МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

кафедра програмних засобів

Звіт

з лабораторної роботи № 4

з дисципліни «Професійна практика програмної інженерії»

на тему:

«Робота з даними»

Виконав

Ст. гр. КНТ-137 В.В. Козлов

Прийняв

Старший викладач І.В. Левада

1. Мета роботи

Ознайомитися з мовою XML і можливостями створення з’єднань з базами даних через програмні додатки та навчитися на практиці використовувати XML-файли і бази даних в якості сховищ даних.

1. Завдання на лабораторну роботу

Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

Виконати аналіз ТЗ та розробленої архітектури системи щодо вимог до організації даних.

Прийняти рішення про систему збереження даних та обґрунтувати їх.

Розробити структуру XML-файлів і/або бази даних, необхідних для збереження даних системи.

Реалізувати функціональність програмного забезпечення, пов’язану зі взаємодією з даними.

Виконати тестування розробленого програмного забезпечення. У процесі тестування має обов’язково застосовуватись модульне тестування. Тестування має виконуватися шляхом взаємодії з різними файлами (в яких зберігаються дані) визначеної структури на пристроях з різними апаратними характеристиками під керуванням різних операційних систем або версій операційних систем.

Виконати аналіз отриманих результатів тестування. У процесі аналізу отриманих результатів має бути порівняно результати, отримані під керування різних операційних систем (або їх версій) та на різних пристроях.

Оформити звіт з роботи.

Відповісти на контрольні запитання.

1. Результати виконання роботи
   1. Аналіз предметної області

В результаті аналізу предметної області й згідно з технічим завданням в якості сховища даних та системи керування базою даних було обрано Microsoft SQL Server Express та Microsoft SQL Server Management Studio.

Microsoft SQL Server Express – це вільна комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів – Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями.

* 1. База даних

На рисунку 3.1 наведено схему бази даних.

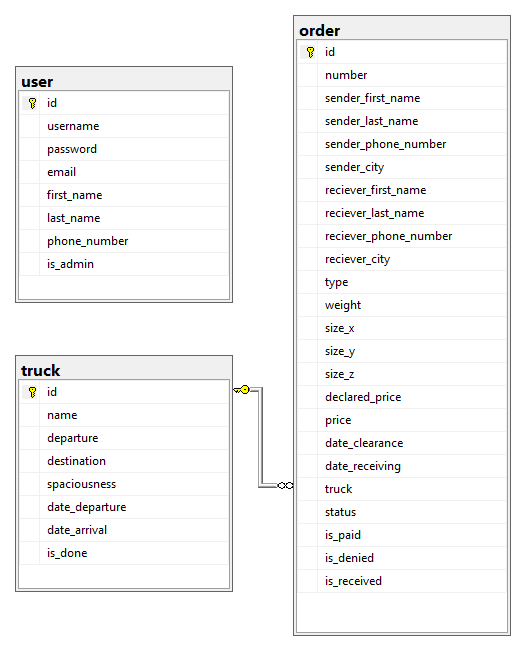


Рисунок 3.1 – Схема бази даних до програми «CargoDelivery»

* 1. Лістинг програми

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using CargoDelivery.Model;

namespace CargoDelivery

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

User NewUser = new User();

NewUser.Username = "valeriyzp";

NewUser.Password = "valeriyzp";

if (NewUser.isLoginAsUser()) MessageBox.Show("This user is Consumer");

if (NewUser.isLoginAsAdmin()) MessageBox.Show("This user is Operator");

Consumer NewConsumer = NewUser.getConsumer();

foreach (Order ConsumerOrder in NewConsumer.getOrders())

MessageBox.Show("Consumer order: " +

ConsumerOrder.SenderFirstName + " " + ConsumerOrder.SenderLastName +" -> " +

ConsumerOrder.RecieverFirstName + " " + ConsumerOrder.RecieverLastName);

Operator NewOperator = new Operator("Valeriy");

MessageBox.Show("Count of trucks for 1st order: " + NewOperator.getTrucksForOrder(1).Count.ToString());

foreach (Truck TruckForOrder in NewOperator.getTrucksForOrder(1))

MessageBox.Show("Truck name: " + TruckForOrder.Name);

}

}

}

* 1. Лістинг моделі

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

namespace CargoDelivery.Model

{

public class User : DBLoader

{

public int ID { get; set; } = -1;

public string Username { get; set; } = "";

public string Password { get; set; } = "";

public bool isLoginAsUser()

{

if (isLoginAsUser(Username, Password)) return true;

else return false;

}

public bool isLoginAsAdmin()

{

if (isLoginAsAdmin(Username, Password)) return true;

else return false;

}

public Consumer getConsumer()

{

return getConsumerByUsername(Username);

}

public Operator getOperator()

{

return new Operator(Username);

}

}

public class Consumer : User

{

public string Email { get; set; } = "";

public string FirstName { get; set; } = "";

public string LastName { get; set; } = "";

public string PhoneNumber { get; set; } = "";

public Consumer(int iD, string username, string email, string firstName, string secondName, string phoneNumber)

{

ID = iD;

Username = username;

Email = email;

FirstName = firstName;

LastName = secondName;

PhoneNumber = phoneNumber;

}

public Consumer()

{

;

}

public bool registerNewConsumer()

{

return registerNewConsumer(Username, Password, Email, FirstName, LastName, PhoneNumber);

}

public bool updateConsumerInfo()

{

return updateConsumerByID(ID, FirstName, LastName);

}

public List<Order> getOrders()

{

return getOrdersByPhoneNumber(PhoneNumber);

}

public bool addOrder(Order NewOrder)

{

return addNewOrder(NewOrder);

}

public bool cancelOrder(int id)

{

return cancelOrderByID(id);

}

}

public class Operator : User

{

public Operator(string username)

{

Username = username;

}

public bool payForOrder(int id)

{

return payForOrderByID(id);

}

public bool recieveOrder(int id)

{

return recieveOrderByID(id);

}

public List<Truck> getTrucksForOrder(int id)

{

return getTrucksForOrderByOrderID(id);

}

public bool addTruckToOrder(int truckID, int OrderID)

{

return addTruckToOrderByID(truckID, OrderID);

}

}

public class Truck

{

public int ID { get; } = -1;

public string Name { get; set; }

public string Departure { get; set; }

public string Destination { get; set; }

public int Spaciousness { get; set; }

public string DateDeparture { get; set; }

public string DateArrival { get; set; }

public bool IsDone { get; set; }

public Truck(int iD, string name, string departure, string destination, int spaciousness, string dateDeparture, string dateArrival, bool isDone)

{

ID = iD;

Name = name;

Departure = departure;

Destination = destination;

Spaciousness = spaciousness;

DateDeparture = dateDeparture;

DateArrival = dateArrival;

IsDone = isDone;

}

}

public class Order

{

public int ID { get; set; } = -1;

public string Number { get; set; } = "";

public string SenderFirstName { get; set; } = "";

public string SenderLastName { get; set; } = "";

public string SenderPhoneNumber { get; set; } = "";

public string SenderCity { get; set; } = "";

public string RecieverFirstName { get; set; } = "";

public string RecieverLastName { get; set; } = "";

public string RecieverPhoneNumber { get; set; } = "";

public string RecieverCity { get; set; } = "";

public string Type { get; set; } = "";

public int Weight { get; set; } = 0;

public int Size\_X { get; set; } = 0;

public int Size\_Y { get; set; } = 0;

public int Size\_Z { get; set; } = 0;

public int DeclaredPrice { get; set; } = 0;

public int Price { get; set; } = 0;

public string DateClearance { get; set; } = DateTime.Now.ToString("dd.MM.yyyy");

public string DateReceiving { get; set; } = "";

public int Truck { get; set; } = -1;

public string Status { get; set; } = "";

public bool IsPaid { get; set; } = false;

public bool IsDenied { get; set; } = false;

public bool IsReceived { get; set; } = false;

public Order(int iD, string number, string senderFirstName, string senderLastName, string senderPhoneNumber, string senderCity,

string recieverFirstName, string recieverLastName, string recieverPhoneNumber, string recieverCity,

string type, int weight, int size\_X, int size\_Y, int size\_Z, int declaredPrice, int price, string dateClearance,

string dateReceiving, int truck, string status, bool isPaid, bool isDenied, bool isReceived)

{

ID = iD;

Number = number;

SenderFirstName = senderFirstName;

SenderLastName = senderLastName;

SenderPhoneNumber = senderPhoneNumber;

SenderCity = senderCity;

RecieverFirstName = recieverFirstName;

RecieverLastName = recieverLastName;

RecieverPhoneNumber = recieverPhoneNumber;

RecieverCity = recieverCity;

Type = type;

Weight = weight;

Size\_X = size\_X;

Size\_Y = size\_Y;

Size\_Z = size\_Z;

DeclaredPrice = declaredPrice;

Price = price;

DateClearance = dateClearance;

DateReceiving = dateReceiving;

Truck = truck;

Status = status;

IsPaid = isPaid;

IsDenied = isDenied;

IsReceived = isReceived;

}

public Order()

{

;

}

public int get\_price()

{

int ResultPrice = 50 + Convert.ToInt32(DeclaredPrice \* 0.01);

if (Type == "Documents") ResultPrice += 20;

ResultPrice += (Convert.ToInt32(Size\_X \* Size\_Y \* Size\_Z \* 0.000001) \* 10);

ResultPrice += (Weight \* 10);

return ResultPrice;

}

}

public class DBLoader

{

private string connectionString = @"Data Source=DESKTOP-IK3PHF1\SQLEXPRESS;Initial Catalog=CargoDeliveryDB;Integrated Security=SSPI";

protected bool isLoginAsUser(string username, string password)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("SELECT count(id) FROM [user] WHERE username = @username AND password = @password AND is\_admin = 0;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter usernameParam = new SqlParameter("@username", username);

SqlParameter passwordParam = new SqlParameter("@password", password);

command.Parameters.Add(usernameParam);

command.Parameters.Add(passwordParam);

if (Convert.ToInt32(command.ExecuteScalar()) > 0) return true;

else return false;

}

catch (Exception ex)

{

return false;

}

}

}

protected bool isLoginAsAdmin(string username, string password)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("SELECT count(id) FROM [user] WHERE username = @username AND password = @password AND is\_admin = 1;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter usernameParam = new SqlParameter("@username", username);

SqlParameter passwordParam = new SqlParameter("@password", password);

command.Parameters.Add(usernameParam);

command.Parameters.Add(passwordParam);

if (Convert.ToInt32(command.ExecuteScalar()) > 0) return true;

else return false;

}

catch (Exception ex)

{

return false;

}

}

}

protected Consumer getConsumerByUsername(string username)

{

Consumer Temp = new Consumer();

Temp.Username = username;

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("SELECT id, email, first\_name, last\_name, phone\_number FROM [user] WHERE username = @username;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter usernameParam = new SqlParameter("@username", username);

command.Parameters.Add(usernameParam);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

Temp.ID = Convert.ToInt32(reader["id"]);

Temp.Email = Convert.ToString(reader["email"]).Trim();

Temp.FirstName = Convert.ToString(reader["first\_name"]).Trim();

Temp.LastName = Convert.ToString(reader["last\_name"]).Trim();

Temp.PhoneNumber = Convert.ToString(reader["phone\_number"]).Trim();

}

catch (Exception ex)

{

;

}

}

return Temp;

}

protected bool registerNewConsumer(string username, string password, string email, string first\_name, string last\_name, string phone\_number)

{

if (username.Count() == 0 || password.Count() == 0 || email.Count() == 0 || first\_name.Count() == 0 || last\_name.Count() == 0 || phone\_number.Count() == 0) return false;

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("SELECT COUNT(id) FROM [user] WHERE username = @username OR email = @email OR phone\_number = @phone\_number;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter usernameParam = new SqlParameter("@username", username);

SqlParameter emailParam = new SqlParameter("@email", email);

SqlParameter phone\_numberParam = new SqlParameter("@phone\_number", phone\_number);

command.Parameters.Add(usernameParam);

command.Parameters.Add(emailParam);

command.Parameters.Add(phone\_numberParam);

if (Convert.ToInt32(command.ExecuteScalar()) > 0) return false;

sqlExpression = string.Format("INSERT INTO [user] (username, password, email, first\_name, last\_name, phone\_number, is\_admin) VALUES (@username, @password, @email, @first\_name, @last\_name, @phone\_number, 0)");

command = new SqlCommand(sqlExpression, connection);

usernameParam = new SqlParameter("@username", username);

emailParam = new SqlParameter("@email", email);

phone\_numberParam = new SqlParameter("@phone\_number", phone\_number);

SqlParameter passwordParam = new SqlParameter("@password", password);

SqlParameter first\_nameParam = new SqlParameter("@first\_name", first\_name);

SqlParameter last\_nameParam = new SqlParameter("@last\_name", last\_name);

command.Parameters.Add(usernameParam);

command.Parameters.Add(emailParam);

command.Parameters.Add(phone\_numberParam);

command.Parameters.Add(passwordParam);

command.Parameters.Add(first\_nameParam);

command.Parameters.Add(last\_nameParam);

if (command.ExecuteNonQuery() > 0) return true;

else return false;

}

catch (Exception ex)

{

return false;

}

}

}

protected bool updateConsumerByID(int id, string first\_name, string last\_name)

{

if (id < 1) return false;

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("UPDATE [user] set first\_name = @first\_name, last\_name = @last\_name WHERE id = @id;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter iDParam = new SqlParameter("@id", id);

SqlParameter firstNameParam = new SqlParameter("@first\_name", first\_name);

SqlParameter lastNameParam = new SqlParameter("@last\_name", last\_name);

command.Parameters.Add(iDParam);

command.Parameters.Add(firstNameParam);

command.Parameters.Add(lastNameParam);

if (command.ExecuteNonQuery() > 0) return true;

else return false;

}

catch (Exception ex)

{

return false;

}

}

}

protected List<Order> getOrdersByPhoneNumber(string phone\_number)

{

List<Order> Orders = new List<Order>();

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("SELECT \* FROM [order] WHERE sender\_phone\_number = @phone\_number;"); // OR reciever\_phone\_number = @phone\_number;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter phoneNumberParam = new SqlParameter("@phone\_number", phone\_number);

command.Parameters.Add(phoneNumberParam);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Order NewOrder = new Order(Convert.ToInt32(reader["id"]), Convert.ToString(reader["number"]).Trim(),

Convert.ToString(reader["sender\_first\_name"]).Trim(), Convert.ToString(reader["sender\_last\_name"]).Trim(),

Convert.ToString(reader["sender\_phone\_number"]).Trim(), Convert.ToString(reader["sender\_city"]).Trim(),

Convert.ToString(reader["reciever\_first\_name"]).Trim(), Convert.ToString(reader["reciever\_last\_name"]).Trim(),

Convert.ToString(reader["reciever\_phone\_number"]).Trim(), Convert.ToString(reader["reciever\_city"]).Trim(),

Convert.ToString(reader["type"]), Convert.ToInt32(reader["weight"]),

Convert.ToInt32(reader["size\_x"]), Convert.ToInt32(reader["size\_y"]), Convert.ToInt32(reader["size\_z"]),

Convert.ToInt32(reader["declared\_price"]), Convert.ToInt32(reader["price"]),

Convert.ToString(reader["date\_clearance"]).Trim(), Convert.ToString(reader["date\_receiving"] == System.DBNull.Value ? "" : reader["date\_receiving"]).Trim(),

Convert.ToInt32(reader["truck"] == System.DBNull.Value ? -1 : reader["truck"]), Convert.ToString(reader["status"]).Trim(),

Convert.ToBoolean(reader["is\_paid"]), Convert.ToBoolean(reader["is\_denied"]), Convert.ToBoolean(reader["is\_received"]));

Orders.Add(NewOrder);

}

}

catch (Exception ex)

{

;

}

}

return Orders;

}

protected bool addNewOrder(Order NewOrder)

{

string date\_clearance = DateTime.Now.ToString("dd.MM.yyyy");

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("SELECT IDENT\_CURRENT('order');");

SqlCommand command = new SqlCommand(sqlExpression, connection);

string number = (Convert.ToInt32(command.ExecuteScalar()) + 1).ToString("2000000000000");

sqlExpression = string.Format("INSERT INTO [order] (number, sender\_first\_name, sender\_last\_name, sender\_phone\_number, sender\_city, " +

"reciever\_first\_name, reciever\_last\_name, reciever\_phone\_number, reciever\_city, type, weight, size\_x, size\_y, size\_z, " +

"declared\_price, price, date\_clearance, status, is\_paid, is\_denied, is\_received) " +

"VALUES (@number, @sender\_first\_name, @sender\_last\_name, @sender\_phone\_number, @sender\_city, " +

"@reciever\_first\_name, @reciever\_last\_name, @reciever\_phone\_number, @reciever\_city, @type, @weight, @size\_x, @size\_y, @size\_z, " +

"@declared\_price, @price, @date\_clearance, @status, 0, 0, 0)");

command.CommandText = sqlExpression;

command.Parameters.AddWithValue("@number", number);

command.Parameters.AddWithValue("@sender\_first\_name", NewOrder.SenderFirstName);

command.Parameters.AddWithValue("@sender\_last\_name", NewOrder.SenderLastName);

command.Parameters.AddWithValue("@sender\_phone\_number", NewOrder.SenderPhoneNumber);

command.Parameters.AddWithValue("@sender\_city", NewOrder.SenderCity);

command.Parameters.AddWithValue("@reciever\_first\_name", NewOrder.RecieverFirstName);

command.Parameters.AddWithValue("@reciever\_last\_name", NewOrder.RecieverLastName);

command.Parameters.AddWithValue("@reciever\_phone\_number", NewOrder.RecieverPhoneNumber);

command.Parameters.AddWithValue("@reciever\_city", NewOrder.RecieverCity);

command.Parameters.AddWithValue("@type", NewOrder.Type);

command.Parameters.AddWithValue("@weight", NewOrder.Weight);

command.Parameters.AddWithValue("@size\_x", NewOrder.Size\_X);

command.Parameters.AddWithValue("@size\_y", NewOrder.Size\_Y);

command.Parameters.AddWithValue("@size\_z", NewOrder.Size\_Z);

command.Parameters.AddWithValue("@declared\_price", NewOrder.DeclaredPrice);

command.Parameters.AddWithValue("@price", NewOrder.get\_price());

command.Parameters.AddWithValue("@date\_clearance", date\_clearance);

command.Parameters.AddWithValue("@status", "Accepted");

if (command.ExecuteNonQuery() > 0) return true;

else return false;

}

catch (Exception ex)

{

return false;

}

}

}

protected bool cancelOrderByID(int id)

{

if (id < 1) return false;

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("Update [order] SET is\_denied = 1 WHERE id = @id AND is\_received = 0;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter iDParam = new SqlParameter("@id", id);

command.Parameters.Add(iDParam);

if (command.ExecuteNonQuery() > 0)

{

return updateOrderStatusByID(id, "Canceled");

}

else return false;

}

catch (Exception ex)

{

return false;

}

}

}

protected bool payForOrderByID(int id)

{

if (id < 1) return false;

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("Update [order] SET is\_paid = 1 WHERE id = @id AND is\_received = 0 AND is\_denied = 0;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter iDParam = new SqlParameter("@id", id);

command.Parameters.Add(iDParam);

if (command.ExecuteNonQuery() > 0) return true;

else return false;

}

catch (Exception ex)

{

return false;

}

}

}

protected bool recieveOrderByID(int id)

{

if (id < 1) return false;

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("Update [order] SET is\_received = 1 WHERE id = @id AND is\_denied = 0 AND is\_paid = 1;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter iDParam = new SqlParameter("@id", id);

command.Parameters.Add(iDParam);

if (command.ExecuteNonQuery() > 0)

{

return updateOrderStatusByID(id, "Completed");

}

else return false;

}

catch (Exception ex)

{

return false;

}

}

}

protected bool updateOrderStatusByID(int id, string status)

{

if (id < 1) return false;

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("Update [order] SET status = @status WHERE id = @id;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.Parameters.AddWithValue("@status", status);

command.Parameters.AddWithValue("@id", id);

if (command.ExecuteNonQuery() > 0) return true;

else return false;

}

catch (Exception ex)

{

return false;

}

}

}

protected List<Truck> getTrucksForOrderByOrderID(int id)

{

List<Truck> Trucks = new List<Truck>();

string departure = "", destination = "", date\_clearance = "";

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("SELECT sender\_city, reciever\_city, date\_clearance FROM [order] WHERE id = @id;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlParameter iDParam = new SqlParameter("@id", id);

command.Parameters.Add(iDParam);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

departure = Convert.ToString(reader["sender\_city"]).Trim();

destination = Convert.ToString(reader["reciever\_city"]).Trim();

date\_clearance = Convert.ToString(reader["date\_clearance"]).Trim();

}

reader.Close();

sqlExpression = string.Format("SELECT \* FROM [truck] WHERE departure = @departure AND destination = @destination AND date\_departure >= @date\_departure;");

command = new SqlCommand(sqlExpression, connection);

command.Parameters.AddWithValue("@departure", departure);

command.Parameters.AddWithValue("@destination", destination);

command.Parameters.AddWithValue("@date\_departure", date\_clearance);

reader = command.ExecuteReader();

while (reader.Read())

{

Truck NewTruck = new Truck(Convert.ToInt32(reader["id"]), Convert.ToString(reader["name"]).Trim(),

departure, destination, Convert.ToInt32(reader["spaciousness"]), Convert.ToString(reader["date\_departure"]).Trim(),

Convert.ToString(reader["date\_arrival"]).Trim(), Convert.ToBoolean(reader["is\_done"]));

Trucks.Add(NewTruck);

}

}

catch (Exception ex)

{

;

}

}

return Trucks;

}

protected bool addTruckToOrderByID(int truckID, int OrderID)

{

if (truckID < 1) return false;

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

string sqlExpression = string.Format("Update [order] SET truck = @truck WHERE id = @id AND truck is null;");

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.Parameters.AddWithValue("@truck", truckID);

command.Parameters.AddWithValue("@id", OrderID);

if (command.ExecuteNonQuery() > 0)

{

return updateOrderStatusByID(OrderID, "On the way");

}

else return false;

}

catch (Exception ex)

{

return false;

}

}

return false;

}

}

}

* 1. Модульне тестування

Метою модульного тестування є ізоляція кожної частини програми та впевненість у тому, що кожна окрема частина є коректною. Зазвичай unit-тести застосовують для того, щоб упевнитися, що код відповідає вимогам архітектури та має очікувану поведінку.

Для тестування було обрано такі реалізовані функції, як розрахунок вартості замовлення, перевірка правильності авторизації користувача, автоматична генерація деяких даних.

* 1. Лістинг модульного тесту

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using CargoDelivery.Model;

namespace CargoDelivery.Tests

{

[TestClass]

public class CargoDeliveryTests

{

[TestMethod]

public void IsRightPrice()

{

Order TestOrder = new Order();

TestOrder.SenderFirstName = "Valeriy";

TestOrder.SenderLastName = "Kozlov";

TestOrder.SenderPhoneNumber = "0950000000";

TestOrder.SenderCity = "Zaporizhzhya";

TestOrder.RecieverFirstName = "Ilon";

TestOrder.RecieverLastName = "Mask";

TestOrder.RecieverPhoneNumber = "0950000001";

TestOrder.RecieverCity = "Lviv";

TestOrder.Type = "Documents";

TestOrder.Weight = 1;

TestOrder.Size\_X = 10;

TestOrder.Size\_Y = 10;

TestOrder.Size\_Z = 10;

TestOrder.DeclaredPrice = 100;

Assert.AreEqual(TestOrder.get\_price(), 81);

}

[TestMethod]

public void IsAdmin()

{

User TestUser = new User();

TestUser.Username = "admin";

TestUser.Password = "admin";

Assert.AreEqual(TestUser.isLoginAsAdmin(), true);

}

[TestMethod]

public void IsUser()

{

User TestUser = new User();

TestUser.Username = "user";

TestUser.Password = "user";

Assert.AreEqual(TestUser.isLoginAsUser(), true);

}

[TestMethod]

public void IsRightNewOrderDateClearance()

{

Order NewOrder = new Order();

Assert.AreEqual(NewOrder.DateClearance, DateTime.Now.ToString("dd.MM.yyyy"));

}

[TestMethod]

public void IsRightUsernameInNewConsumer()

{

User NewUser = new User();

NewUser.Username = "TestUsername";

Assert.AreEqual(NewUser.getConsumer().Username, "TestUsername");

}

}

}

* 1. Результати тесту

Результати модульного тесту наведено на рисунку 3.2.

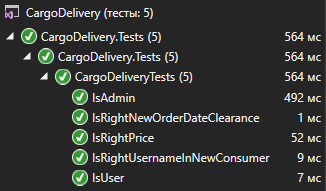


Рисунок 3.2 – Результати модульного тесту

Висновки

Протягом виконання цієї лабораторної роботи ознайомився з мовою XML і можливостями створення з’єднань з базами даних через програмні додатки. Навчився аналізувати технічне завдання й предметну область. Навчився на практиці використовувати базу даних в якості сховища даних.

На основі проведеного аналізу й дослідження розробив схему бази даних та на її основі створив базу даних.

Реалізував клас та методи для роботи з базою даних за допомогою мови програмування C#.

Провів модульні тести для перевірки працездатності основних функції програми. Всі тести було пройдено, а отже програма працездатна.

Отримані результати задовольняють поставленим задачам та виконуються з достатньою швидкістю.