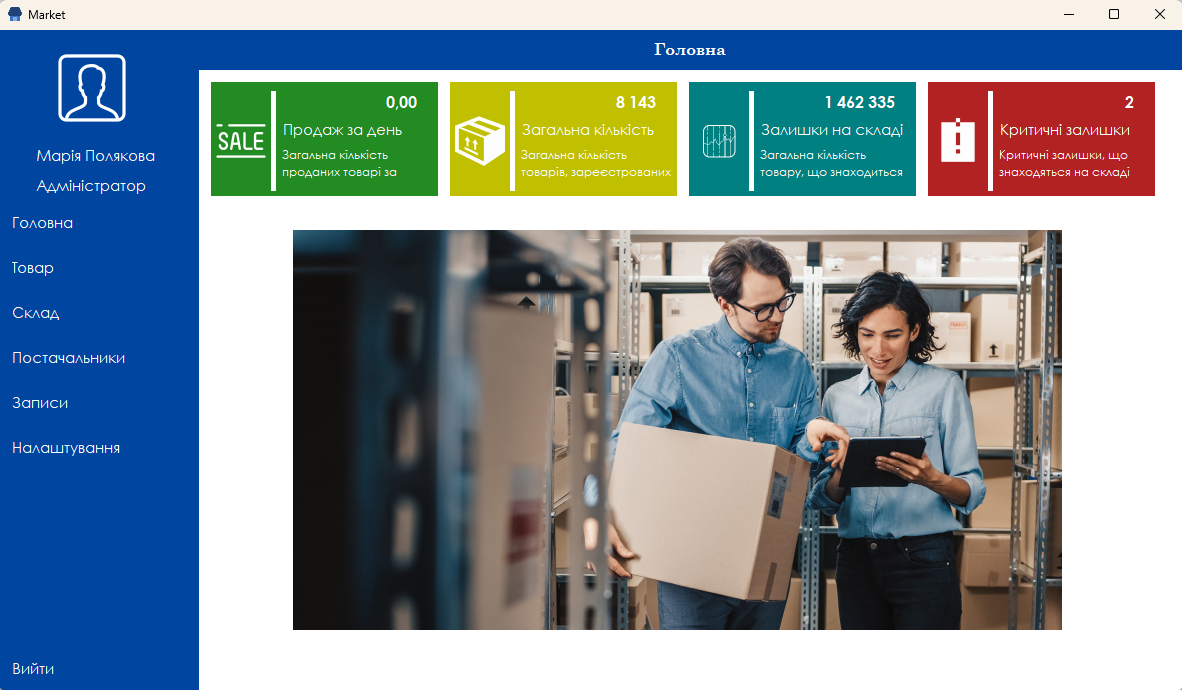


Рисунок 3.2 – Головна форма для адміністратора

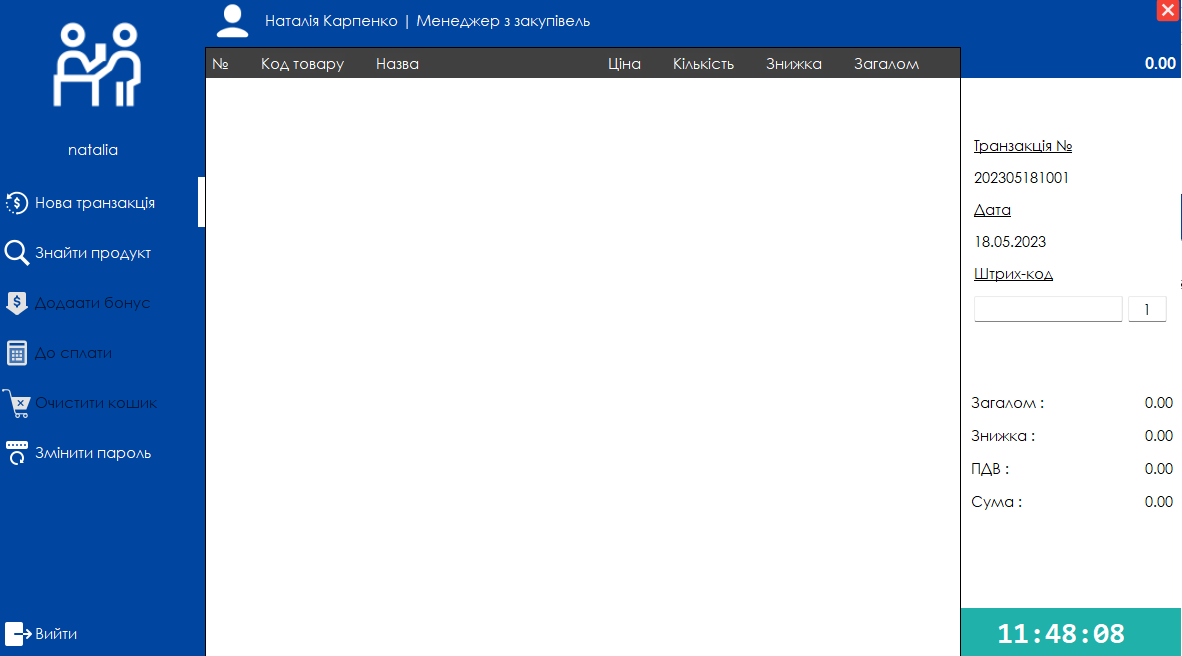


Рисунок 3.3 – Головна форма для менеджера з продаж

Головна форма адміністратора зліва має перелік вкладок. При натисканні на вкладку відкривається перелік підвкладок, кожна з яких відкриває відповідну форма, з якою вже можна працювати(див. рис. 3.4).

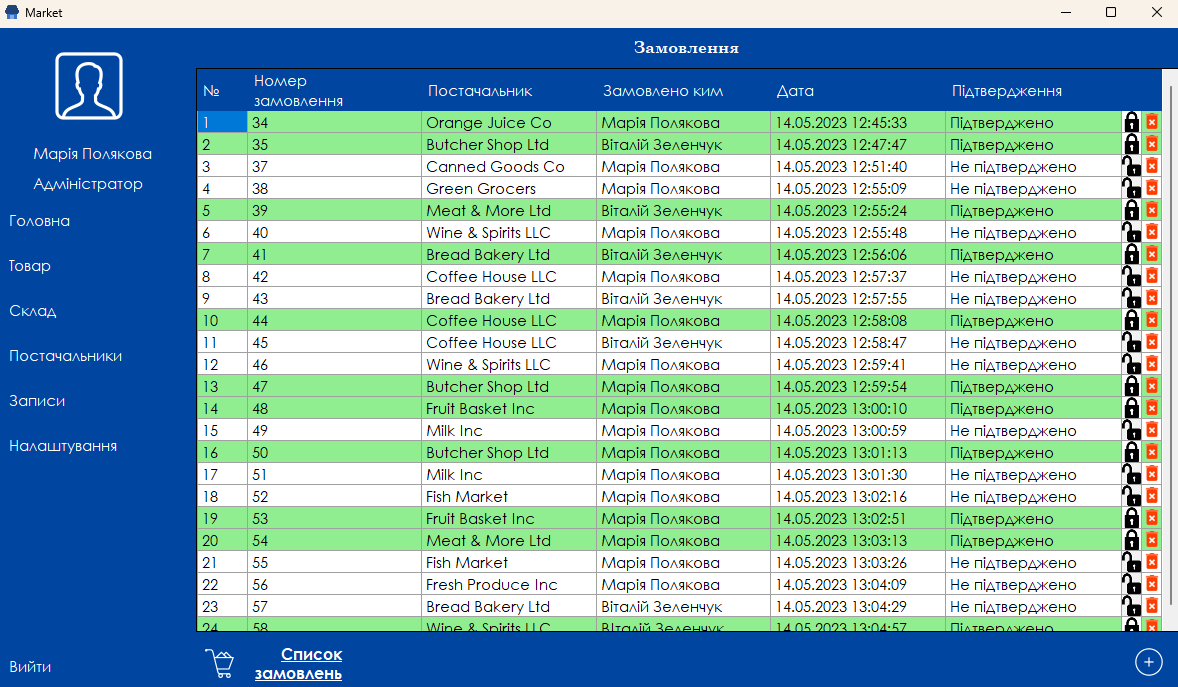


Рисунок 3.4 – Форма зі списком замовлень

На головній формі менеджера з продаж при натисканні кнопки Додати продукт можна додати продукт та підтвердити його кількість (див. рис. 3.5).

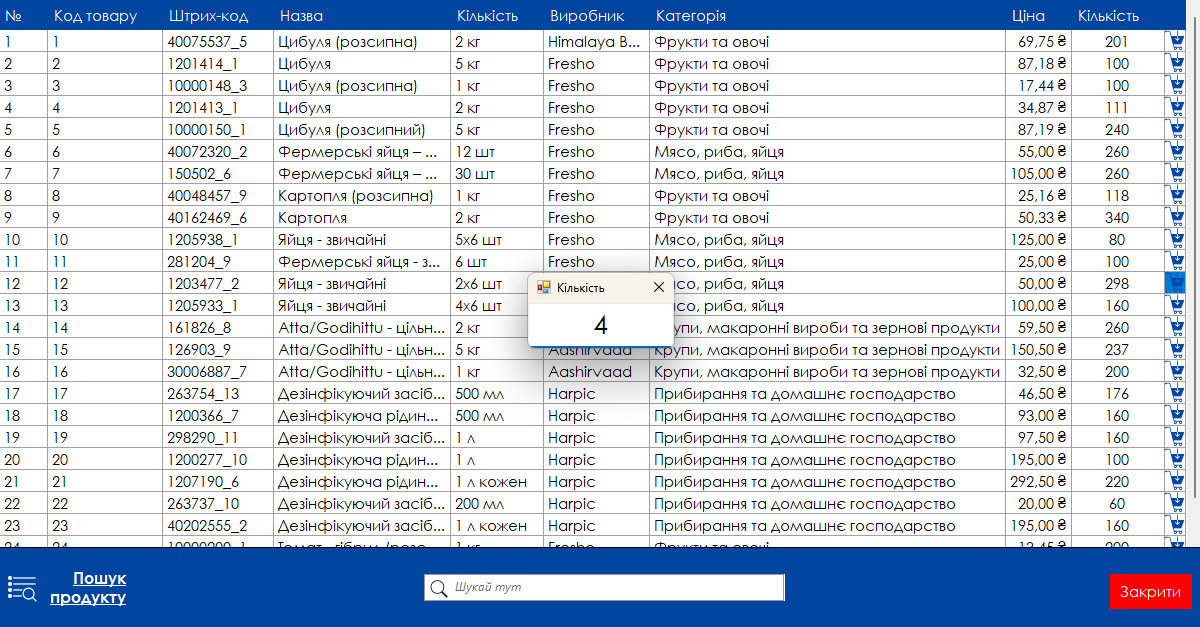


Рисунок 3.5 – додавання продукту до кошика

Після додавання всіх потрібних продуктів можна перейти до оплати (див. рис. 3.6 ).

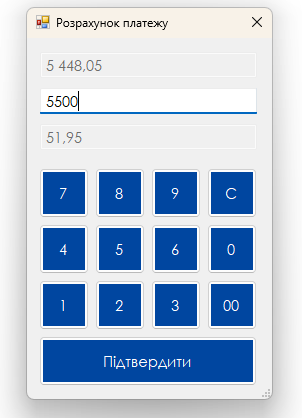


Рисунок 3.6 – форма розрахунку оплати

Після оплати на екрані з’явиться чек (див. рис. 3.7).

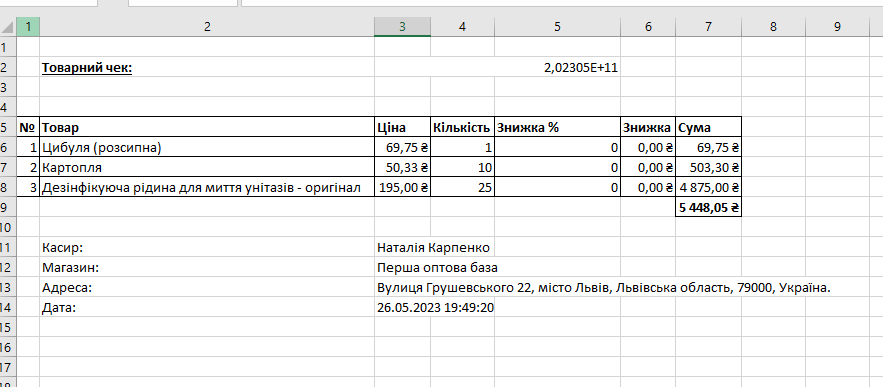


Рисунок 3.7 – роздрук товарного чеку

АВТОМАТИЗАЦІЯ ОБЛІКУ ТА АНАЛІЗУ ПРОДАЖІВ НА ОПТОВІЙ БАЗІ

## Специфікація

ДП.ХПТК.00452 – 41

(компакт-диск)

Аркушів 2

2023

* 1. СПЕЦИФІКАЦІЯ

|  |  |  |
| --- | --- | --- |
| Позначення | Найменування | Примітка |
| ДП.ХПФК.00452 – 41 12 01 | Текст програми | с. 67 – 105 |
| ДП.ХПФК.00452 – 41 33 01 | Посібник програміста | с.107 |
| ДП.ХПФК.00452 – 41 34 01 | Посібник оператора | с. 109 - 112 |
| ДП.ХПФК.00452 – 41 81 01 | Пояснювальна записка | с. 1 - 67 |