**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**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”

спеціальність 121 – Програмна інженерія

на тему: Програмний додаток електронного магазину товарів

(назва теми)

**Студент**

**групи** КП-03 **Павлюшин М. Ю.**

(підпис)

**Викладач**

**к.т.н, доцент кафедри**

**СПіСКС Радченко К.В.**

(підпис)

Київ – 2021

Зміст

**Вступ 3**

[**1. Аналіз предметної галузі 3**](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.b73pd3816f8p)

[1.1. Аналіз вимог до функціональності програмних засобів 3](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.qz6vreit6wm5)

[**2. Аналіз мов програмування та технологій розроблення 5**](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.ewtylcuh8bvc)

[2.1. Мова програмування С# 5](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.fh0j2unhluf2)

[2.2. Фреймворк .Net 5](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.ayd32pcv0jlc)

[2.3. База даних SQLite 5](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.hul0g052l08h)

[2.4. Бібліотека Terminal,Gui 5](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.6jqqk9eelxe)

[2.5. Бібліотека Microsoft.Data.SQLite](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.6jqqk9eelxe) 6

2.6. System.Collections.Generic 6

2.7. Бібліотека System.IO 6

[2.8. Бібліотека System](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.6jqqk9eelxe) 6

2.9. Бібліотека System.Linq 6

[2.10. Бібліотека System.IO.Compression](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.6jqqk9eelxe) 6

[2.11. Бібліотека System.XML.Linq](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.6jqqk9eelxe) 6

[2.12. Бібліотека System.Text](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.6jqqk9eelxe) 6

[2.13. Бібліотека System.Sequrity.Cryptography](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.6jqqk9eelxe) 6

**3. Опис розроблених програмних засобів 7**

[3.1. Загальна структура системи](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.nlu4ckytvy2k) 7

[3.2. Структура бази даних](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.ny4p6qq0v8a9) 8

[3.3. Модулі репозиторіїв даних](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.vkb2811r6l9f) 10

[3.4. Модуль імпорту та експорту](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.dpq6kpdwm35k) 23

[3.5. Модуль аутентифікації](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.7nq5dney5jkd) 25

3.6. Модуль генерації даних 26

3.6. Бібліотека 26

[**4. Аналіз розроблених програмних засобів**](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.g0l8gqqu407h) **28**

[4.1. Особливості реалізації](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.5ajysoaxe2s5) 28

[4.2. Дизайн та вміст вікон](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.c8eyxdz8sp3u) 28

[**Висновки**](https://docs.google.com/document/d/13udNoC-uATtf-lLjJbCEjmSlD841DjKLHfhroFxrJ14/edit#heading=h.6dbqg2jsslr5) **34**

Вступ

В процесі роботи в інтернеті часто виникає необхідність купляти певні товари

Проблемою купівлі товарів в реальному житті є відсутність цих товарів на складі або в магазині. Виникає необхідність створення спеціальної програми, за допомогою якої можна було б робити покупки в інтернеті. Саме тому пропонується розробка програмних засобів, призначених для роботи з великою кількістю даних.

Створення системи, що забезпечує контроль та безпеку цих даних, є актуальною задачею.

Даний проект присвячено розробленню електронного магазину товарів, який призначено для управління товарами, замовленнями користувачів.

1. Аналіз предметної галузі

1.1. Аналіз вимог до функціональності програмних засобів

В процесі аналізу вимог до системи було виділено такі функціональні вимоги Програма має задовольняти такі 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.

a. Можна видаляти дані (з підтвердженням цієї дії).

b. 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.

a. 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).

a. 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. Частина проекту винесена у власні бібліотеки класів.

Винесення саме цього коду обгрунтоване.

2. Аналіз мов програмування та технологій розроблення

2.1. Мова програмування С#  
Серед можливостей мови програмування було використано базові типи даних, цикли, блоки розгалуження. Окрім того описані власні типи даних за допомогою класів і наслідування

2.2. Фреймворк .NET.Net надає модулі взаємодії з базами даних, серіалізації і десереалізації XML, нові типи даних для представлення додаткових потоків виконання і паралельного виконання декількох функцій, представлення сокетів користувачів і сервера у межах персонального комп’ютера, їх підключення і взаємодії

2.3. База даних SQLiteБуло використано базу даних SQLite для зберігання великої кількості об’єктів різних типів даних і їх зв’язків у вигляді записів. Для взаємодії з БД використовувалась мова запитів SQL

2.4. Бібліотека Terminal.Gui(бібліотека, призначена для створення консольних програм за допомогою C #. Структура розроблена для спрощення написання програм, які працюватимуть на монохромних терміналах, а також сучасних кольорових терміналах з підтримкою миші.)

2.5. Бібліотека Microsoft.Data.Sqlite(У цій бібліотеці реалізуються спільні абстракції ADO.NET для підключень, команд, модулів читання даних і інших елементів з таблиць баз даних)

2.6. Бібліотека System.Collections.Generic(Містить інтерфейси і класи, що визначають універсальні колекції, які дозволяють користувачам створювати строго типізовані колекції.)

2.7. Бібліотека System.IO(Містить типи, що дозволяють здійснювати читання і запис у файли і потоки даних, а також типи для базової підтримки файлів і папок)

2.8. Бібліотека System(Містить фундаментальні і базові класи, що визначають часто використовувані типи значень і даних, події і обробники подій, інтерфейси, атрибути і виключення обробки.)

2.9. Бібліотека System.Linq(Надає класи та інтерфейси, що підтримують запити з використанням LINQ.)

2.10. Бібліотека System.IO.Compression(Містить класи, що надають основні служби стиснення і розпакування для потоків.)

2.11. Бібліотека System.Xml.Linq(Містить класи для LINQ to XML. LINQ to XML - це розташований в пам'яті програмний інтерфейс XML, який дозволяє легко і ефективно змінювати документи XML.)

2.12. Бібліотека System.Text(Містить класи, які представляють кодування ASCII і Юнікоду; абстрактні базові класи для перетворення блоків знаків в блоки байтів і назад.)

2.13. Бібліотека System.Security.Cryptography(Надає криптографічні служби, включаючи безпечне кодування і декодування даних, а також безліч інших операцій, таких як хешування, генерація випадкових чисел і перевірка справжності повідомлень.)

3. Опис розроблених програмних засобів

3.1. Загальна структура системи

Програмні засоби реалізовані у вигляді десктопного додатку з графічним інтерфейсом користувача.

Проект розбитий на такі модулі:

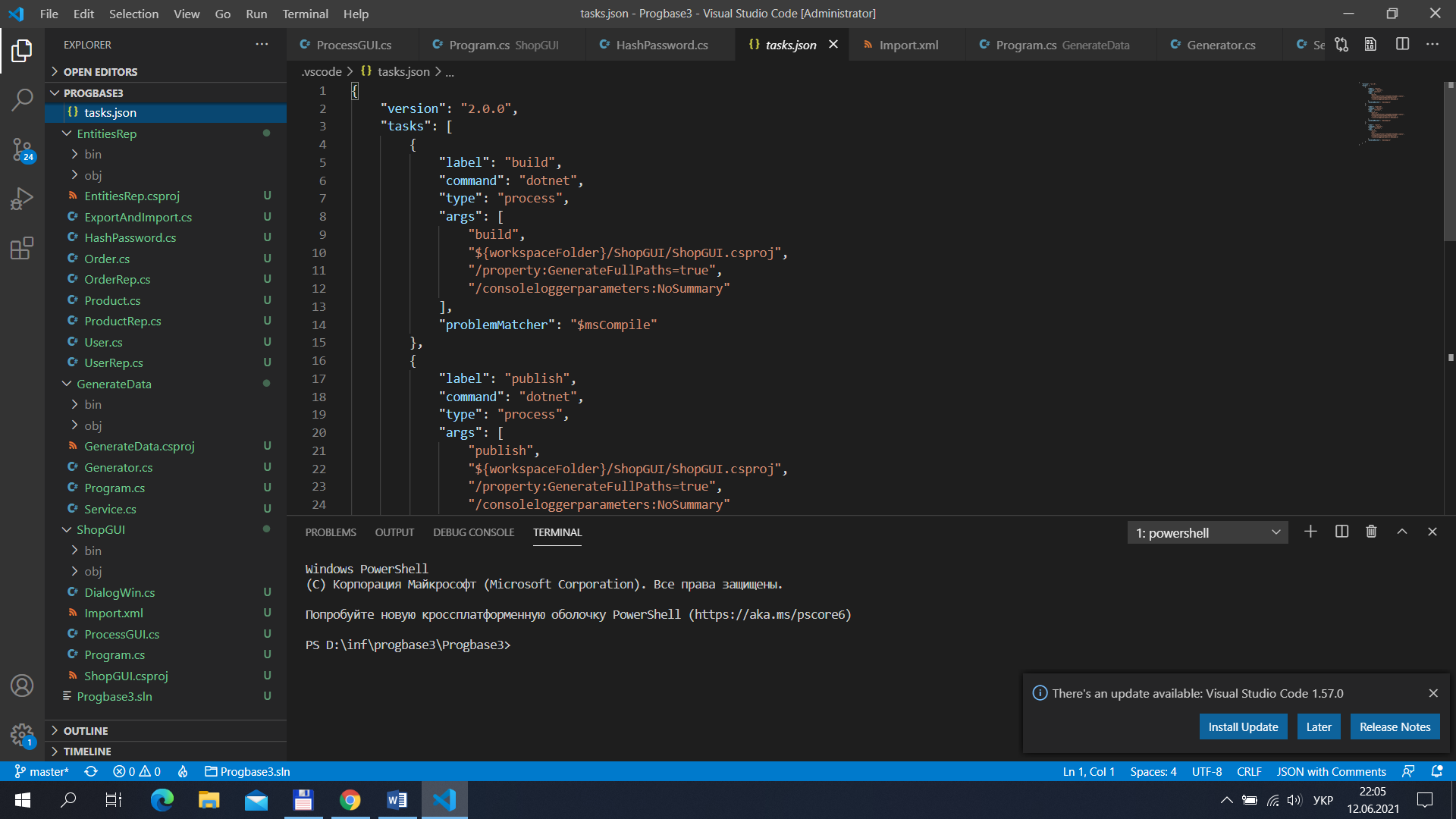
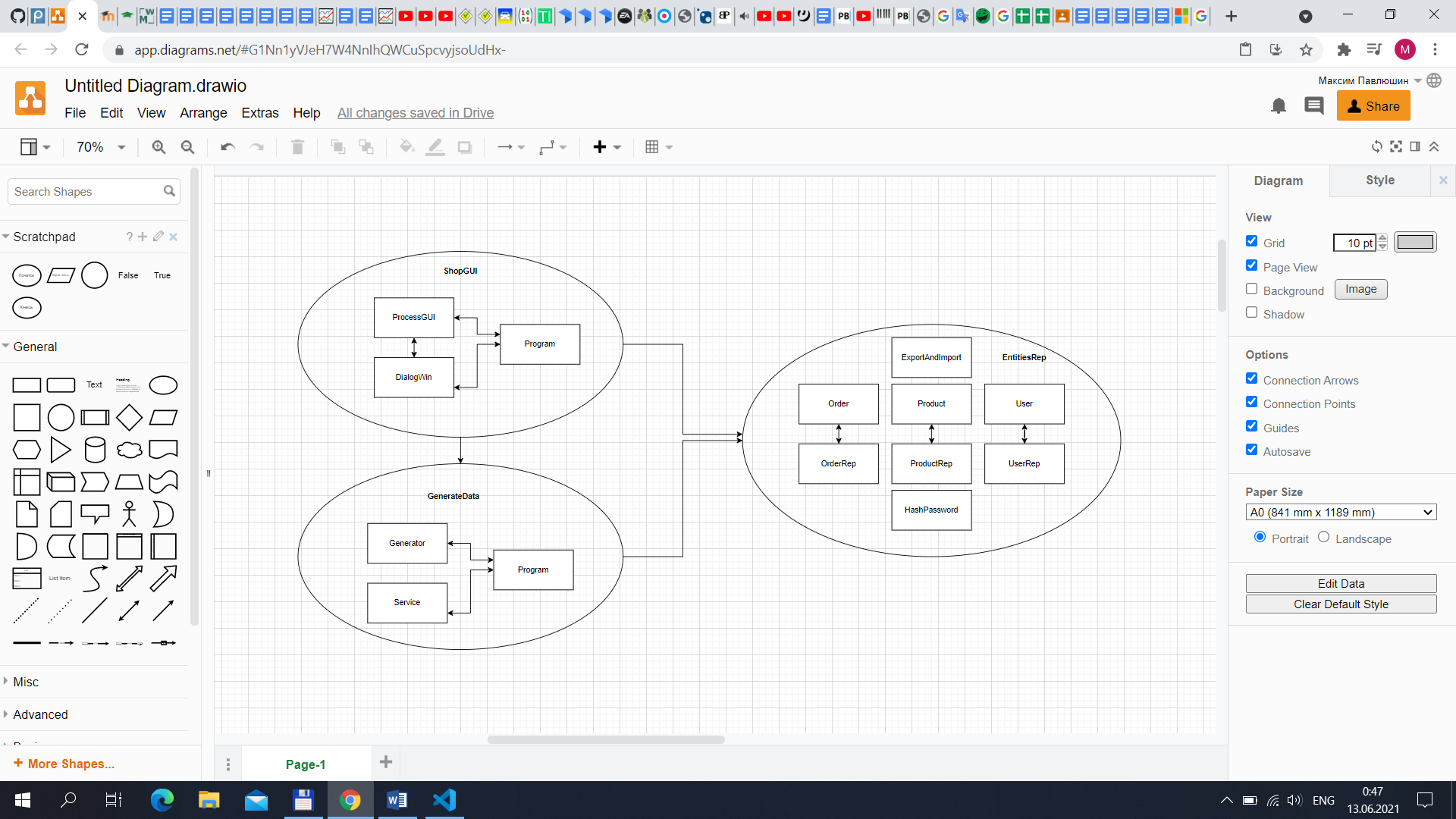


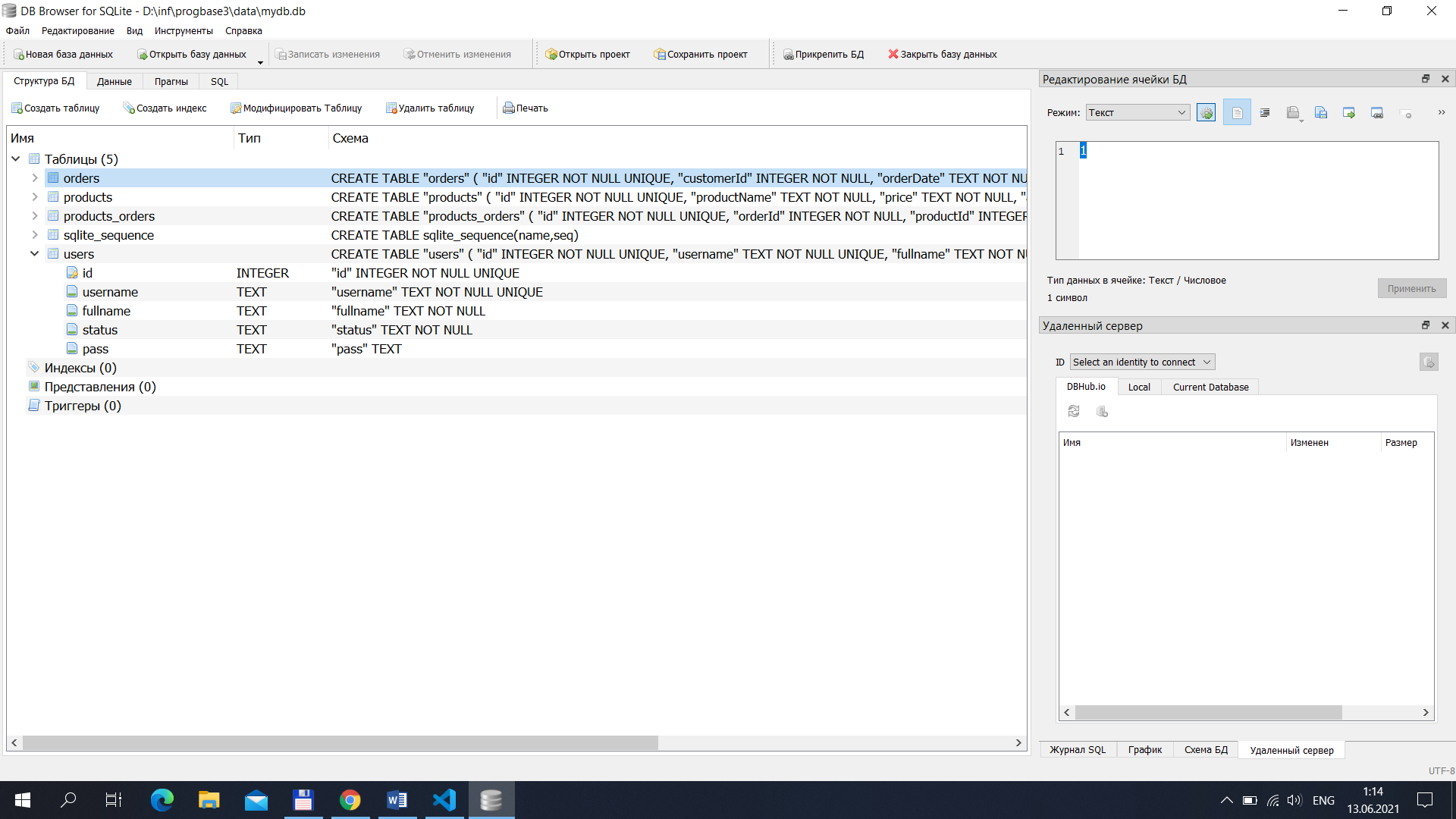
Рис 1. Модулі проекту

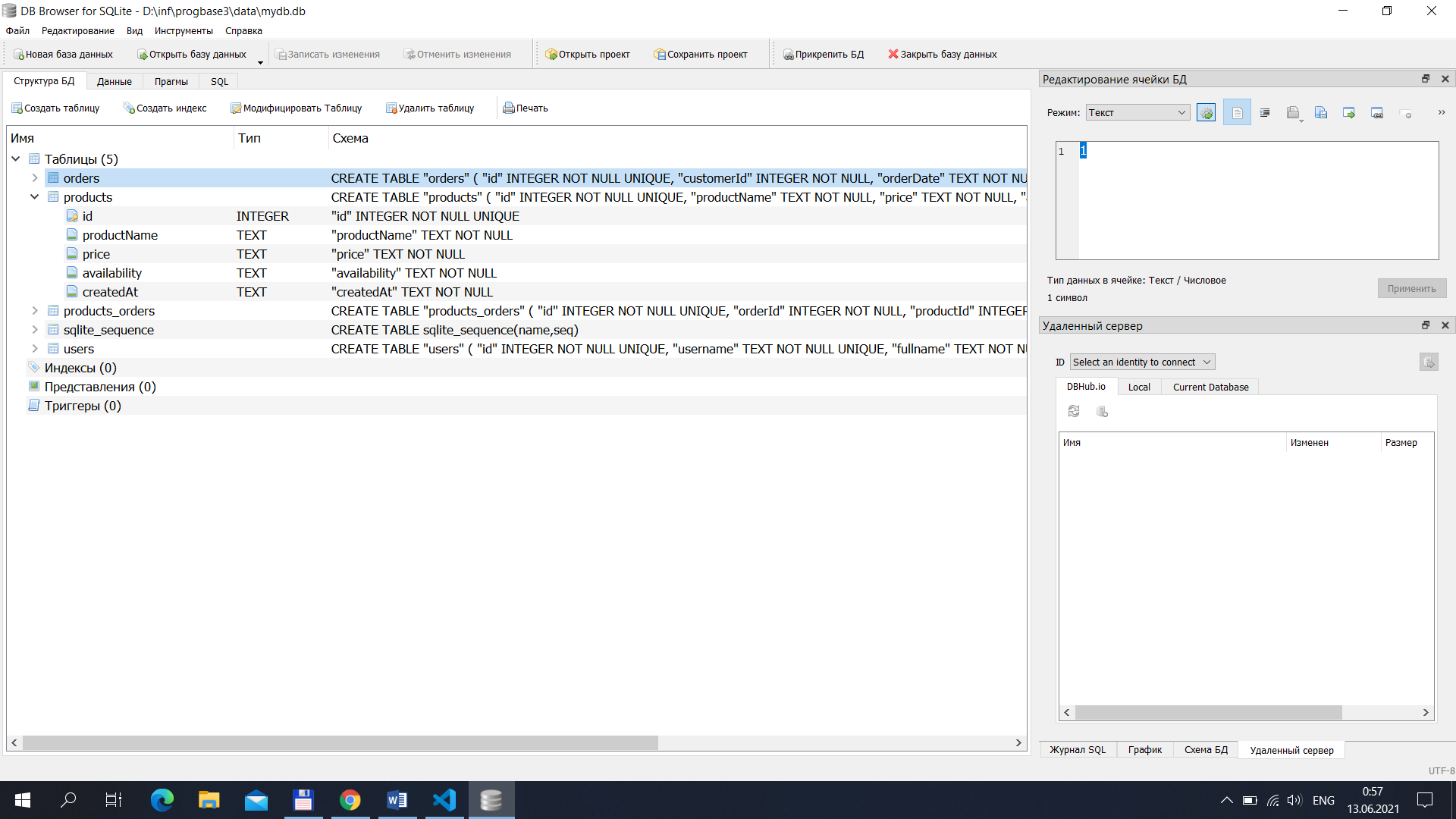
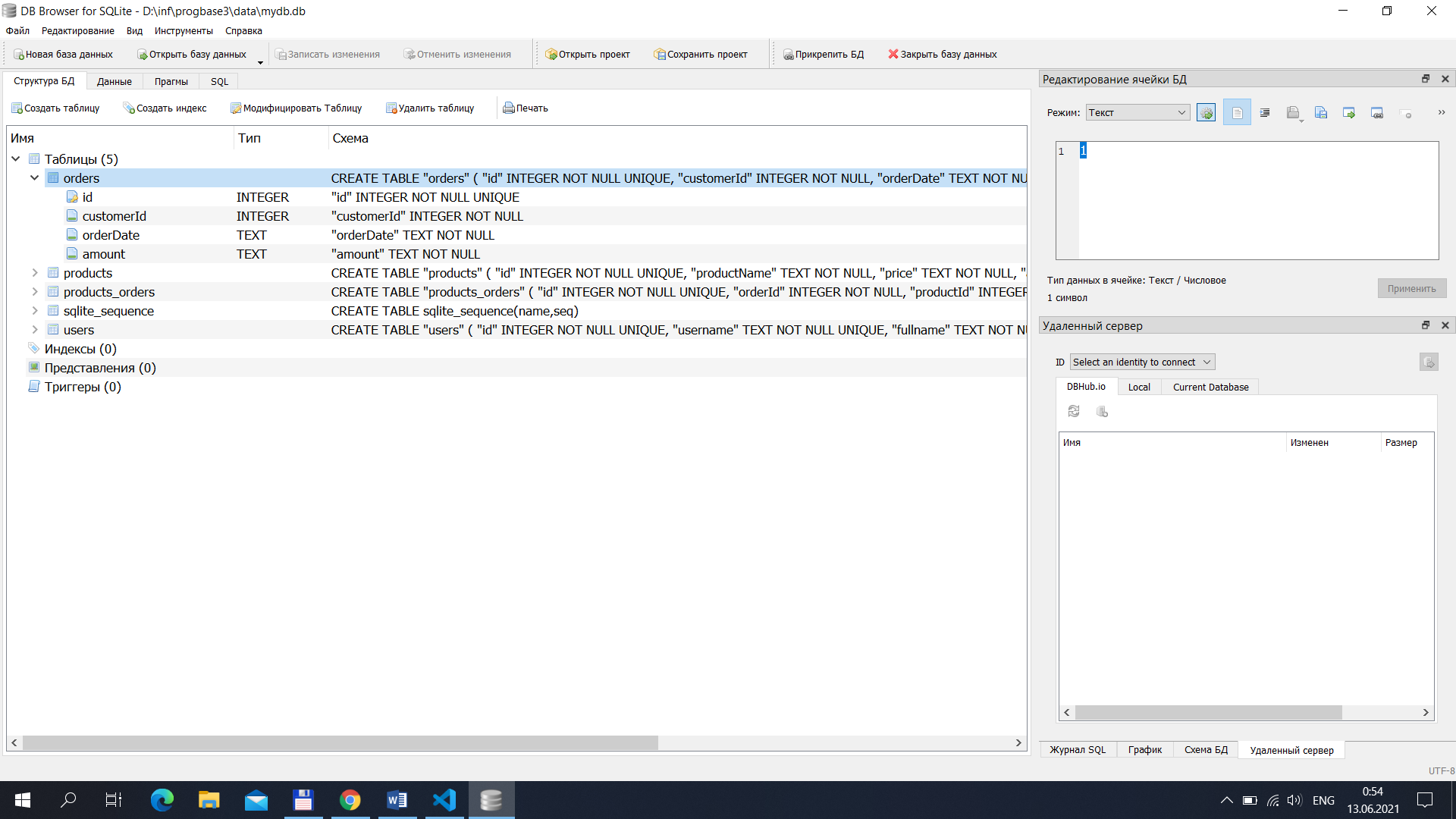
Діаграма залежностей модулів IDE:



3.2. Структура бази даних

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





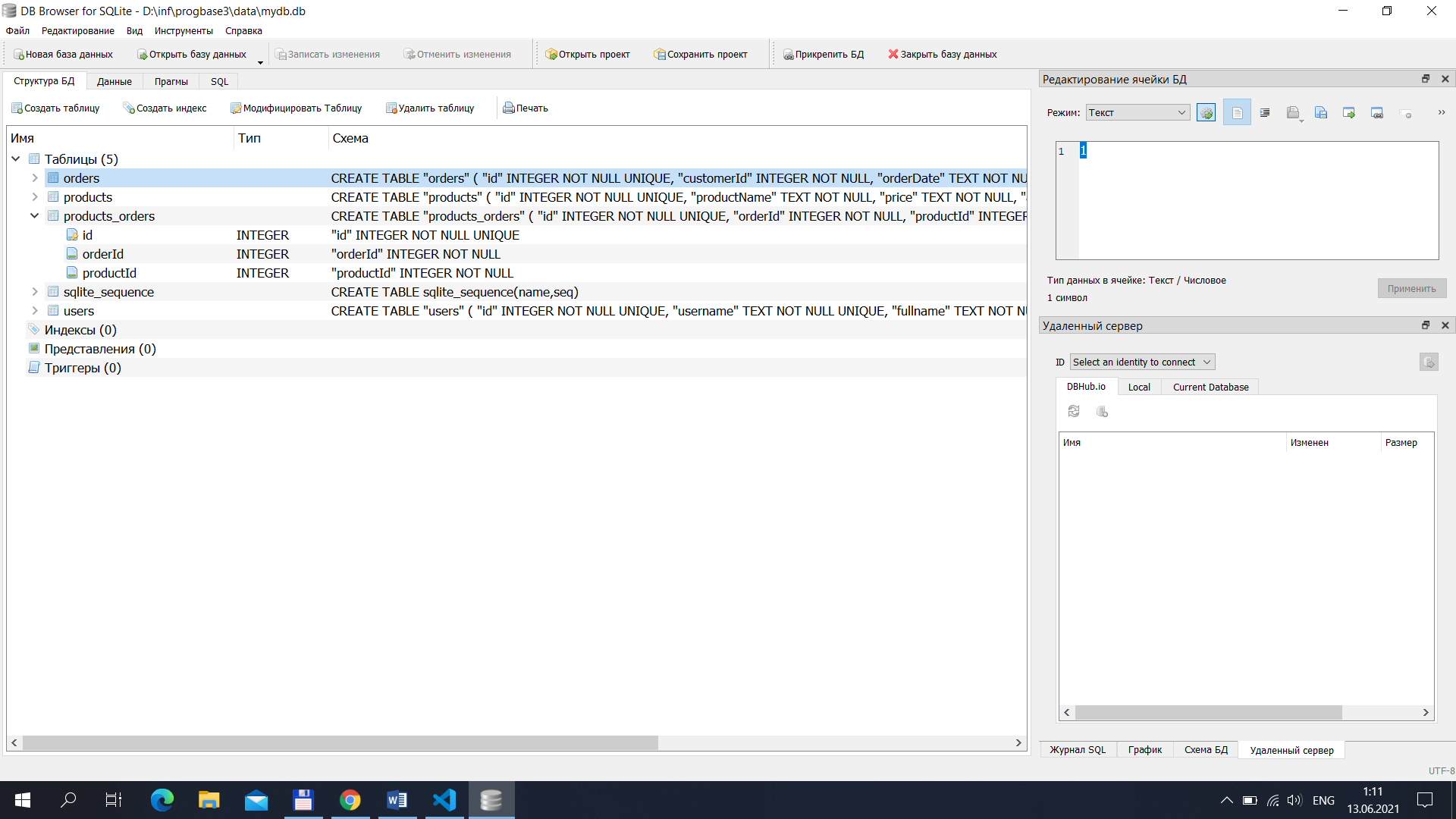


Рис.

Структура бази даних

Таблиці:

users(містить id, псевдонім, повне ім’я, роль та пароль користувачів)

products(містить id, назву, ціну, наявність товару, а також час появи його в базі даних)

orders(містить id замовлення, id замовника, час замовлення)

products\_orders(містить id товару, містить id замовлення)

3.3. Модулі репозиторіїв даних

|  |
| --- |
| **OrderRepository.сs** |
| using System;  using Microsoft.Data.Sqlite;  using System.Collections.Generic;  namespace EntitiesRep  {  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;  }  }  } |

Модуль призначений для керування сутностями типу Order в базі даних

Має такий функціонал:

1.Отримати всі замовлення

2.Отримати замовлення по id

3.Отримати всі id замовлених продуктів

4.Видалити замовлення

5.Редагувати замовлення

5.Отримати id людей, які робили замовлення

|  |
| --- |
| **ProductRepository.сs** |
| using System;  using Microsoft.Data.Sqlite;  using System.Collections.Generic;  namespace EntitiesRep  {  public class ProductRepository  {  private SqliteConnection connection;  public ProductRepository(SqliteConnection connection)  {  this.connection = connection;  }  public bool IsExistProducts()  {  connection.Open();  Console.WriteLine(connection.State);  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT COUNT(\*) FROM products";  SqliteDataReader reader = command.ExecuteReader();  int count = 0;  if (reader.Read() && reader.GetValue(0) != DBNull.Value)  {  count = int.Parse(reader.GetString(0));  }  connection.Close();  if (count == 0)  {  return false;  }  return true;  }  public bool Insert(Product product)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText =  @"INSERT INTO products (productName, price, availability, createdAt)  VALUES ($productName, $price, $availability, $createdAt);  SELECT last\_insert\_rowid();";  command.Parameters.AddWithValue("$productName", product.productName);  command.Parameters.AddWithValue("$price", product.price);  command.Parameters.AddWithValue("$availability", product.availability);  command.Parameters.AddWithValue("$createdAt", product.createdAt.ToString("o"));  long lastId = (long)command.ExecuteScalar();  connection.Close();  if (lastId == 0)  {  return false;  }  return true;  }  public List<Product> GetAllGoods()  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM products";  SqliteDataReader reader = command.ExecuteReader();  List<Product> products = new List<Product>();  while (reader.Read())  {  Product product = GetProduct(reader);  products.Add(product);  }  reader.Close();  connection.Close();  return products;  }  public Product GetProductById(long id)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM products WHERE id = $id";  command.Parameters.AddWithValue("$id", id);  SqliteDataReader reader = command.ExecuteReader();  Product product = new Product();  if (reader.Read())  {  product = GetProduct(reader);  }  connection.Close();  return product;  }  static Product GetProduct(SqliteDataReader reader)  {  Product product = new Product();  product.id = long.Parse(reader.GetString(0));  product.productName = reader.GetString(1);  product.price = int.Parse(reader.GetString(2));  if (int.Parse(reader.GetString(3)) == 1)  { product.availability = true; }  else { product.availability = false; }  product.createdAt = DateTime.Parse(reader.GetString(4));  return product;  }  public int DeleteById(long id)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"DELETE FROM products 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, Product product)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = $"UPDATE products SET productName = $productName, price = $price, availability = $availability, createdAt = $createdAt WHERE id = $id";  command.Parameters.AddWithValue("$id", id);  command.Parameters.AddWithValue("$productName", product.productName);  command.Parameters.AddWithValue("$price", product.price);  command.Parameters.AddWithValue("$availability", product.availability);  command.Parameters.AddWithValue("$createdAt", product.createdAt);  int rowChange = command.ExecuteNonQuery();  connection.Close();  if (rowChange == 0)  {  return false;  }  return true;  }public List<Product> GetOrderProducts(List<long> prodIds)  {  List<Product> prodList = new List<Product>();  connection.Open();  foreach (long id in prodIds)  {  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM products WHERE id = $id";  command.Parameters.AddWithValue("$id", id);  SqliteDataReader reader = command.ExecuteReader();  Product product = new Product();  if (reader.Read())  {  product = GetProduct(reader);  prodList.Add(product);  }  }  connection.Close();  return prodList;  }  public List<Product> GetExportProducts(string substring)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM products WHERE productName LIKE '%' || $value || '%' ";  command.Parameters.AddWithValue("$value", substring);  SqliteDataReader reader = command.ExecuteReader();  List<Product> productsList = new List<Product>();  while (reader.Read())  {  Product product = new Product();  product.id = long.Parse(reader.GetString(0));  product.productName = reader.GetString(1);  product.price = int.Parse(reader.GetString(2));  if (int.Parse(reader.GetString(3)) == 1)  { product.availability = true; }  else { product.availability = false; }  product.createdAt = DateTime.Parse(reader.GetString(4));  productsList.Add(product);  }  reader.Close();  connection.Close();  return productsList;  }  }  } |

Модуль призначений для керування сутностями типу Product в базі даних

Має такий функціонал:

1.Перевірка товару на наявність

2.Отримання всіх товарів

3.Отримання товару по id

4. Видалення товару по id

5.Отримання товарів, які були в замовленні

|  |
| --- |
| **UserRepository.сs** |
| using System;  using Microsoft.Data.Sqlite;  using System.Collections.Generic;  namespace EntitiesRep  {  public class UserRepository  {  private SqliteConnection connection;  public UserRepository(SqliteConnection connection)  {  this.connection = connection;  }  public bool IsExistUsers()  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT COUNT(\*) FROM users";  SqliteDataReader reader = command.ExecuteReader();  int count = 0;  if (reader.Read() && reader.GetValue(0) != DBNull.Value)  {  count = int.Parse(reader.GetString(0));  }  connection.Close();  if (count == 0)  {  return false;  }  return true;  }  public bool Insert(User user)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText =  @"INSERT INTO users (username, fullname, status, pass)  VALUES ($username, $fullname, $status, $pass);  SELECT last\_insert\_rowid();";  command.Parameters.AddWithValue("$username", user.username);  command.Parameters.AddWithValue("$fullname", user.fullname);  command.Parameters.AddWithValue("$status", user.status);  command.Parameters.AddWithValue("$pass", user.pass);  long lastId = (long)command.ExecuteScalar();  connection.Close();  return lastId != 0;  }  public List<User> GetAllUsers()  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM users";  SqliteDataReader reader = command.ExecuteReader();  List<User> users = new List<User>();  while (reader.Read())  {  User user = GetUser(reader);  users.Add(user);  }  reader.Close();  connection.Close();  return users;  }  public long CheckUserName(string name)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM users";  SqliteDataReader reader = command.ExecuteReader();  ;  while (reader.Read())  {  if (name == reader.GetString(1))  {  long lastId = long.Parse(reader.GetString(0));  reader.Close();  connection.Close();  return lastId;  }  }  reader.Close();  connection.Close();  return 0;  }  public User GetUserById(long id)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM users WHERE id = $id";  command.Parameters.AddWithValue("$id", id);  SqliteDataReader reader = command.ExecuteReader();  User user = new User();  if (reader.Read())  {  user = GetUser(reader);  }  reader.Close();  connection.Close();  return user;  }  private User GetUser(SqliteDataReader reader)  {  User user = new User();  user.id = int.Parse(reader.GetString(0));  user.username = reader.GetString(1);  user.fullname = reader.GetString(2);  user.status = reader.GetString(3);  return user;  }  public bool DeleteById(long id)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"DELETE FROM users WHERE id = $id";  command.Parameters.AddWithValue("$id", id);  int nChanged = command.ExecuteNonQuery();  connection.Close();  if (nChanged == 1)  {  return true;  }  return false;  }public bool Update(long id, User user)  {  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = $"UPDATE users SET username = $username, fullname = $fullname, status = $status WHERE id = $id";  command.Parameters.AddWithValue("$id", id);  command.Parameters.AddWithValue("$fullname", user.fullname);  command.Parameters.AddWithValue("$username", user.username);  command.Parameters.AddWithValue("$status", user.status);  int nChanged = command.ExecuteNonQuery();  connection.Close();  if (nChanged == 1)  {  return true;  }  return false;  }  public List<Order> ReadAllUsersOrders(User user)  {  if (user == null)  {  throw new Exception("Not exixsting user.");  }  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM users WHERE id = $id";  command.Parameters.AddWithValue("$id", user.id);  SqliteDataReader reader = command.ExecuteReader();  List<Order> usersOreders = new List<Order>();  while (reader.Read())  {  Order order = OrderRepository.GetOrder(reader);  usersOreders.Add(order);  }  reader.Close();  connection.Close();  return usersOreders;  }  public List<long> GetAllUserOrdersId(long userId)  {  List<long> ordersId = new List<long>();  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT \* FROM orders WHERE customerId = $customerId";  command.Parameters.AddWithValue("$customerId", userId);  SqliteDataReader reader = command.ExecuteReader();  while (reader.Read())  {  long orderId = long.Parse(reader.GetString(0));  ordersId.Add(orderId);  }  connection.Close();  return ordersId;  }  public bool CheckUserInDb(string username, string pass2)  {  string pass = HashPassword.HashCode(pass2);  connection.Open();  SqliteCommand command = connection.CreateCommand();  command.CommandText = @"SELECT count(rowid) username, pass  FROM users  WHERE username = $username AND pass = $pass";  command.Parameters.AddWithValue("$username", username);  command.Parameters.AddWithValue("$pass", pass);  long index = (long)command.ExecuteScalar();  connection.Close();  if (index == 1)  {  return true;  }  else  {  return false;  }  }  }  } |

Модуль призначений для керування сутностями типу User в базі даних

Має такий функціонал:

1.Додавання нового користувача

2.Перевірка на наявність користувача

3.Повернення кількості користувачів

4.Отримання користувача по id

5.Видаляти користувача

6.Знайти користувача по імені

3.4. Модуль імпорту та експорту

|  |
| --- |
| **ImportAndExport.сs** |
| using System;  using System.Collections.Generic;  using System.IO;  using System.IO.Compression;  using System.Xml.Linq;  namespace EntitiesRep  {  public static class ImportAndExport  {  public static void Export(long id, string path, OrderRepository orderRep, ProductRepository productRep)  {  new DirectoryInfo(path + "/Export").Create();  Order order = orderRep.GetOrderById(id);  XDocument doc = new XDocument();  XElement element = new XElement($"Order\_{id}", new List<XAttribute>()  {  new XAttribute("ID",order.id),  new XAttribute("CustomerID",order.customerId),  new XAttribute("OrderDate",order.orderDate),  new XAttribute("Amount",order.amount)  });  doc.Add(element);  doc.Save(path + "/Export/ExportOrder.xml");  doc = new XDocument();  element = new XElement("Products");  foreach (long l in orderRep.GetProductIds(order))  {  Product p = productRep.GetProductById(l);  XElement product = new XElement($"{p.productName.Replace(" ", "\_").Replace("'", "")}");  XAttribute xId = new XAttribute("ID", p.id);  XAttribute xPrice = new XAttribute("Price", p.price);  XAttribute xAvailability = new XAttribute("Availability", p.availability);  XAttribute xData = new XAttribute("Created\_At", p.createdAt);  product.Add(xId, xPrice, xAvailability, xData);  element.Add(product);  }  doc.Add(element);  doc.Save(path + "/Export/ExportProducts.xml");  ZipFile.CreateFromDirectory(path + "/Export", path + "/Export.zip");  }  public static void Import(long id, string fullName, OrderRepository orderRep, out bool error)  {  try  {  XDocument doc = XDocument.Load(fullName);  XElement el = doc.Element("Connections");  foreach (XElement element in el.Elements())  {  if (element.Attribute("ProductID").Value == id.ToString())  orderRep.AddProductConection(id, Convert.ToInt64(element.Attribute("OrderID").Value));  }  error = false;  }  catch  {  error = true;  }  }  }  } |

Модуль призначений для імпорту та експорту товарів

Має такий функціонал:  
1.Імпортувати товари у базу даних з xml файлу  
2. Експортувати товари у xml файл з бази даних

3.5. Модуль аутентифікації

|  |
| --- |
| **HashPassword.сs** |
| using System;  using System.Security.Cryptography;  using System.Text;  namespace EntitiesRep  {  public static class HashPassword  {  public static string HashCode(string pass)  {  SHA256 sha256Hash = SHA256.Create();  string hash = GetHash(sha256Hash, pass);  sha256Hash.Dispose();  return hash;  }  private static string GetHash(HashAlgorithm hashAlgorithm, string input)  {  byte[] data = hashAlgorithm.ComputeHash(Encoding.UTF8.GetBytes(input));  var sBuilder = new StringBuilder();  for (int i = 0; i < data.Length; i++)  {  sBuilder.Append(data[i].ToString("x2"));  }  return sBuilder.ToString();  }  }  } |

Модуль призначений для хешування паролів

Має такий функціонал:  
1.Хешує введений користувачем пароль для того, щоб захистити дані від зловмисників

3.6. Модуль генерації даних

|  |
| --- |
| **Generator.сs** |
| using System;  using EntitiesRep;  using Microsoft.Data.Sqlite;  namespace GenerateData  {  static class Generator  {  public static void GenerateCargos(int number, SqliteConnection connection)  {  string[] cargosArr = new string[] { "Mi Note 10 Lite", "Redmi 6A", "Mi8", "Pocophone f2 pro", "Mi 8 Lite", "Redmi 9", "Redmi Note 10 5G", "Redmi Note 10 Pro",  "Mi 11 Lite", "Mi 11i", "Redmi Note 10", "Black Shark 4", "Black Shark 3", "Mi 10T Lite", "Poco F3", "Redmi Note 9 Pro", "Mi 11", "Redmi Note 10S", "Mi 11 Ultra" };  string[] boolean = new string[] { "true", "false" };  Random random = new Random();  ProductRepository productsRep = new ProductRepository(connection);  for (int i = 0; i < number; i++)  {  Product newGood = new Product();  newGood.productName = cargosArr[random.Next(0, cargosArr.Length - 1)];  newGood.price = random.Next(8500, 26000);  newGood.availability = Boolean.Parse(boolean[random.Next(0, 2)]);  newGood.createdAt = DateTime.Now;  productsRep.Insert(newGood);  }  }  public static void GenerateUsers(int number, SqliteConnection connection)  {  string[] names = new string[] { "Cristiano", "Lionel", "Neymar", "Paul", "Alexis", "Marcus", "David", "Victor", "Phil", "Bruno", "Thomas", "Manuel", "Mark-Andre", "Karim", "Lorenzo", "Anthony", "Rud", "Marco", "Kevin", "Raheem", "Sergio", "Erling", "Kylian", "Bernardo", "Mayson" };  string[] lastNames = new string[] { "Ronaldo", "Messi", "Junior", "Pogba", "Sanchez", "Rashford", "De Gea", "Lindelof", "Jones", "Fernandes", "Neuer", "Ter Stegen", "Benzema", "Insigne", "Martial", "Gullit", "Verrati", "De Bruyne", "Sterling", "Haaland", "Mbappe", "Silva", "Greenwood" };  Random random = new Random();  UserRepository usersRep = new UserRepository(connection);  for (int i = 0; i < number; i++)  {  User newUser = new User();  newUser.fullname = names[random.Next(0, names.Length - 1)] + " " + lastNames[random.Next(0, lastNames.Length - 1)];  newUser.username = "user" + i.ToString();  newUser.status = "customer";  usersRep.Insert(newUser);  }  }  }  } |

Модуль призначений для генерації даних

Має такий функціонал:  
1.Генерує n товарів  
2.Генерує n користувачів

3.7. Бібліотека EntitiesRep

У даній бібліотеці класів зберігаються класи сутностей, репозиторії, модулі експорту/імпорту та аутентифікації даних

Консольні проекти використовують модулі даної біблітеки для певних задач і процесів

4. Аналіз розроблених програмних засобів

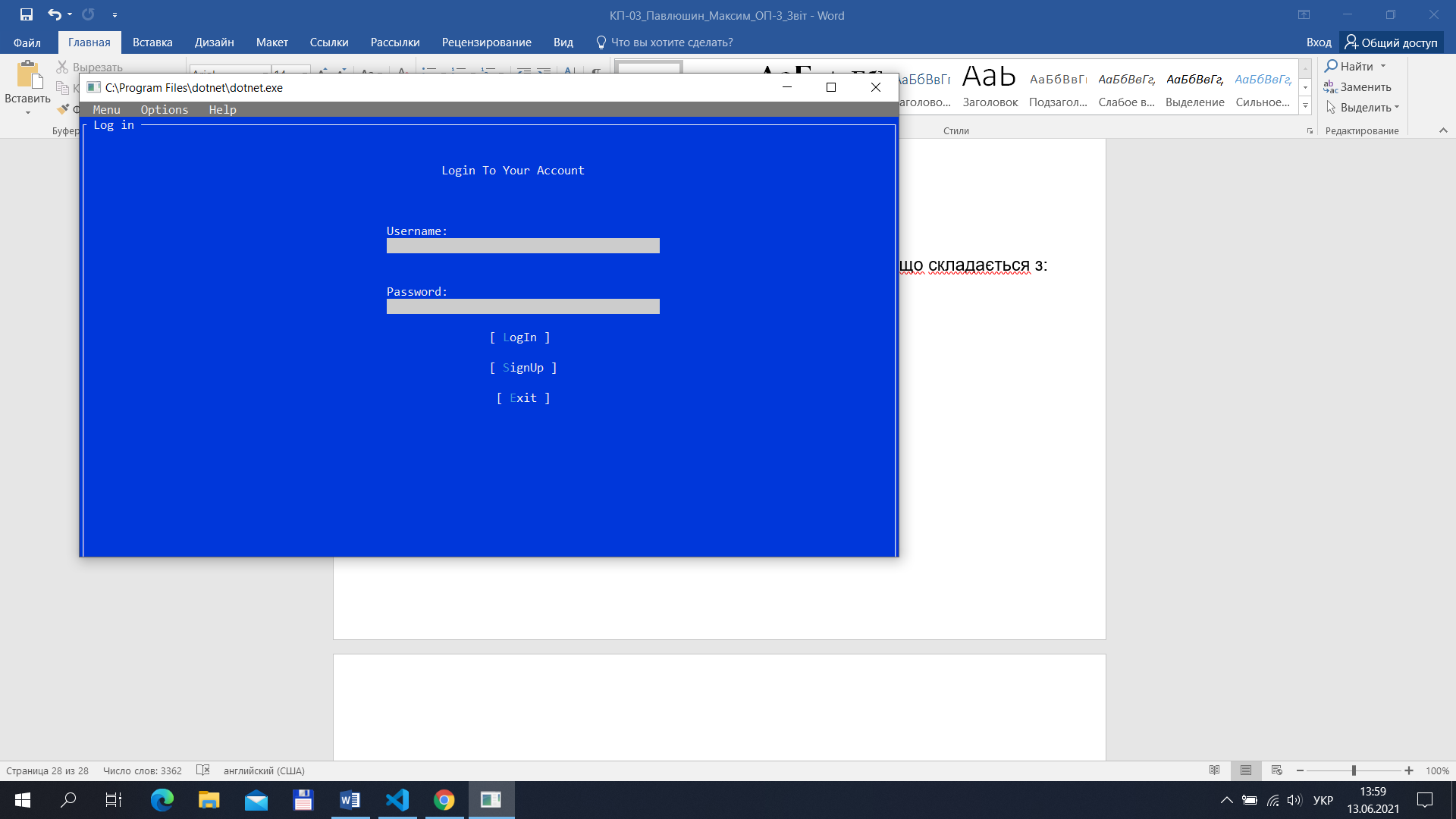
4.1. Особливості реалізації

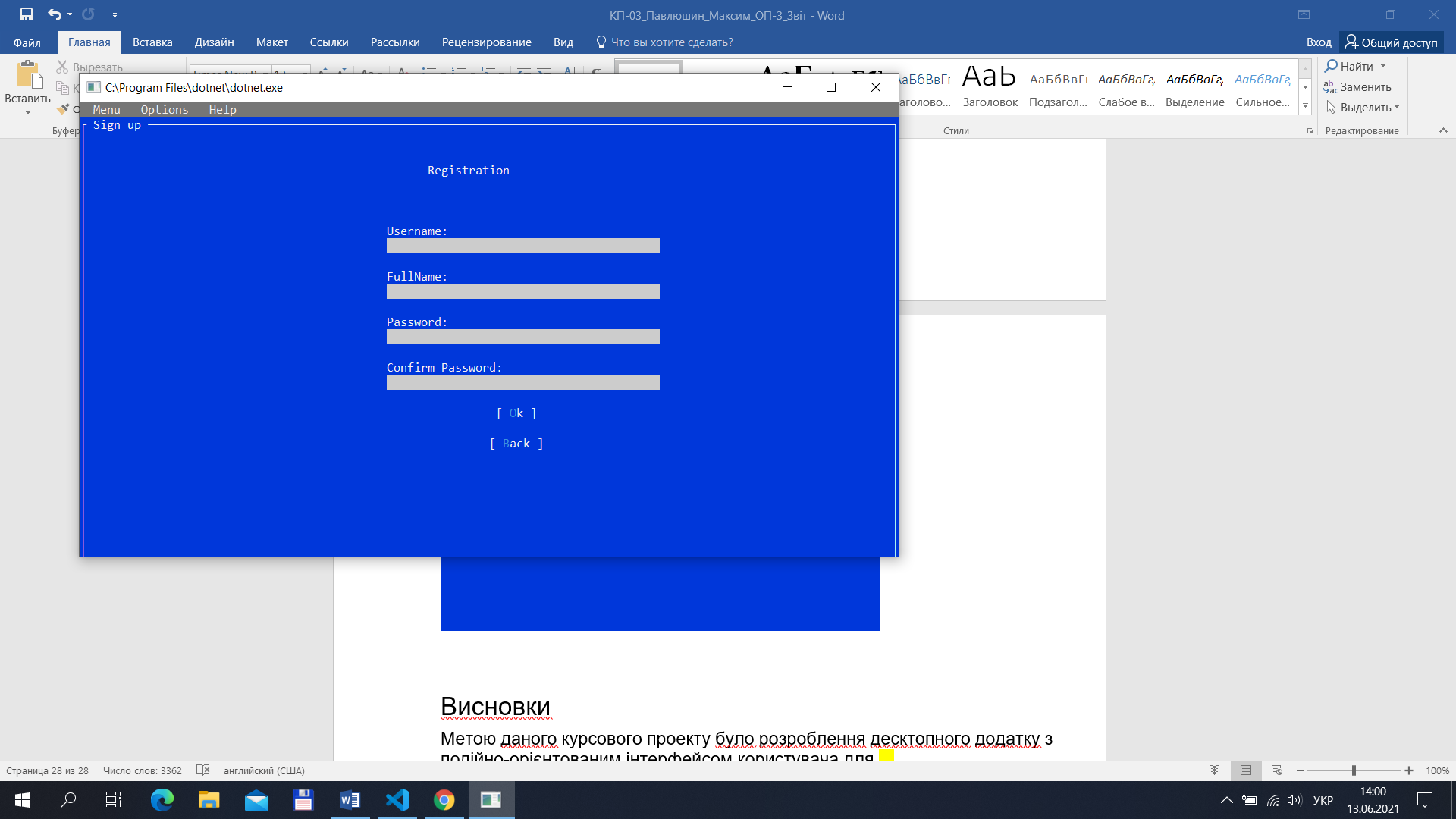
Особливістю авторизації користувачів є те, що вони не можуть мати однакових псевдонімів. Юзери мають дві ролі: адмін і користувач. Власне корустувачі можуть переглянути список товарів, зробити замовлення, знайти товар. Крім попередньо зазначених дій адміни можуть додати товар, переглянути список товарів, користувачів та юзерів. Паролі хешуються по алгоритму SHA-256 для того, щоб їх не можна було переглянути в базі даних. Також є можливість імортувати дані з xml файлу або експортувати дані в xml файл. Серіалізація — процес перетворення будь-якої структури даних у послідовність бітів. Зворотною до операції серіалізації є операція десеріалізації — відновлення початкового стану структури даних із бітової послідовності.

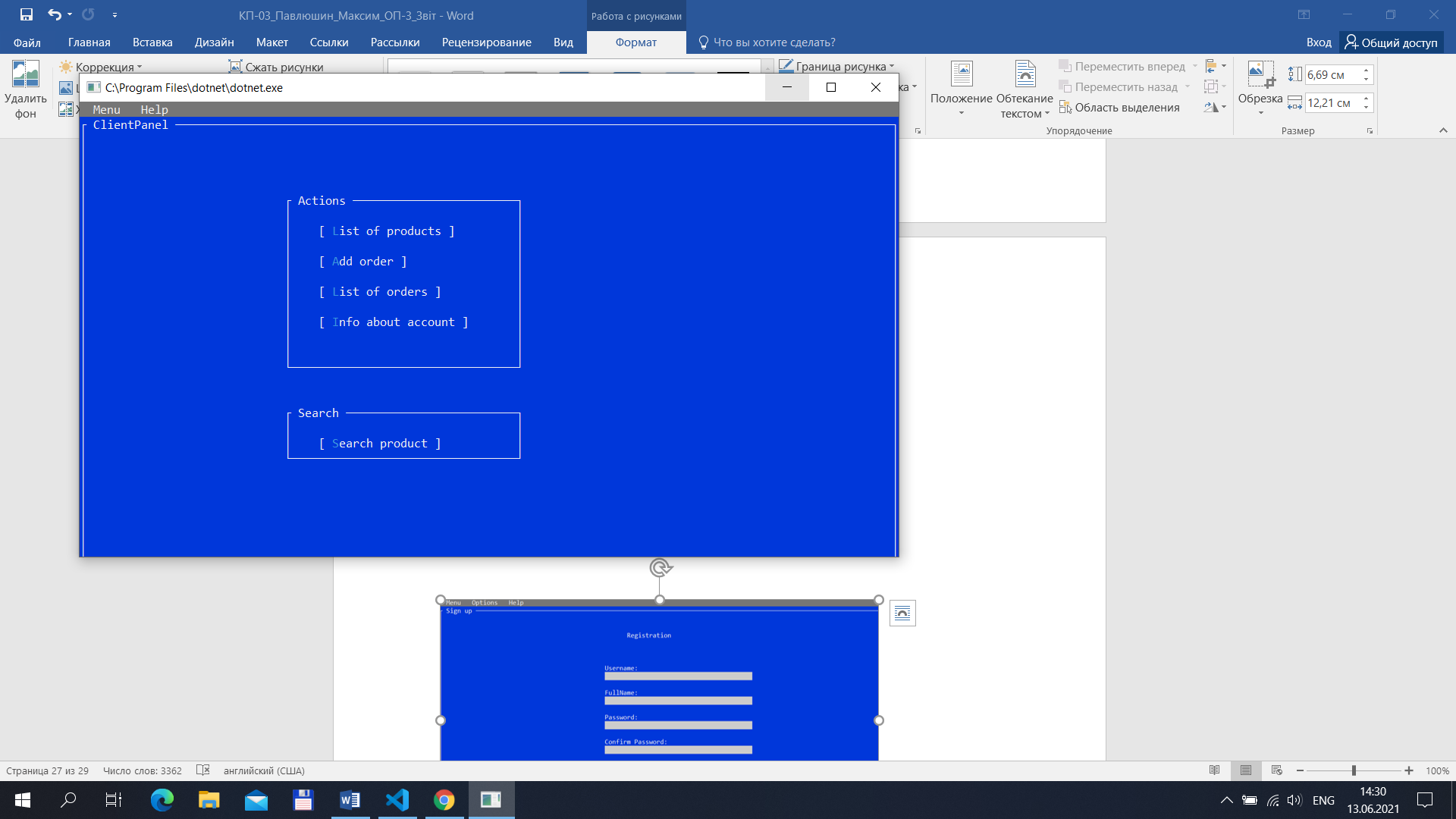
4.2. Інтерфейс користувача

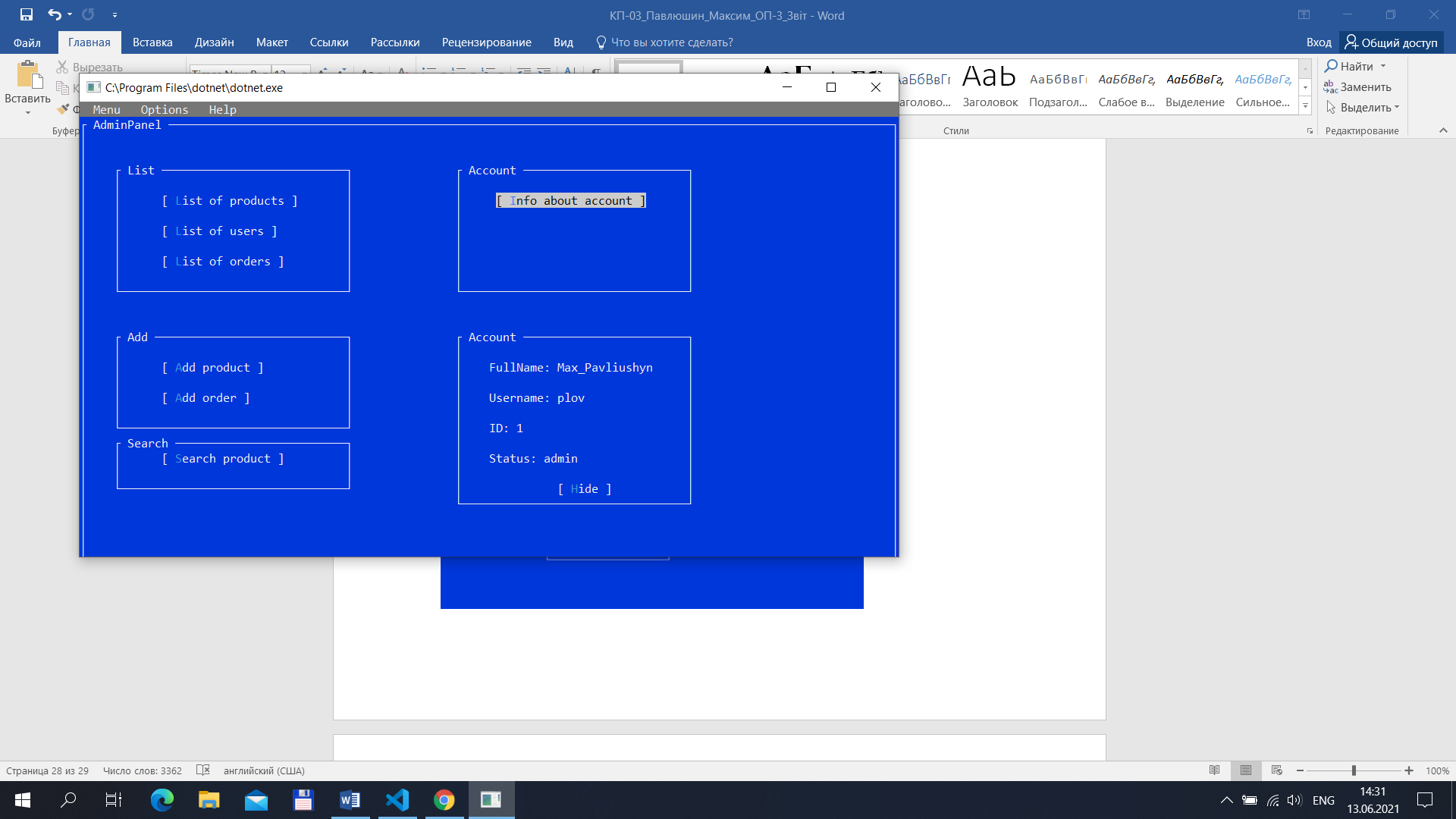
Для показу даних система має інтерфейс користувача, що складається з:

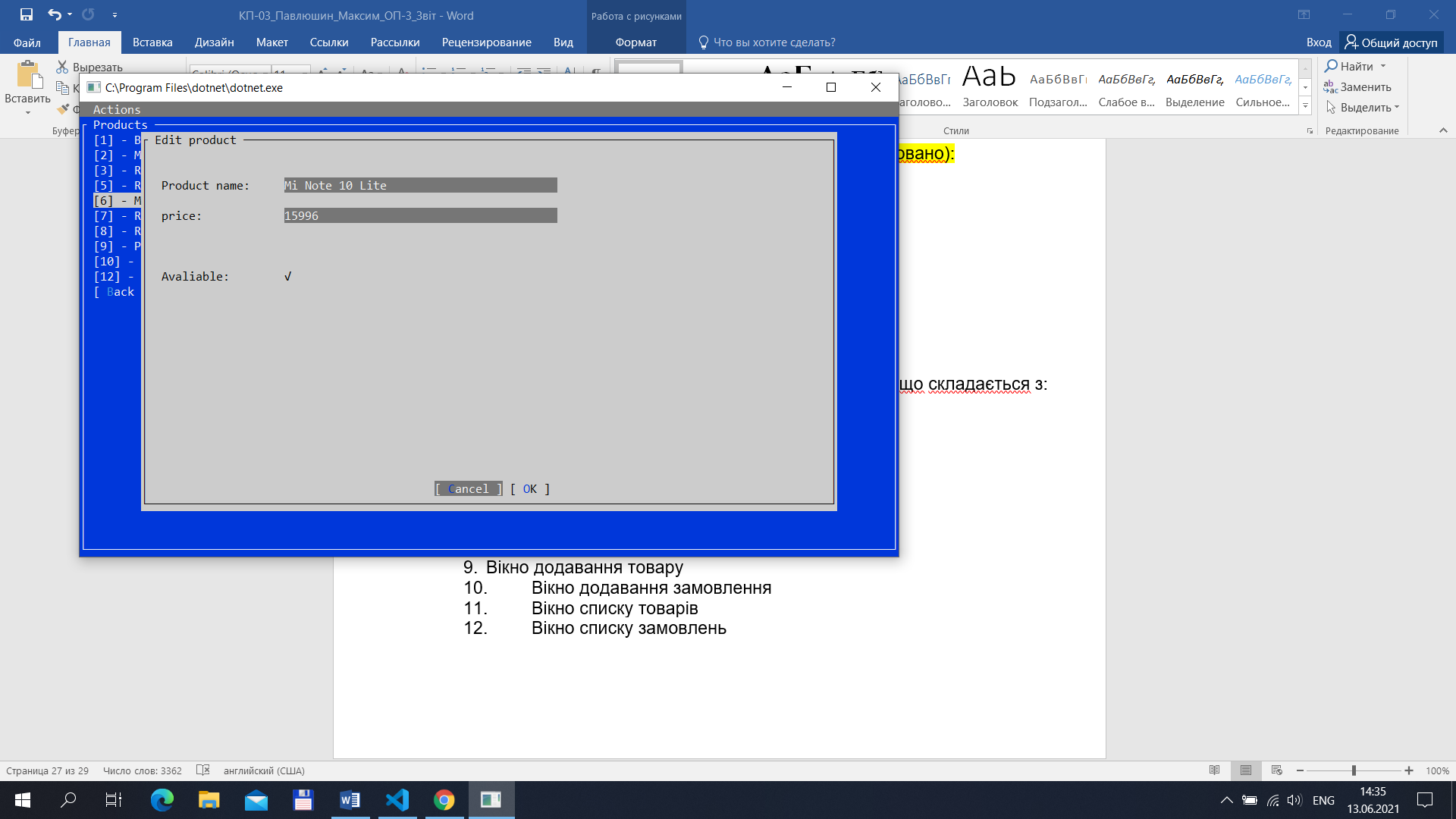
1. Вікно аутентифікації
2. Вікно реєстрації
3. Вікно користувача
4. Вікно адміна
5. Вікно редагування товару
6. Вікно імпорту
7. Вікно експорту
8. Вікно пошуку товару
9. Вікно додавання товару
10. Вікно додавання замовлення
11. Вікно списку товарів
12. Вікно списку замовлень
13. Вікно списку користувачів

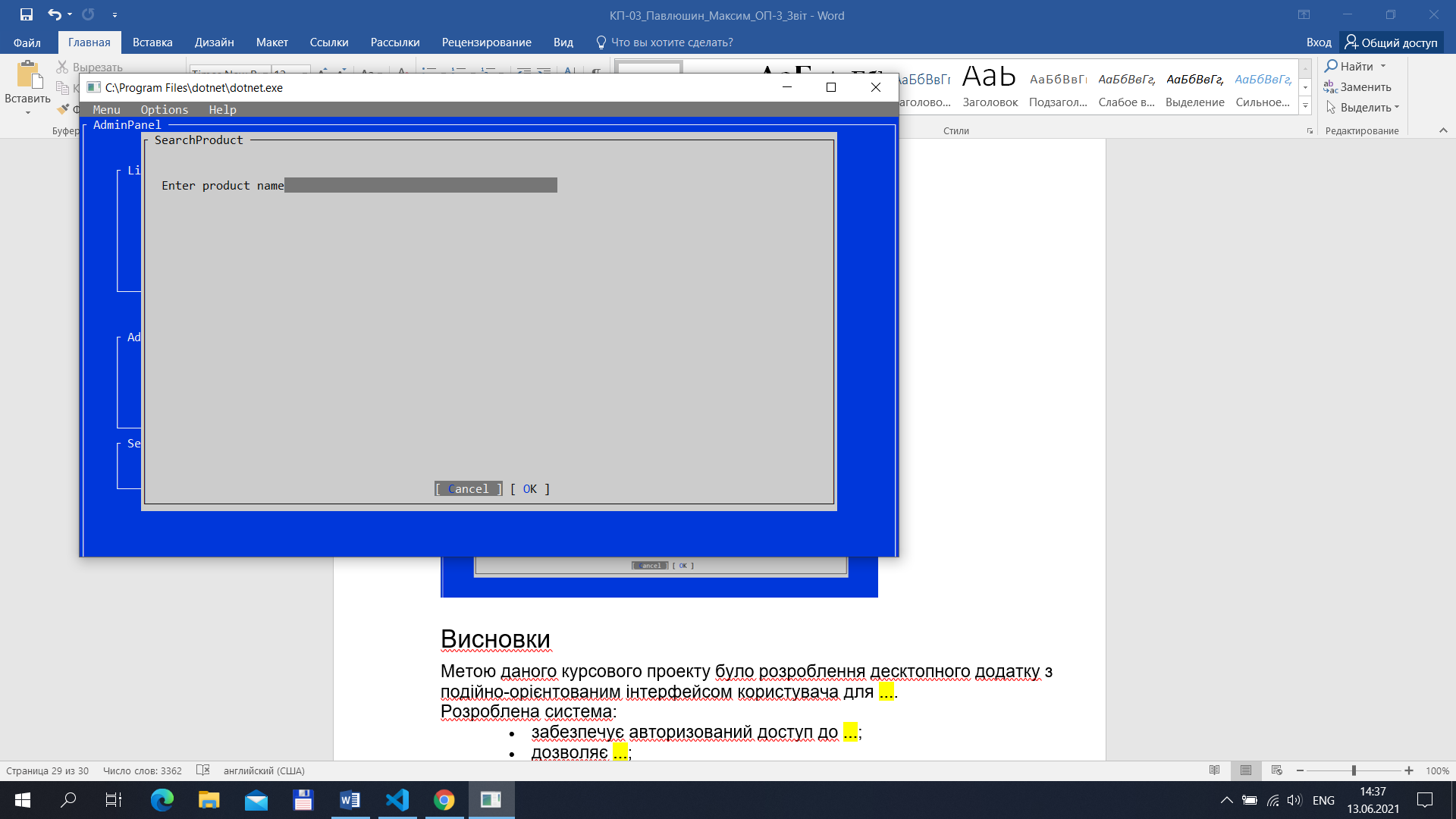
Вхід під вже існуючим іменем і паролем

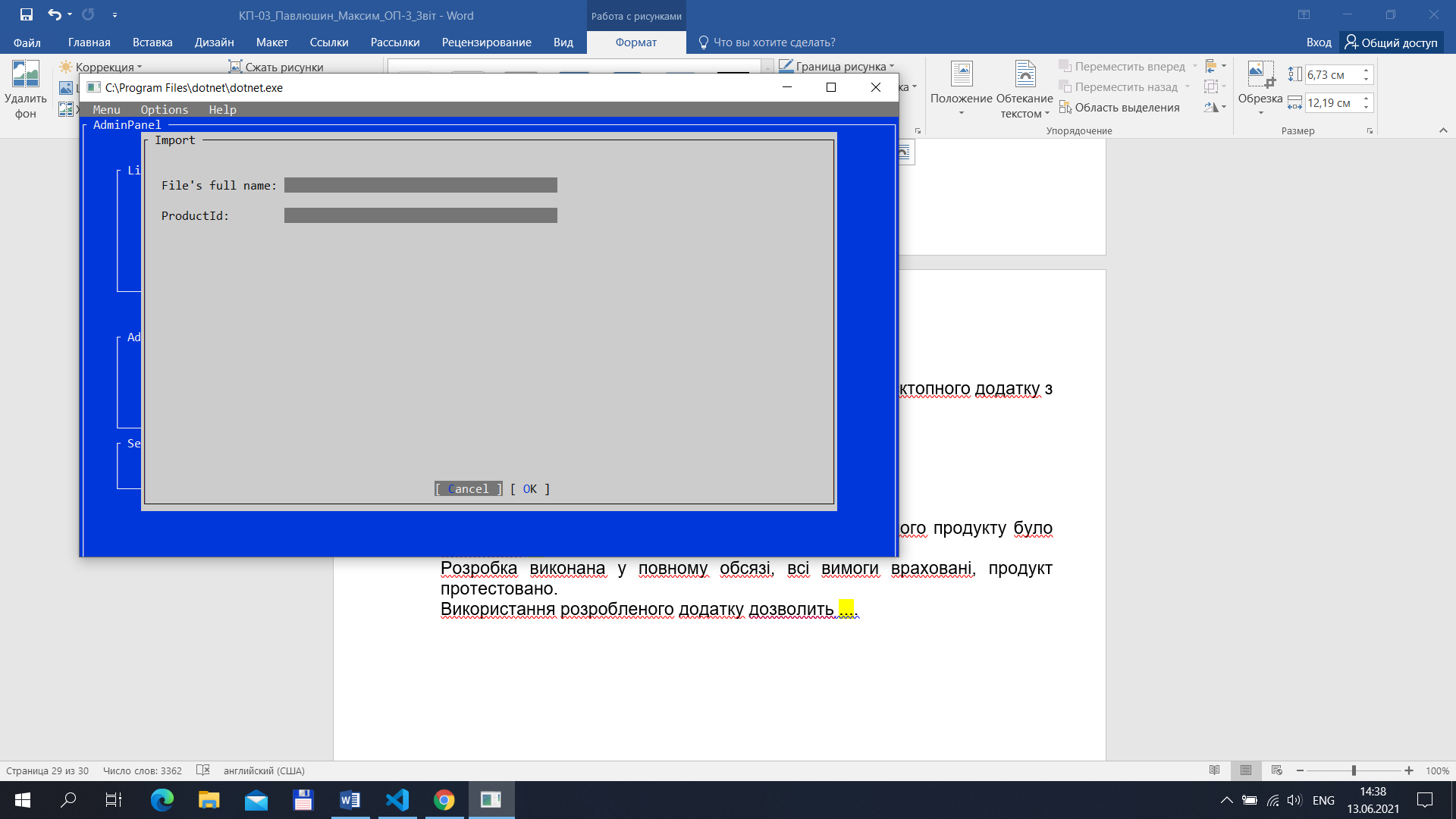
Реєстрація нового користувача

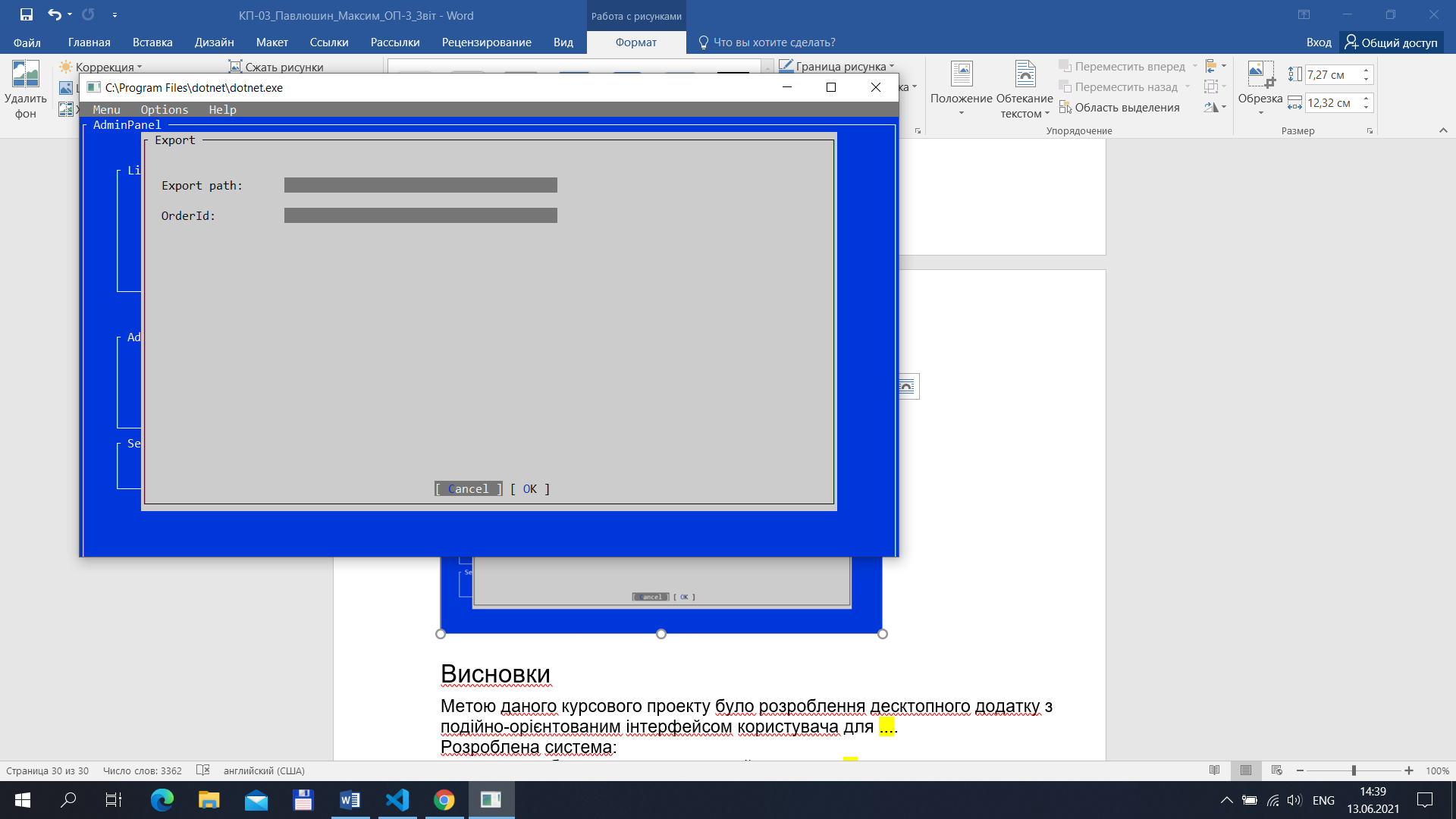
Панель користувача

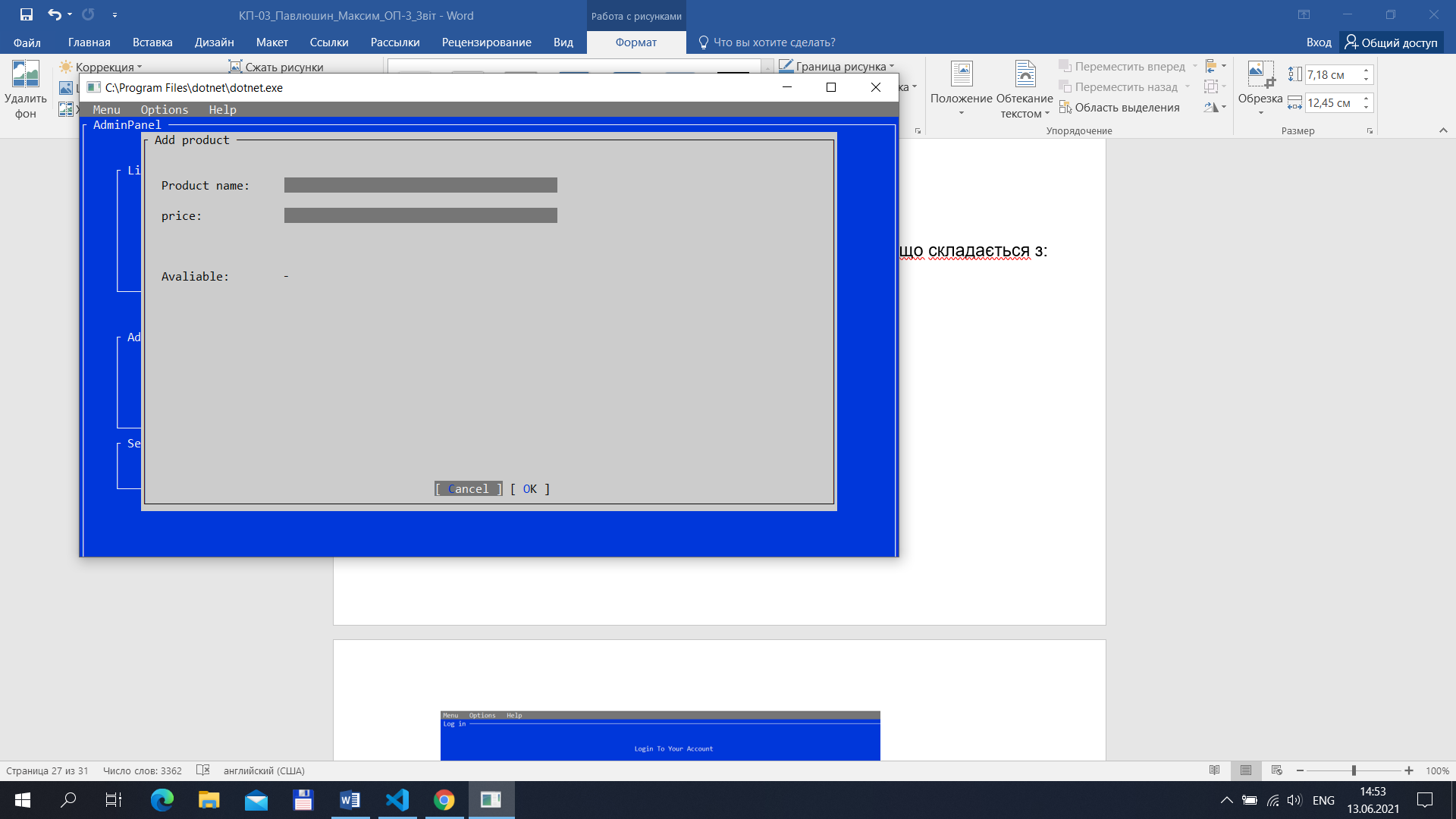
Панель адміна

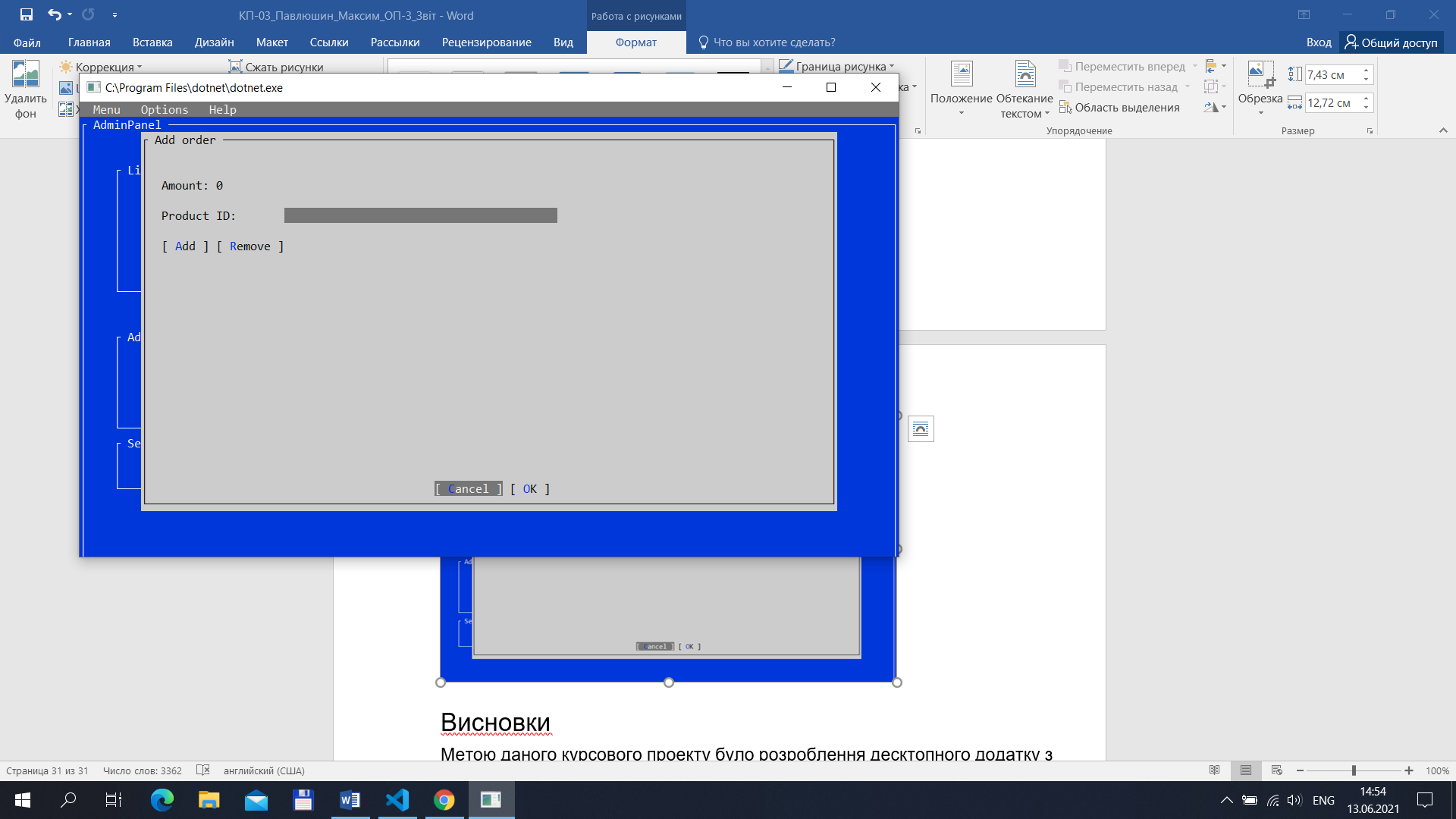
Редагування товару

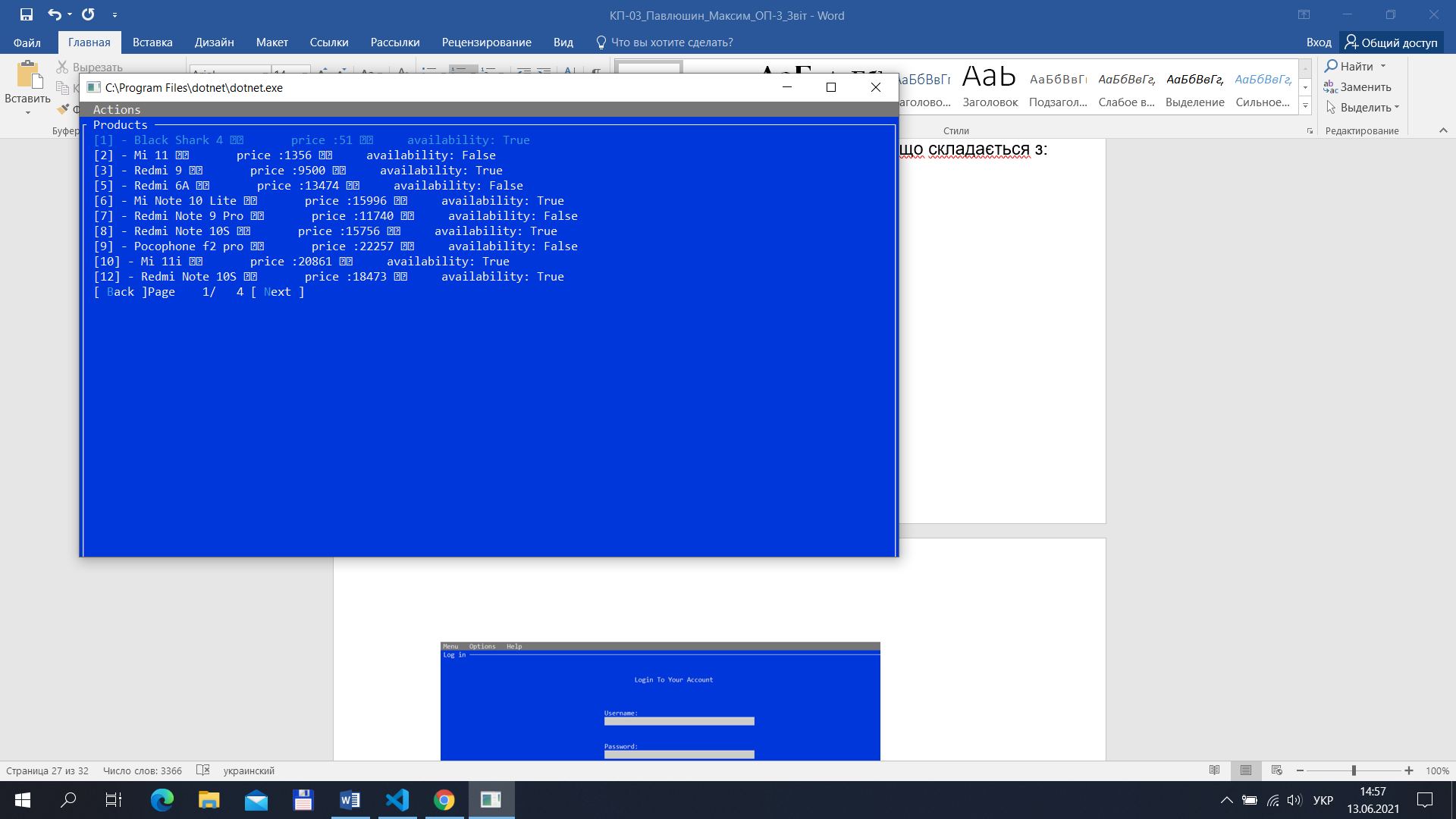
Пошук товару

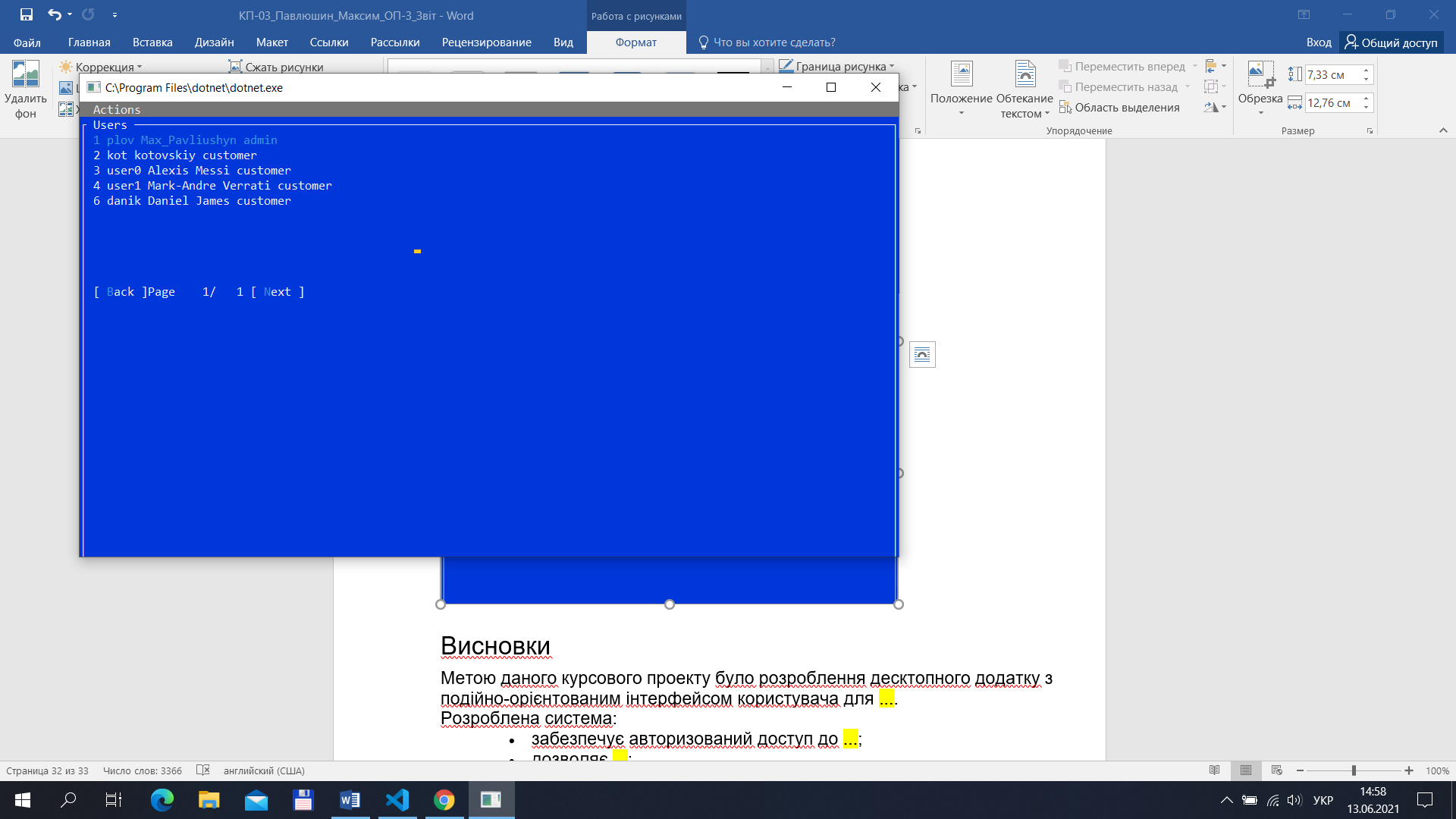
Імпорт товару з xml файлу

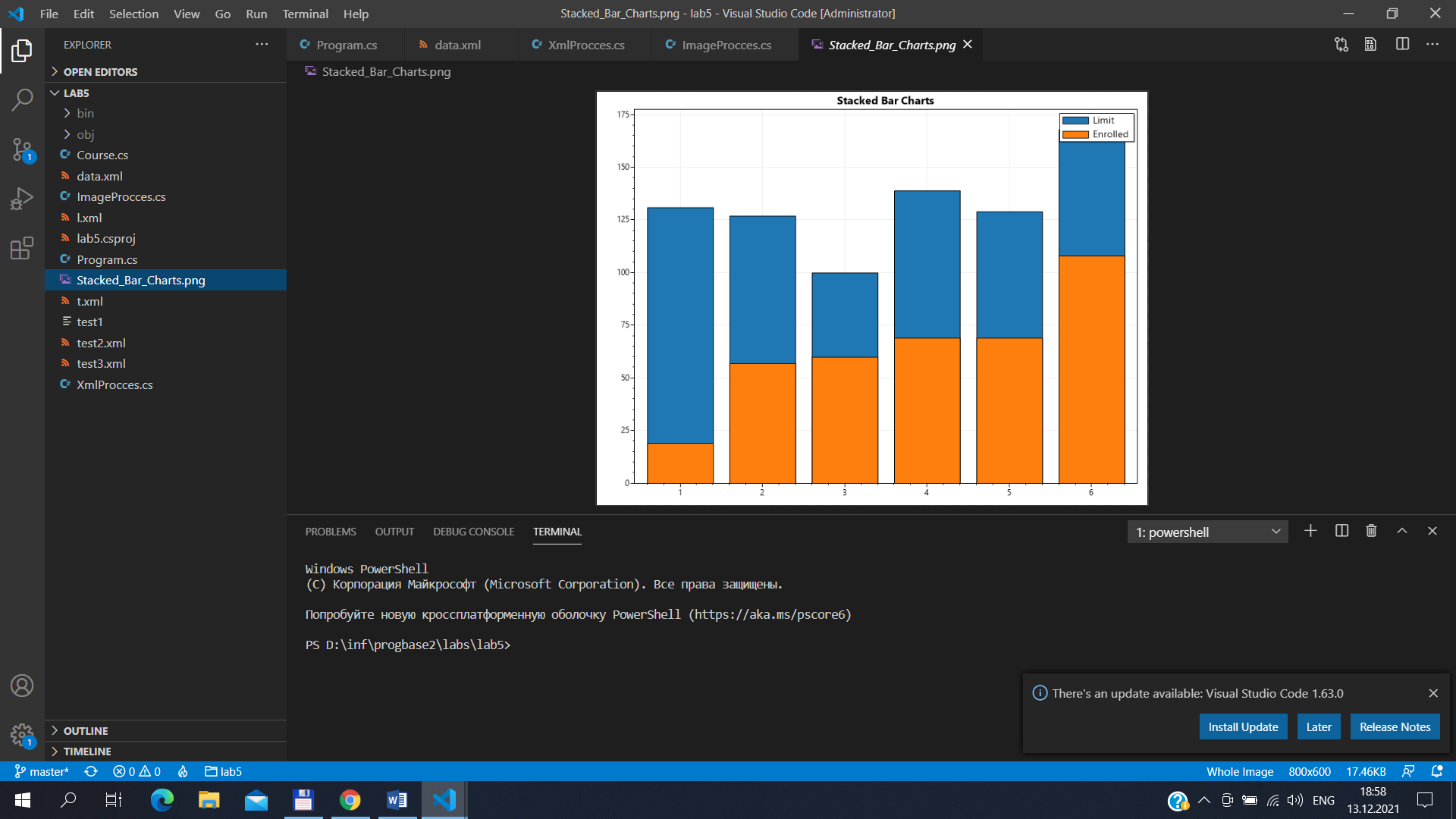
Експортувати замовлення в XML файл

  
Додати новий товар

Зробити замовлення

Список товарів

Список користувачів



Висновки

Метою даного курсового проекту було розроблення десктопного додатку з подійно-орієнтованим інтерфейсом користувача для того, щоб показати свої навички програмування.

Розроблена система:

* забезпечує авторизований доступ до аккаунтів користувачів, дані про яких зберігаються в базі даних;
* дозволяє шукати товари, замовлення, додавати товари, замовлення, користувачів, робити замовлення, видаляти товари, користувачів;
* дає можливість імпорту з xml файлу та експорту в xml файл;

Особливу увагу під час розроблення даного програмного продукту було приділено розбиттю коду на модулі та інтерфейсу користувача.

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

Використання розробленого додатку дозволить на простому прикладі подивитися як працюють інтернет магазини зсередини.