НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

***з дисципліни "Бази даних"***

Виконав: Підручний Владислав Миколайович

Група: КП-01

Допущено до захисту

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

1 семестр 2021/2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

| Узгоджено  Керівник роботи    \_\_\_\_\_\_\_\_\_\_/Радченко К../ | ЗАХИЩЕНА  "\_\_"\_\_\_\_\_\_\_\_\_2021р.  з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_/Радченко К./ |
| --- | --- |

***Програмний додаток електронного магазину товарів***

Виконавець роботи

Підручний Владислав Миколайович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021р.

Зміст

[**Вступ**](#_30j0zll) **[4](#_30j0zll)**

[**1. Аналіз предметної галузі**](#_1fob9te) **[5](#_1fob9te)**

[1.1. Аналіз вимог до функціональності програмних засобів](#_3znysh7) [5](#_3znysh7)

[**2. Аналіз мов програмування та технологій розроблення**](#_2et92p0) **[5](#_2et92p0)**

[2.1. Мова програмування С](#_tyjcwt)# [5](#_tyjcwt)

[2.2. Фреймворк Qt](#_1t3h5sf) [5](#_1t3h5sf)

[2.3. База даних SQLite](#_4d34og8) [5](#_4d34og8)

[2.4. Бібліотека …. (назва бібліотеки, тут лише не ваші бібліотеки)](#_2bn6wsx) [5](#_2bn6wsx)

[**3. Опис розроблених програмних засобів**](#_3rdcrjn) **[6](#_3rdcrjn)**

[3.1. Загальна структура системи](#_26in1rg) [6](#_26in1rg)

[3.2. Структура бази даних](#_lnxbz9) [7](#_lnxbz9)

[3.3. Бібліотека для роботи з форматом CSV](#_3j2qqm3) [8](#_3j2qqm3)

[3.4. Модуль сховища даних](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.dpq6kpdwm35k) [8](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.dpq6kpdwm35k)

[3.5. Модуль X](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.7nq5dney5jkd) [8](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.7nq5dney5jkd)

[3.6. Модуль XX](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.8x2ts4hd9895) [9](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.8x2ts4hd9895)

[**4. Аналіз розроблених програмних засобів**](#_4i7ojhp) **[10](#_4i7ojhp)**

[4.1. Особливості реалізації](#_2xcytpi) [10](#_2xcytpi)

[4.2. Дизайн та вміст вікон](#_qsh70q) [10](#_qsh70q)

[**Висновки**](#_3as4poj) **11**

# **Вступ**

В процесі постійної потреби купувати товари різних категорії, люди витрачають багато часу на походи в магазини. Тоді виникає необхідність створення інтернет-магазинів, адже це набагато зручніше ніж звичайний похід в магазин. Саме тому пропонується розробка програмного додатку електронного магазину товарів. Саме цей додаток дозволяє користувачу зробити замовлення бажаного товару з будь-якої точки світу.

# 1. Аналіз предметної галузі

## 1.1. Аналіз вимог до функціональності програмних засобів

В процесі аналізу вимог до системи було виділено такі функціональні вимоги:

1. Наявний консольний інтерфейс користувача. Команди використані відповідно до їхнього основного призначення.

a. Наявно декілька станів інтерфейсу користувача.

b. Дотримується єдиний стиль для всіх команд

2. Наявні стани для створення і редагування даних.

1. Можна видаляти дані (з підтвердженням цієї дії).
2. Наявний стан редагування даних має мати вже заповнені поля із попередніми значеннями. Є можливість відмінити оновлення сутності.

3. Використання списків та таблиць для показу колекцій даних.

1. Дані в таблиці не є лише відображенням однієї таблиці БД, а містять склеєні чи змінені дані.

b. Дані у списках та таблицях пагінуються, доступний пошук та фільтрація їх записів.

4. Всі вхідні дані від користувача та з файлів перевіряються, помилки обробляються, користувачу відображається інформація про помилки.

5. Програма має можливість завантаження зображень з файлової системи, їх збереження у сховище даних та їх показ у інтерфейсі користувача.

6. Програма має можливість експорту-імпорту даних у форматі даних XML. Є можливість зберігання даних у файлах, та передача даних до інших процесів через протокол TCP.

7. Програма взаємодіє з реляційною базою даних. SQL запити до бази даних не конкатенуються, а їх виконання винесено у спеціальний окремий модуль (вид репозиторія).

1. База даних містить мінімум 3 таблиці. Таблиці в БД приведені до 1-ї та 2-ї нормальних форм.

b. Між таблицями бази даних є one-to-many та/або many-to-many зв'язки

8. Наявна реєстрація користувачів у системі. Паролі користувачів не зберігаються у відкритому виді, а хешуються обраним алгоритмом хешування рядків.

a. Наявний стан аутентифікації. Пароль користувача при введені прихований. Цей стан має сповіщення про некоректність введених користувачем даних.

b. Програма авторизує доступу до даних по користувачах. У базі є дані, що належать конкретному користувачу (наприклад, авторство сутностей).

9. Система дозволяє генерувати зображення та файлові звіти на основі даних, що зберігаються у БД.

10. Використання окремих потоків виконання для паралельних фонових дій.

11. Використання клієнт-серверної архітектури програм та TCP сокетів (розділення системи на клієнт і сервер з передачею даних у одному із стандартних форматів даних)

12. Частина проекту винесена у власні бібліотеки класів. Винесення саме цього коду обгрунтоване.

13. Наявний проект з модульними тестами функціональності системи.

# 2. Аналіз мов програмування та технологій розроблення

## 2.1. Мова програмування С#

Безперечними перевагами середовища C-Шарп є:

* зручний і функціональний редактор;
* продумані призначені для користувача конструктори;
* вбудований відладчик коду та інші помічники розробника.

За допомогою можливостей мови C# було реалізовано програмний додаток Інтернет - магазину.

## 2.2. Фреймворк .NET

* Забезпечення об’єктно-орієнтованого середовища ,для локального зберігання і виконання коду.
* Надання середовища виконання в якому гарантується безпечне виконання коду
* Забезпечення єдиних принципів розробки для різних типів додатків.

## 2.3. База даних SQLite

SQLite - це вбудована бібліотека, яка реалізує автономний, безсерверний, нульової конфігурації, транзакційний механізм СУБД SQL. Це база даних, яка налаштована на нуль, що означає, як і інші бази даних, які вам не потрібно налаштовувати в вашій системі.

2.4. Бібліотека Terminal.Gui -

Terminal.Gui - це бібліотека, призначена для створення консольних програм на основі C #. Структура розроблена для спрощення написання програм, які працюватимуть на монохромних терміналах, а також сучасних кольорових терміналах з підтримкою миші.

Ця бібліотека надає текстовий набір інструментів, який працює подібним чином до графічних.

## 2.5. Бібліотека Microsoft.Data.Sqlite -

* SQLite легка і автономна. Це бібліотека коду без інших залежних компонентів. Її не потрібно налаштовувати.
* Тут немає сервера бази даних. Клієнт і сервер працюють в одному процесі.
* База даних SQLite знаходиться в загальнодоступному домені, тому ви можете вільно використовувати і поширювати її в вашому додатку.
* SQLite працює на різних платформах і архітектурі.

## 2.6. Бібліотека ScottPlot

ScottPlot - бібліотека для побудови графіків, яка полегшує інтерактивне відображення великих наборів даних. Лінійні графіки, гістограми, кругові діаграми, діаграми розсіювання та багато іншого можна створити лише за можливостей цієї бібліотеки. У цій роботі було створено графік для дослідження середньої вартості замовлення протягом певного періоду часу.

# 3. Опис розроблених програмних засобів

## 3.1. Загальна структура системи

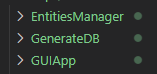
Програмні засоби реалізовані у вигляді десктопного додатку з графічним інтерфейсом користувача. Структурна схема системи зображена на рис.



Рис. Структурна схема системи

Проект розбитий на такі модулі:

Рис. Модулі проекту



## 

## 

## 3.2. Структура бази даних

На рис. представлена ERD-діаграма структури бази даних системи.

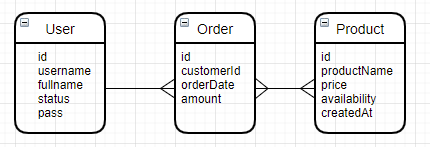


Рис. Зв’язки між таблицями в ERD

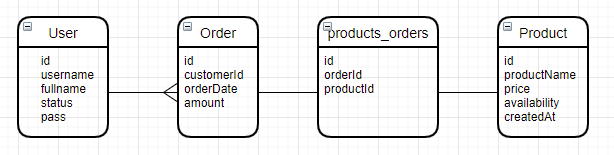


Рис. з додатковою сутностю для реалізація зв’язку many-to-many

База даних налічує 4 таблиці з наступними назвами -

1. users
2. orders
3. products
4. products\_orders

User -

Атрибути-

* id
* username
* fullname
* status
* pass

Дана таблиця має зв'язок one - to - many з таблицею orders, тобто один користувач може мати декілька замовлень.Але замовлення завжди належить лише одному покупцю.

Orders-

Атрибути-

* id
* customerId
* orderDate
* amount

Products\_orders-

Атрибути-

* id
* orderId
* productId

Дана таблиця є допоміжною для реалізації зв'язку many - to - many між Orders та Products, адже товари можуть належати різним замовленням, а замовлення можуть мати різні товари.

Ця таблиця має зв'язки one - to - one з таблицями Orders та Products.

Products -

Атрибути-

* id
* productName
* price
* availability
* createdAt

Таблиця має зв'язок з таблицею Orders за допомогою таблиці Products\_orders.

## 3.3. Модулі репозиторіїв даних

Інтерфейс модуля:

| OrderRep.cs |
| --- |
| using System;  using Microsoft.Data.Sqlite;  using System.Collections.Generic;  namespace EntitiesManager  {  public class OrderRep  {  private SqliteConnection connection;  public OrderRep(SqliteConnection connection)  {  this.connection = connection;  }  public long Insert(Order order)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText =  @"INSERT INTO orders (customerId, orderDate, amount)  VALUES ($customerId, $orderDate, $amount);  SELECT last\_insert\_rowid();";  command.Parameters.AddWithValue("$customerId", order.customerId);  command.Parameters.AddWithValue("$orderDate", order.orderDate.ToString("o"));  command.Parameters.AddWithValue("$amount", order.amount);  long lastId = (long)command.ExecuteScalar();  connection.Close();  return lastId;  }  public List<Order> GetAllOrders()  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM orders";  SqliteDataReader reader = command.ExecuteReader();  List<Order> orders = new List<Order>();  while (reader.Read())  {  Order order = GetOrder(reader);  orders.Add(order);  }  reader.Close();  connection.Close();  return orders;  }  public List<Order> GetAllById(long id)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = $"SELECT \* FROM orders WHERE customerId = {id}";  SqliteDataReader reader = command.ExecuteReader();  List<Order> orders = new List<Order>();  while (reader.Read())  {  Order order = GetOrder(reader);  orders.Add(order);  }  reader.Close();  connection.Close();  return orders;  }  public Order GetOrderById(long id)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM orders WHERE id = $id";  command.Parameters.AddWithValue("$id", id);  SqliteDataReader reader = command.ExecuteReader();  Order order = new Order();  if (reader.Read())  {  order = GetOrder(reader);  }  connection.Close();  return order;  }  public static Order GetOrder(SqliteDataReader reader)  {  Order order = new Order();  order.id = int.Parse(reader.GetString(0));  order.customerId = long.Parse(reader.GetString(1));  order.orderDate = DateTime.Parse(reader.GetString(2));  order.amount = int.Parse(reader.GetString(3));  return order;  }  public List<long> GetProductIds(Order order)  {  connection.Open();  List<long> productIds = new List<long>();  SqliteCommand command = connection.CreateCommand();  command.CommandText = $"SELECT productId FROM products\_orders WHERE orderId={order.id}";  SqliteDataReader reader = command.ExecuteReader();  while (reader.Read())  {  productIds.Add(reader.GetInt64(0));  }  connection.Close();  return productIds;  }  public int DeleteById(long id)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"DELETE FROM orders WHERE id = $id";  command.Parameters.AddWithValue("$id", id);  int nChanged = command.ExecuteNonQuery();  connection.Close();  if (nChanged == 0)  {  return 0;  }  return 1;  }  public bool Update(long id, Order order)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = $"UPDATE orders SET customerId = $customerId, orderDate = $orderDate, amount = $amount WHERE id = $id";  command.Parameters.AddWithValue("$id", order.id);  command.Parameters.AddWithValue("$customerId", order.customerId);  command.Parameters.AddWithValue("$orderDate", order.orderDate);  command.Parameters.AddWithValue("$amount", order.amount);  int rowChange = command.ExecuteNonQuery();  connection.Close();  if (rowChange == 0)  {  return false;  }  return true;  }  public bool AddProductConection(long productId, long orderId)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText =  @"INSERT INTO products\_orders (orderId, productId)  VALUES ($orderId, $productId);  SELECT last\_insert\_rowid();";  command.Parameters.AddWithValue("$orderId", orderId);  command.Parameters.AddWithValue("$productId", productId);  long lastId = (long)command.ExecuteScalar();  connection.Close();  if (lastId == 0)  {  return false;  }  return true;  }  public bool DeleteProductConection(long orderId)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText =  @"DELETE FROM products\_orders WHERE orderId = $orderId";  command.Parameters.AddWithValue("$orderId", orderId);  object a = command.ExecuteScalar();  long lastId = a == null ? 0 : (long)a;  connection.Close();  if (lastId == 0)  {  return false;  }  return true;  }  public List<long> GetAllOrderProductsId(long orderId)  {  List<long> prodIdList = new List<long>();  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM products\_orders WHERE orderId = $orderId";  command.Parameters.AddWithValue("$orderId", orderId);  SqliteDataReader reader = command.ExecuteReader();  while (reader.Read())  {  long productId = long.Parse(reader.GetString(2));  prodIdList.Add(productId);  }  connection.Close();  return prodIdList;  }  public List<Order> GetAllUserOrdersById(List<long> orderId)  {  List<Order> ordersList = new List<Order>();  connection.Open();  foreach (long id in orderId)  {  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM orders WHERE id = $id";  command.Parameters.AddWithValue("$id", id);  SqliteDataReader reader = command.ExecuteReader();  if (reader.Read())  {  Order order = GetOrder(reader);  ordersList.Add(order);  }  }  connection.Close();  return ordersList;  }  }  } |

Даний модуль призначений для для керування БД із даними користувача і надає можливість здійснювати такі операції -

* створення нового замовлення
* отримання списку замовлення
* видалення замовлення
* редагування замовлення
* отримання всіх записів з бд

## 3.5. Модуль експорту -

Модуль експорту містить методи для автоматичної серелізації та десерелізації.

Використовується цей модуль, якщо необхідно імпортувати дані про користувача та всі його замовлення або потрібно отримати ці дані.

## 3.6. Модуль Авторизації -

Модуль авторизації містить методи перевірки існування введеного логіну, отримання захешованого паролю та перевірки введеного паролю. Даний модуль використовується при реєстрації нового користувача або при спробі користувача увійти до профілю.

## 3.7.Модуль GenerateDocument -

Даний модуль використовується при генерації звіту з інформацією про певного користувача. Модуль містить методи для виконання розархівування та архівування та наявний метод за допомогою якого відбувається зміна текстових даних а також зміна зображення.

## 3.8 Бібліотека EntitiesManager -

Бібліотека містить модулі всіх сутностей та модулі репозиторіїв відповідних сутностей.

Також до бібліотеки було винесено модулі експорту та імпорту у новий проект. Було додано посилання на бібліотеку класів у існуючий консольний проект, а також посилання на бібліотеку у консольний проект генератора.

## 3.9 Бібліотека GenerateDB -

Бібліотека містить генерацію сутностей(користувачі, продукти). Було додано посилання на бібліотеку класів у існуючий консольний проект, а також посилання на бібліотеку EntitiesManager.

# 4. Аналіз розроблених програмних засобів

## 4.1. Особливості реалізації

1. Особливості авторизації користувачів -

Після запуску програми, користувачу необхідно або авторизуватися або зареєструватися. Відповідно користувачу надається можливість ввести дані, а саме для входу в акаунт - username, password , а також повна інформація про користувача при реєстрації. Після введення відбувається перевірка логіну на наявність в базі даних та перевірка паролю на відповідність (при авторизації). При успішному наданні даних, користувач потрапляє в головне меню нашого інтернет-магазину.

1. Ролі користувачів- -

За умовою завдання необхідно поділити користувачів на ролі – адмін та “простий” користувач. Адміну надається право створювати нові товари і визначати їх наявність і всі можливості користувача. Користувач в свою чергу може створювати замовлення, переглядати історію замовлень, згенерувати звіт з інформацією про використання інтернет - магазину.

1. Спосіб зберігання паролів -

При реєстрації пароль хешується за допомогою алгоритму SHA256.

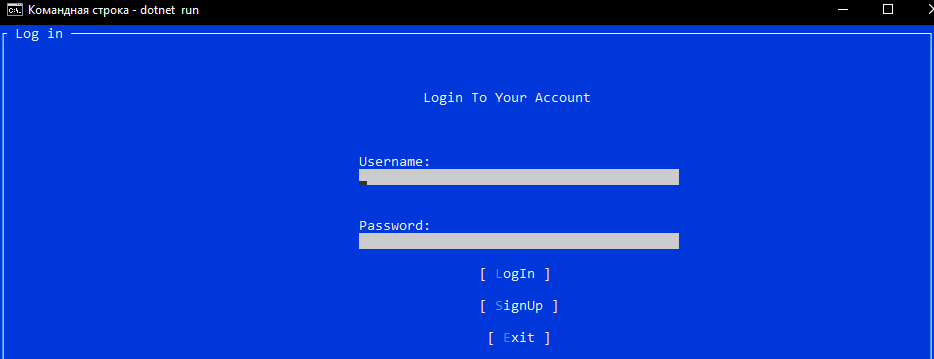
До бази даних додається новий користувач, пароль якого зберігається у хешованому вигляді.

1. Серіалізація

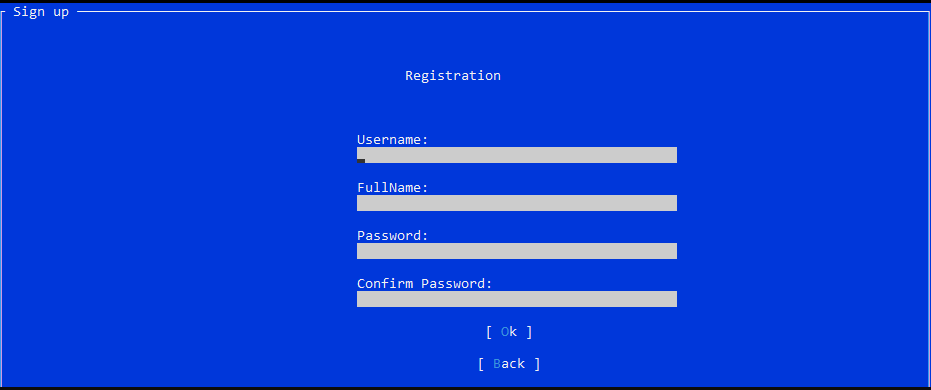
При імпортуванні даних відбувається автоматична серелізація, тобто користувач з його замовленнями перетворюється у XML рядок.

## 4.2. Інтерфейс користувача

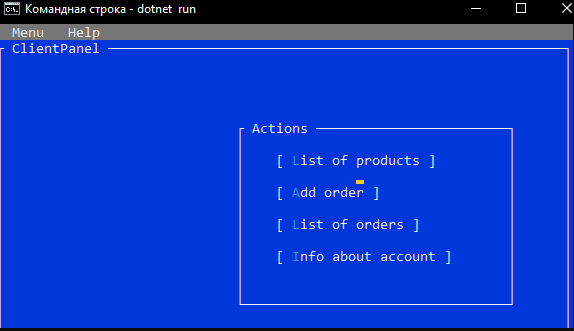
Для показу даних система має інтерфейс користувача, що складається з:



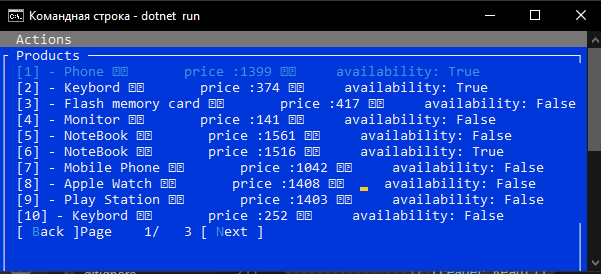
***Рис.1.*** *Головне вікно програмного додатку.*



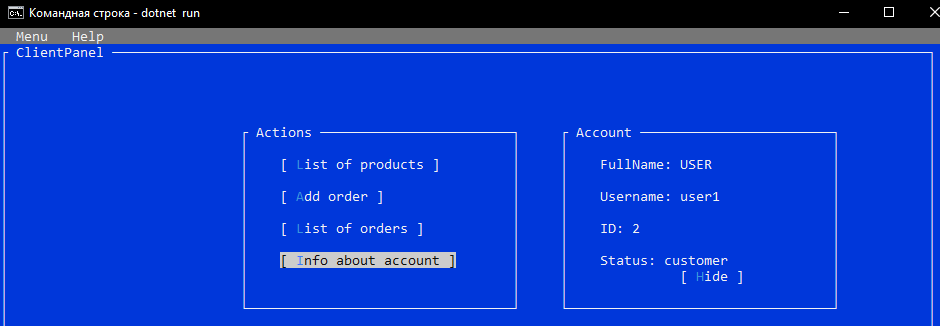
***Рис.2.*** *Діалогове вікно реєстрації користувача*

**

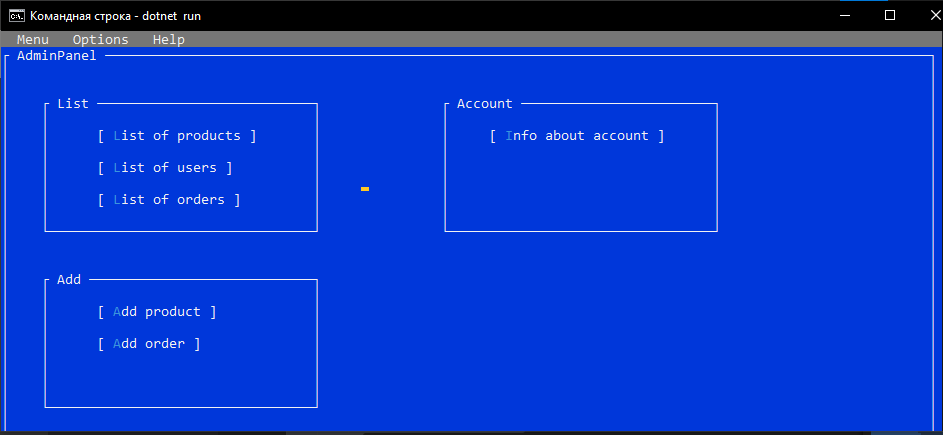
***Рис.3.*** *Головне меню користувача*

**

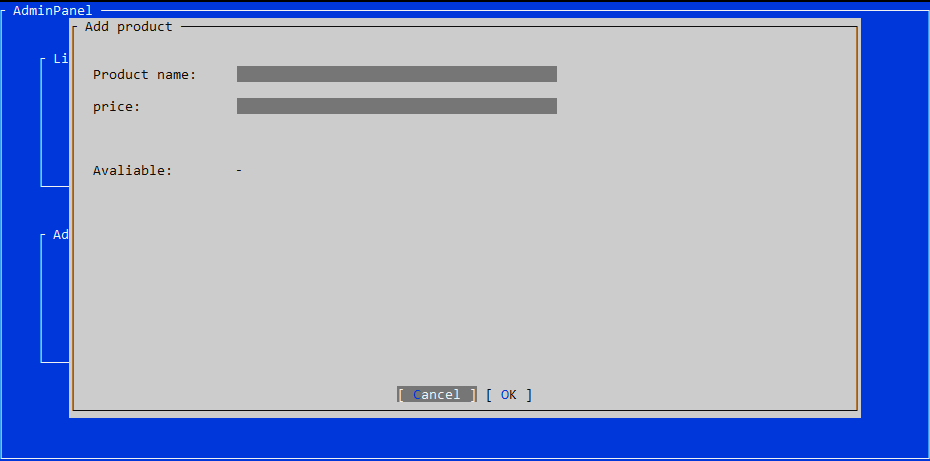
***Рис.4.*** *Список продуктів*

**

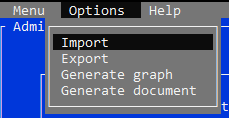
***Рис.5.*** *Інформація про аккаунт користувача*

******

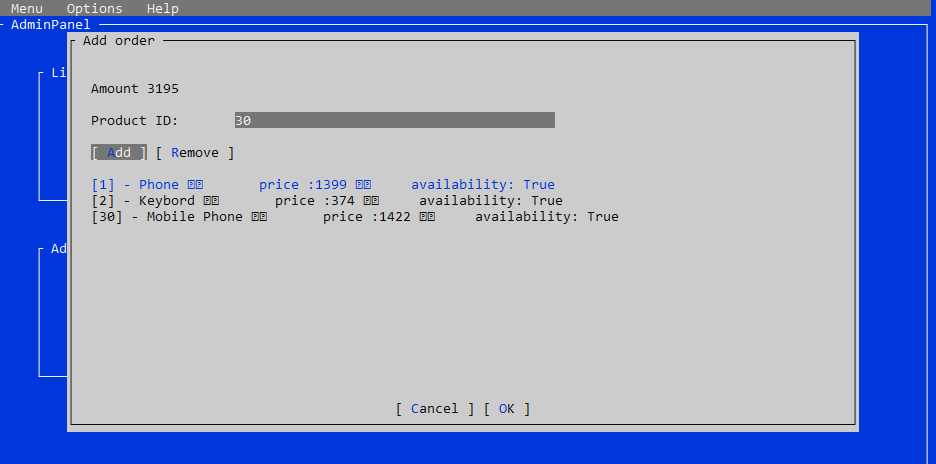
***Рис.6.*** *Головне меню адміністратора*

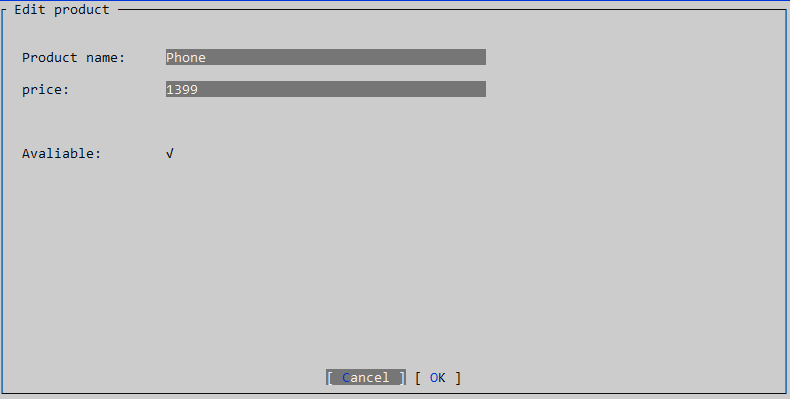
**

***Рис.7.*** *Вікно додавання продукту*

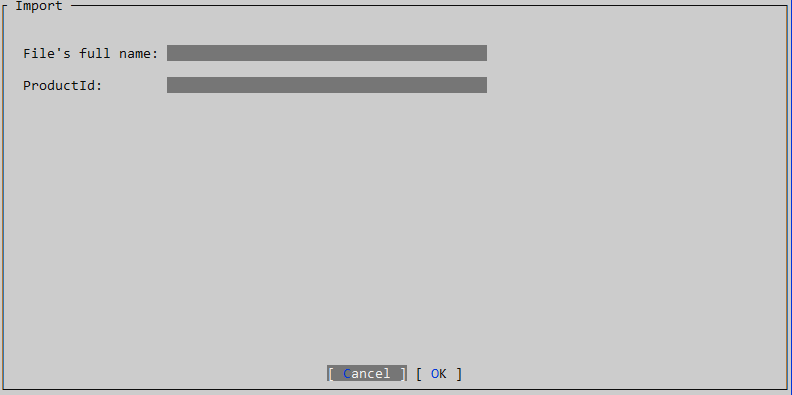
**

***Рис.8.*** *Функції меню “Options”*

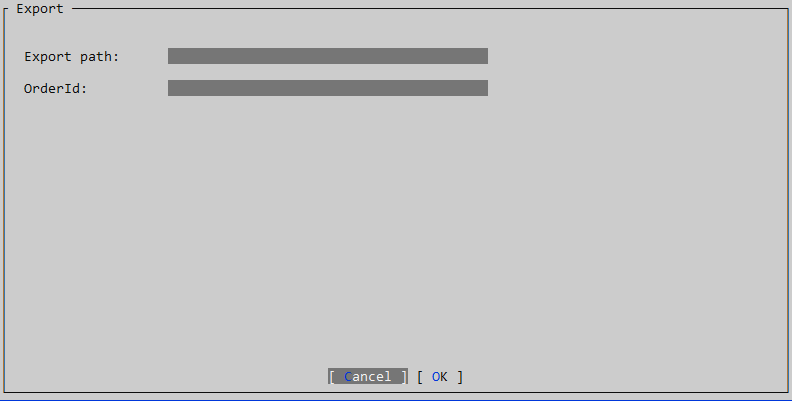
***Рис.9.*** *Вікно додавання замовлення*

**

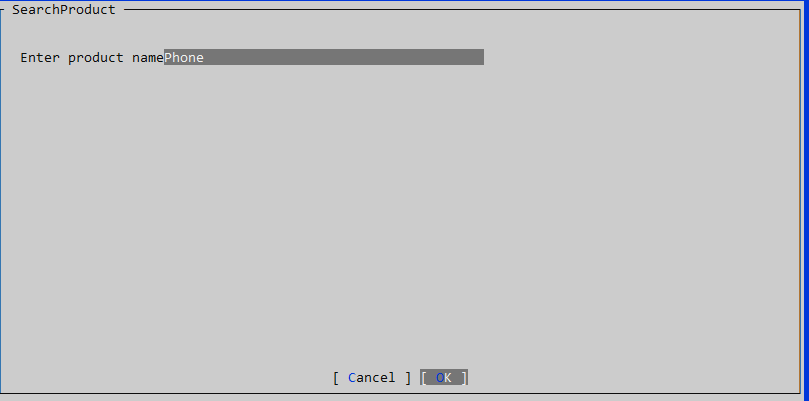
***Рис.10.*** *Вікно редагування продукту*

**

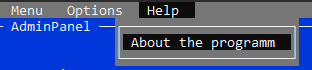
***Рис.11.*** *Вікно імпорту даних*

**

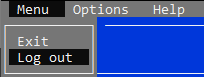
***Рис.12.*** *Вікно експорту даних*

**

***Рис.12.*** *Вікно пошуку даних*



***Рис.13.*** *Функції меню “Help”*

**

***Рис.14.*** *Функції меню “Menu”*

# Висновки

Метою даного курсового проекту було розроблення десктопного додатку з подійно-орієнтованим інтерфейсом користувача - програмного додатку електронного магазину.

Розроблена система:

* забезпечує авторизований доступ до електронного магазину;
* дозволяє створювати замовлення, редагувати, оновлювати та видаляти їх, а також здійснювати відповідні операції для управління своїм профілем;

При реалізації проекту були вдосконалені навички розроблення інтерфейсу користувача та роботи з базою даних Sqlite.

Особливу увагу під час розроблення даного програмного продукту було приділено створенню зручного користувацького середовища, де користувач мав би можливість легко отримувати необхідну інформацію, вводити нові дані для подальшого оброблення та збереження.

Розробка виконана у повному обсязі, всі вимоги враховані, продукт протестовано.

Використання розробленого додатку дозволить створювати замовлення, переглядати додані та збережені дані у базі даних за допомогою пагінового списку, видаляти дані та редагувати їх. Також можна експортувати та імпортувати дані.