АНОТАЦІЯ

Магістерська робота на тему «Розробка програмного забезпеченнядля реалізації продажу електроніки з використанням технології .NET».

Методи розробки базується на технології .Net Framework, сервер бази даних MsSQL.

Заоборний Сергій Олегович Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженерії, група СПм–61, Тернопіль, 2019.

Дана робота містить 75 сторінок, 6 таблиць, 23 рисунків, список використаної літератури з 18 найменувань та 3 додатки.

Використана методика проектування інформаційних систем з модульною структурою та застосуванням сучасних технологій програмування.

Здійснена програмна реалізація дозволяє забезпечення ефективного обміну інформацією і реалізації базових функції обробки інформації для вибраної сфери.

Ключові слова: БАЗИ ДАНИХ, ІНТЕРНЕТ, ІНТЕРФЕЙС, ДОВІДКА, ПРОГРАММА, СЕРВЕР, API, .NET FRAMEWORK, MSSQL.

ABSTRACT

Master's thesis on "Software Development for Sales of Electronics Using .NET Technology".

The development methods are based on .Net Framework technology, an MsSQL database server.

Zaoborny Sergiy Olegovich Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, SPm-61 Group, Ternopil, 2019.

This work contains 75 pages, 6 tables, 23 figures, a list of used literature of 18 titles and 3 appendices.

The methodology of information systems design with modular structure and application of modern programming technologies is used.

The implemented software implementation allows for efficient exchange of information and implementation of basic information processing functions for the selected area.

Keywords: DATABASE, INTERNET, INTERFACE, HELP, PROGRAM, SERVER, API, .NET FRAMEWORK, MSSQL.

ЗМІСТ

Вступ 7
1 Аналіз предметної області та використані технології 8
1.1 Аналіз предметної області 8
1.2 Постановка задачі 9
1.3 Модель розробки системи 10
1.4 Актори та варіанти використання 11
1.5 Використані технологій для розробки 13
1.5.1 Мова програмування 13
1.5.2 База даних 14
1.5.3 Підходи до розробки бази даних 18
1.5.4 ORM 20
1.5.5 OData 26
2 Проектування бази даних та розробка бізнес-логіки 29
2.1 Розробка бази даних 29
2.1.1 Побудова UML-діаграми класів для взаємодії з базою даних 29
2.1.2 Класи використані для бази даних 32
2.2 Розробка бізнес-логіки 40
3 Розробка користувацького інтерфейсу 51
3.1 Технології застосовані для розробки програми 51
3.2 Інтерфейс програми 54
4 Організаційно-економічна частина 58
4.1 Розрахунок норм часу на виконання науково-дослідної роботи 58
5 Охорона праці та безпека в надзвичайних ситуац 64
5.1 ОХОРОНА ПРАЦІ 64
5.2 ПІДВИЩЕННЯ СТІЙКОСТІ РОБОТИ ОБ'ЄКТІВ ГОСПОДАРСЬКОЇ ДІЯЛЬНОСТІ У ВОЄННИЙ ЧАС 67
ВИСНОВКИ 72

Перелік використаних джерел 73
ДОДАТКИ……...75

ВСТУП

Ведення обліку товару в паперовому вигляді - це відлуння минулого. Сучасні фахівці використовують у своїй роботі інформаційний комплекс спеціалізованого ПЗ. Його впровадження в процес обліку та торговельної діяльності дозволяє створювати статистику в електронному вигляді і ефективно виконувати інші процеси торгівлі. Це економить час і кошти. Всі дані передаються до віртуальної бази даних для подальшого ведення обліку та статистики проданого товару. Система володіє широким спектром функцій. Підвищується не тільки швидкість виконання роботи фахівцями організації, але і якість результатів.

Ведення віртуального обліку товару забезпечує оптимізацію бізнес-процесів і підвищення ефективності роботи фахівців фінансового, бухгалтерського та торгівельного відділів. Завдяки використанню системи покращується якість роботи організації в цілому. Застосування сучасних інструментів сприяє успішному розвитку бізнесу.

Характерними рисами системи є те, що вона можуть пропонувати значно більшу кількість додаткової інформації, та в будь-який час, ніж реальний журнал і забезпечувати користувачів значно більшим обсягом інформації, необхідної для моніторингу статистики продажів.

Для створення відповідного програмного продукту використані технології .Net фреймворку, та базу даних MsSQL.

Інформаційні системи потребують значно менших витрат на утримання та організацію роботи, оскільки у них значно обмеженіша матеріально-технічна база (будівлі, споруди, приміщення) та кількість обслуговуючого персоналу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИКОРИСТАНІ ТЕХНОЛОГІЇ

1.1 Аналіз предметної області

Продаж товарів – це договір, за яким продавець передає або погоджується передати право власності на товар покупцеві в обмін на гроші або послуги. Якщо право власності має перейти у майбутній час, договір називається договором про продаж..

Система управління складом - це програмно-апаратна система управління складом, яка забезпечує комплексну автоматизацію управління складськими та логістичними процесами.

Авторизація - це механізм захисту для визначення рівнів доступу або привілеїв користувача / клієнта, пов’язаних із системними ресурсами, включаючи файли, послуги, комп'ютерні програми, дані та функції додатків[3].

Ідентифікація – це процес розпізнавання користувача певною системою за допомогою наперед встановленого ідентифікатора, наприклад ім'я або ID користувача[3].

Автентифікація – процедура перевірки приналежності користувачеві інформації в системі [3].

C# - об'єктно-орієнтована мова програмування від компанії Microsoft, що має на меті поєднати обчислювальну потужність C++ з простотою програмування Visual Basic та кросплатформовість.

.NET — це набір інструментів, що складаються з компонентів часу виконання, бібліотеки та компілятора, а також надає можливість створювати додатки, які працюють на Windows, Mac OS X та Linux.

Кросплатформність — властивість програмного забезпечення працювати більш ніж на одній програмній або апаратній платформі[4].

MySQL - це найпопулярніша база даних із відкритим кодом у світі. Завдяки своїй перевіреній продуктивності, надійності та простоті використання MySQL стало провідним вибором баз даних для веб-додатків, включаючи Facebook, Twitter, YouTube, Yahoo! і багато іншого[4].

‘

СУБД –це спеціальна система керування базою даних[4].

1.2 Постановка задачі

Після проведення детального аналізу предметної області в попередньому пункті і виявлення основних проблем, перед нами постали наступні завдання, які потрібно вирішити. Перше завдання – який тип бази даних нам необхідно використати. Які процеси можна автоматизувати в даній предметній області. Яким чином буде проходити реєстрації користувачів у систему. Чи необхідні додаткові дані для внесення в базу даних, чи достатньо тих, що були згадані в попередньому пункті. Які функції будуть приорітетними для кінцевого користувача і чи вони необхідні йому. Чи взагалі необхідна інформаційна система для даної предметної області. Які процеси потріно покращити і чи потребують вони цього взагалі. Який буде графічний інтерфейс користувача і наскільки він буде зручним. Чи достатньо сутностей, які знайдені в аналізі предметної області, можливо є зайві або їх навпаки не достатньо. Чи в достатньому обсязі вивчена предметна область, чи необхідно додатковий її аналіз. Цей перелік може бути значно довшим, але зупинимося на цих, томущо вони основні. Сформулюємо наступний перелік завдання, провівши аналіз попередніх даних:

1) який тип БД використиати;
2) яким чином зберігати дані;
3) як з’єднуватися з БД;
4) процеси для покращення;
5) функції, які необхідні для користувача;
6) Обмеження для різних типів користувачів;
7) сутності бази даних і зв’язки між ними;

Основні запити, які задає користувач будуть здійснюватися через графічний інтерфейс, який буде приховувати їх у в собі. Проте основні запити будуть на вибірку і сортування даних у БД, щоб спростити сприйняття інтерфейсу користувачем. Будуть запити на додавання і

редагування даних, які буде виконувати система після заповнення відповідних полів.

1.3 Модель розробки системи

У ході розробки була вибрана спіральна модель розробки програмного забезпечення.

Дана модель життєвого циклу характерна при розробці новаторських (нетипових) систем. На початку роботи над проектом у замовника і розробника немає чіткого бачення підсумкового продукту (вимоги не можуть бути чітко визначені) або стовідсоткової впевненості в успішній реалізації проекту (ризики дуже великі). У зв'язку з цим приймається рішення розробки системи по частинах з можливістю зміни вимог або відмови від її подальшого розвитку. Як показано на рис. 1.7, розвиток проекту може бути завершено не тільки після стадії впровадження, але і після стадії аналізу ризику..

ЖЦ спіральної розробки ПЗ (див. рис 1.1).

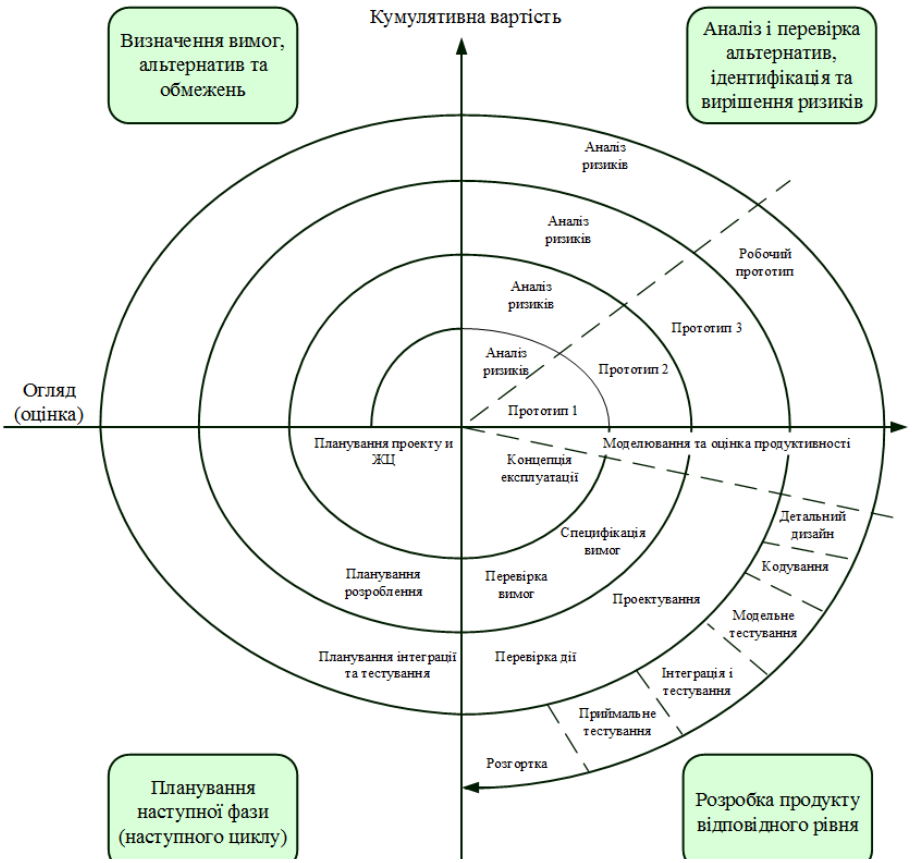


Рисунок 1.1 Принцип роботи спіральної моделі

‘

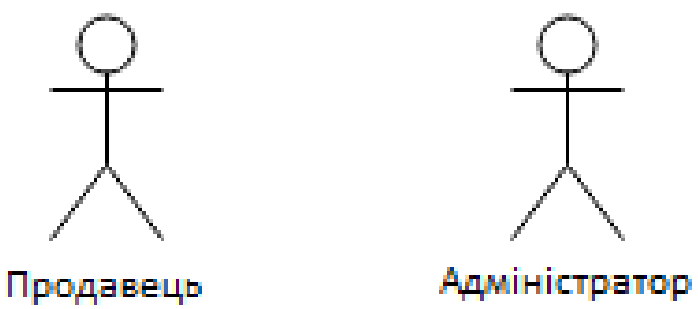
Спіральна модель - одна з найважливіших моделей життєвого циклу розробки програмного забезпечення, яка забезпечує підтримку управління ризиками. У своєму схематичному зображенні вона виглядає як спіраль з безліччю петель. Точна кількість спіральних петель невідома і може змінюватись від проекту до проекту. Кожна петля спіралі називається Фазою процесу розробки програмного забезпечення. Точна кількість фаз, необхідних для розробки продукту, може змінювати менеджер проекту залежно від ризиків проекту. Оскільки менеджер проекту динамічно визначає кількість фаз, менеджер проекту відіграє важливу роль у розробці продукту за допомогою спіральної моделі.

Радіус спіралі в будь-якій точці представляє собівартість (вартість) проекту до цього часу, а кутовий розмір являє собою прогрес, досягнутий в поточній фазі.

Ризик - це будь-яка несприятлива ситуація, яка може вплинути на успішне завершення програмного проекту. Найважливішою особливістю спіральної моделі є поводження з цими невідомими ризиками після початку проекту. Такі рішення щодо ризику простіше зробити шляхом розробки прототипу. Спіральна модель підтримує подолання ризиків, надаючи можливості для побудови прототипу на кожному етапі розробки програмного забезпечення.

1.4 Актори та варіанти використання

Згідно з посталеним завданням, система повинна поділятись на дві частини дистрибутивну і адміністративну(рис 1.2).



Короткий опис акторів представлений в таблиці. 1.1.

Табл. 1.1. Виявлення акторів

Актор	*Короткий опис*
Продавець	Особа яка має обмежений доступ до даних. Може продавати товар та переглядати списки наявних товарів, та тих, що закінчилися.
Адміністратор	Особа яка має доступ до всіх даних системи: списки проданих, наявних товарів, тих, що закінчилися. Також надається доступ до системи редагування , видалення та додавання товару.

Виявлення варіантів використання зведене в таблицю 1.2.

Табл. 1.2. Виявлення варіантів використання

Продавець	Продаж товару	Продавцеві надається можливість здійснювати продаж наявних товарів.
Продавець, адміністратор	Перегляд списку товарів	Продавцеві та адміністраторові надається можливість здійснювати перегляд наявних товарів, а також тих, що закінчилися.
Адміністратор	Додавання нового товару в БД	Адміністратору надається можливість додати новий товар.
Адміністратор	Редагування товару в БД	Адміністратору надається можливість редагувати наявний товар.
Адміністратор	Видалення товару з БД	Адміністратору надається можливість видалити наявний товар.
Адміністратор	Перегляд проданого товару	Адміністратору надається можливість переглянути список проданого товару за певний період часу: день, тиждень,

		місяць.

Всі варіанти використання приведені на (рис. 1.3).

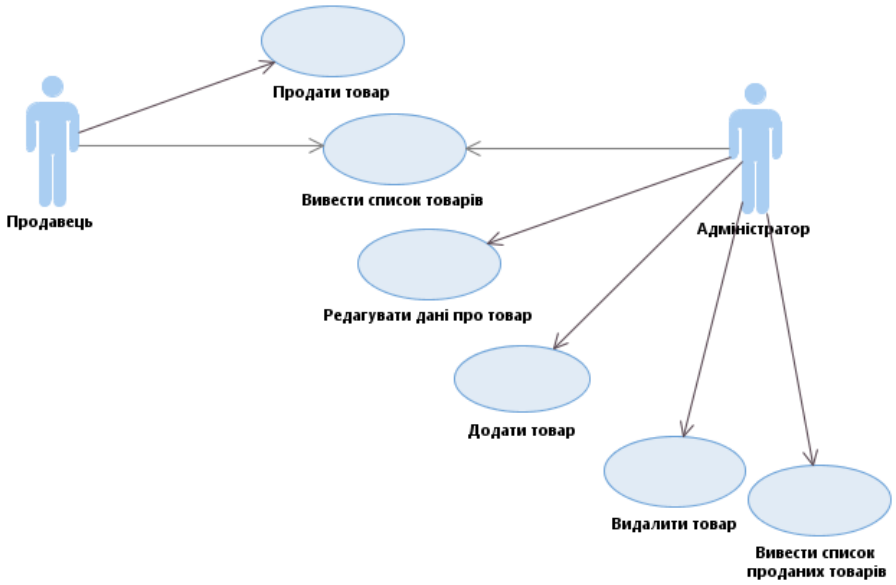


Рисунок 1.3 - Діаграма прецедентів системи.

1.5 Використані технологій для розробки

1.5.1 Мова програмування

Для реалізації проекту було обрана мова програмування C#, яка найкраще підходить для розробки програм для PC, і має такі особливості:

1) об’єктно-орієнтована мова;
2) легко розробляти ПЗ будь – якого рівня складності завдяки великій кількість функцій та класів, які реалізовані;
3) cross-платформа що дозволяє запускати програму на різних платформах;
4) постійна підтримка та покращення мови.

Для отримання даних з таблиці БД в C# є два варіанти: використовувати SQL запити або Linq до DbSet, який дозволяє перетворювати запити в SQL. Запити Linq є простішими для читання і сприйняття, ніж запити SQL, тому був обраний цей спосіб звернення до БД.

‘

Linq розширює мову шляхом додавання виразів-запитів, які схожі з SQL, і може бути використаний, щоб легкого отримувати і обробити дані з масивів, перелічуваних класів, реляційних баз даних і джерел.

Linq також визначає набір імен методів (названих стандартні оператори запитів, або стандартні оператори умов), а також з правилами трансформування, використовуваних компілятором, щоб перевести вирази-запити довільного стилю у виразах, що використовують ці імена методів, лямбда-виразів і анонімних типів [4].

Багато з концепцій LINQ були спочатку випробувані в США дослідницького проекту Microsoft. LINQ була випущена в якості основної частини .NET Framework 3.5 19.

Тепер наведемо приклад роботи Linq, який використаний в проекті.

Лістинг 1.1 – Linq запиту

```
DbList
     .Where(n => n.Marks
                 .Any(m => CourseId.Contains(m.CourseId))
 && (n.FullName.Contains(Name) || n.StudentId == AccountId))
                      .Select(m => new StudentColumn
                      {
                          Marks = m.Marks.ToList(),
            Headers = m.Marks.Select(n => n.Column).ToList(),
          CourseTitle = m.Marks.FirstOrDefault().Course.Title
                      })
                      .ToList();
```

Даний код показує використання Linq запитів, перший крок це умова Where, яка фільтрує результат, Any дозволяє вказати будь-яка умова з масиву значень, Select дозволяє спроектувати послідовність в інший об’єкт, що у свою чергу дозволяє спростити запит.

1.5.2 База даних

База даних — впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно та призначені для задоволення інформаційних

потреб користувачів. У технічному розумінні включно й система керування БД [1].

Сьогодні БД стали невід’ємною частиною нашого життя. Вони введені й успішно використовуються практично в усіх сферах людської діяльності. У сучасних умовах однією з найбільш розповсюджених задач, яка виникає при користуванні комп'ютером є використання системи які є засновані на БД.

До використання БД призвели ряд функцій, які вони забезпечують для даних, а саме:

1) велика кількість даних;
2) незалежність даних;
3) вільний доступ до даних;
4) підтримка транзакцій;
5) гарантована відсутність втрати даних;
6) одночасна робота з великою кількістю користувачів.

В наш час розрізняють 2 типи БД структуровані та неструктуровані. Структуровані дані - це дані, які були організовані у форматоване сховище, як правило, в базу даних, щоб його елементи могли бути адресованими для більш ефективної обробки та аналізу.

Структура даних - це своєрідне сховище, яке організовує інформацію. Наприклад, у базі даних кожне поле дискретне, і його інформація може бути отримана окремо або разом із даними з інших полів у різних комбінаціях. Потужність бази даних полягає в її здатності робити дані всеохоплюючими, щоб вони надавали корисну інформацію. Мова запитів бази даних, така як SQL (стандартна мова запитів), дозволяє адміністратору бази даних взаємодіяти з базою даних..

В якості БД використовується реляційний тип оскільки для сайту не потрібний швидкий пошук.

Реляційна база даних - це набір формально описаних таблиць, з яких можна отримати доступ або повторно зібрати дані різними способами без необхідності реорганізації таблиць БД. Стандартний інтерфейс

програмування користувачів та прикладних програм (API) реляційної бази даних - це структурована мова запитів (SQL). На даному етапі в основному використовуються Oracle, MySQL, Microsoft SQL Server які є найбільш популярним згідно http://db-engines.com/en/ranking. Тому зупинимо вибір на одній з них, але перед цим необхідно дізнатися про них більше, щоб зробити вибір згідно вимог.

База даних Oracle - це сукупність даних, що трактуються як одиниця. Призначення бази даних - зберігання та отримання пов’язаної інформації. Сервер бази даних - це ключ до вирішення проблем управління інформацією. Взагалі сервер надійно управляє великою кількістю даних у багатокористувацькому середовищі, тому багато користувачів можуть одночасно отримувати доступ до самих даних. Все це досягається при забезпеченні високої продуктивності. Сервер бази даних також запобігає несанкціонованому доступу та забезпечує ефективні рішення для відновлення несправностей.

Oracle Database - це перша база даних, розроблена для корпоративних сіткових обчислень, найбільш гнучкий і економічно ефективний спосіб управління інформацією та програмами. Сіткові обчислення підприємства створюють великі пули стандартних для промисловості модульних сховищ та серверів. Завдяки цій архітектурі кожну нову систему можна швидко забезпечити з пулу компонентів. Немає необхідності в пікових навантаженнях, оскільки потужність може бути легко додана або перерозподілена з ресурсних пулів за потребою. Останні версії СУБД Oracle значно простіше в установці і початковому налаштуванні. Також зросли можливості по спеціалізованій налаштування роботи СУБД під конкретну задачу. В результаті, і при роботі з OLTP-системою, і з сховищем даних, використовуючи ці можливості з налаштування СУБД Oracle, можна досягти справді вражаючих результатів.

Microsoft SQL Server - це система управління реляційними базами даних (RDBMS), яка підтримує широкий спектр програм для обробки транзакцій, бізнес-аналітики та аналітики в корпоративних IT-середовищах.

Microsoft SQL Server є однією з трьох лідируючих на ринку технологій баз даних, поряд з Oracle Database та IBM DB2.

Як і інше програмне забезпечення RDBMS, Microsoft SQL Server побудований на основі SQL, стандартизованої мови програмування, яку адміністратори бази даних (DBA) та інші IT-фахівці використовують для управління базами даних та запиту даних, які вони містять. SQL Server пов'язаний з Transact-SQL (T-SQL), реалізацією SQL від Microsoft, яка додає набір стандартних розширень програмного забезпечення до стандартної мови[3].

Як і інші технології RDBMS, SQL Server в основному побудований навколо структури на основі рядків, яка з'єднує пов'язані елементи даних у різних таблицях один з одним, уникаючи необхідності надмірно зберігати дані в декількох місцях у базі даних. Реляційна модель також забезпечує референтну цілісність та інші обмеження цілісності для підтримки точності даних. Ці перевірки є частиною більш широкого дотримання принципів атомності, узгодженості, ізоляції та довговічності, спільно відомих як властивості ACID, і розроблені таким чином, щоб гарантувати надійну обробку операцій з базами даних.

Основним компонентом Microsoft SQL Server є SQL Server Database Engine, який контролює зберігання, обробку даних та безпеку. Він включає реляційний двигун, який обробляє команди та запити, і механізм зберігання даних, який управляє файлами баз даних, таблицями, сторінками, індексами, буферами даних та транзакціями. Збережені процедури, тригери, перегляди та інші об'єкти бази даних також створюються та виконуються Механіком бази даних.

Під двигуном бази даних знаходиться операційна система SQL Server або SQLOS. SQLOS обробляє функції нижчого рівня, такі як управління пам'яттю та введення-виведення, планування роботи та блокування даних, щоб уникнути конфліктних оновлень. Мережевий рівень інтерфейсу розташовується над Database Engine і використовує табличний протокол потоку даних Microsoft для полегшення взаємодії запитів і відповідей із

серверами баз даних. І на рівні користувача, DBA і розробники SQL Server записують T-SQL заяви для створення та зміни структур баз даних, маніпулювання даними, здійснення захисту безпеки та резервного копіювання баз даних, серед інших завдань[3].

Основні причина для вибору БД Microsoft SQL Server, як базу даних для розробки:

1) регулярна підтримка користувачів;
2) легка в користуванні і підтримці;
3) автоматизація складних завдань;
4) дозволяє створювати моделі БД, яка є ефективною методологією для проектування;
5) дозволяє переносити моделі з існуючої БД;
6) полегшує написання документації;
7) мова для програмування працює безпосередньо з Microsoft SQL Server;
8) наявність localDD, що спрощує розробку.

1.5.3 Підходи до розробки бази даних

Два найбільш популярних і гнучких підходи: Code First і Database First, їх особливості, і проблеми, які виникають при розробці додатків, що використовують дані підходи. Для вибору між ними спочатку необхідно дізнатися про кожний з них більшу характеристику та результати практичного застосування двох даних підходів, доцільність та ефективність використання даних підходів в розробці реальних веб-додатків [4].

Code First – підхід до побудови сховища даних програми, підтримка якого здійснюється за допомогою технології ORM. Основною ідеєю даного підходу є написання коду (моделі даних або сутностей, які являють собою об’єкти - класи, логічно взаємопов’язані між собою) і генерація бази даних на основі цієї моделі за допомогою технології ORM [4].

В даному випадку програміст описує клас, який буде представляти таблицю бази даних і поля, які представлятимуть собою поля в таблиці бази

даних. Модель даних дуже легко змінити, внаслідок чого зміниться і сама база даних. Ці зміни ORM відстежує за допомогою своєї службової таблиці – «MetaData» (в різних ORM різні таблиці), яка створюється під час генерації бази даних з моделі Вона зберігає хеш-код моделі даних програми. Порівнюючи цей хеш-код, ORM знає, чи відбулися зміни в моделі чи ні, і якщо так, то її треба оновити.

Використання даного підходу вельми привабливо для програміста у зв'язку з тим, що для того щоб оновити структуру бази даних, необхідно оновити модель шляхом додавання або видалення полів, назви яких, надалі, стануть полями таблиць бази даних. Застосування даного підходу тягне і ряд проблем таких, як втрата даних після оновлення моделі, що в бізнес системах попросту не допустимо.

Дана проблема, наприклад, вирішується шляхом створення резервної копії бази даних, з якої згодом можна витягти вже існуючу інформацію, проте це являє собою певні проблеми, оскільки для синхронізації баз даних доведеться виділяти час на написання окремих модулів, які будуть цим займатися, що не багато замовників зможуть схвалити. Найбільш частіше застосованим є зміна назв полів, сучасні ORM дозволяють змінити імена без втрати даних, зі зміною типу складніше.

Database First – основною ідеєю якого є в першу чергу створення бази даних, а потім, на її основі, згенерувати модель даних програми за допомогою ORM. Для отримання контексту бази даних потрібно використати вбудовані засоби для отримання класів з таблиць. Контекст бази даних це своєрідний шар доступу до даних, що володіє CRUD (Create, Read, Update, Delete) можливостями. Даний файл являє собою набір взаємозалежних проксі-класів, які відображають модель бази даних. Даний підхід покладає проблему створення моделі на ORM, а також дозволяє програмно викликати збережені процедури з генерованого коду, що істотно заощаджує час [4].

Дана можливість відсутня в підході Code First, і змушує розробників вручну додавати додатковий код, який буде працювати з збереженими процедурами. Даний підхід є противагою Code First, проте модифікація

структури моделі даних при такому підході істотно утруднена – через необхідності оновлення всієї структури даних програми цілком. Також, код, згенерований ORM, має багато сторонніх об’єктів, які погіршують швидкість доступу до даних.

Були розглянуті два найбільш гнучких підходи підтримуваних сучасними ORM, для створення сховища даних для застосунків, заснованих на веб-платформі. Виходячи з переваг і недоліків даних підходів, можна вибрати найбільш підходящу модель для розробки. В якості моделі розробки виберемо Code First оскільки можливі часті зміни в моделях, а робити це вручну довго.

Наступним важливим кроком є вибір власне ORM для використовування в проекті найбільш популярним для мови проекту C# є Hibernate і Entity Framework. Хоча це певним чином випливає з обраної бази даних, оскільки була обрана Microsoft SQL Server то логічним є вибір Entity Framework.

1.5.4 ORM

У об’єктно-орієнтованому програмуванні об'єкти в програмі представляють об’єкти з реального світу. Як приклад можна розглянути адресну книгу, яка містить список людей разом з кількома телефонами і кількома адресами.

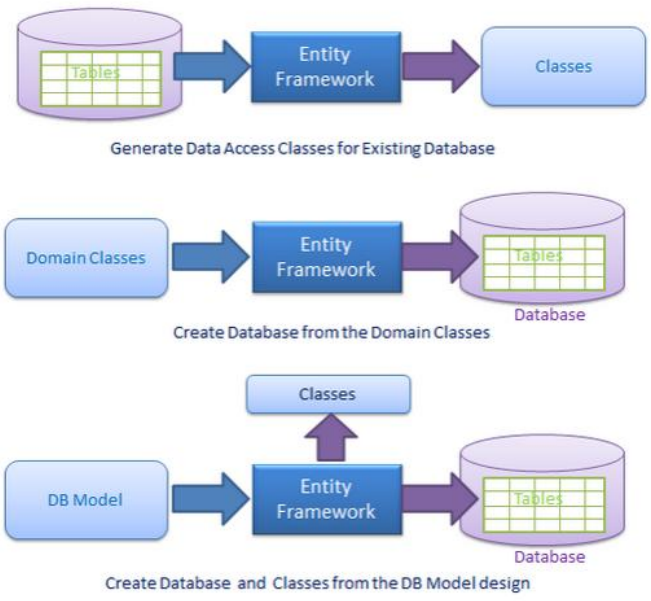
Суть проблеми полягає в перетворенні таких об’єктів у форму, в якій вони можуть бути збережені у файлах або базах даних, і які легко можуть бути витягнуті в подальшому зі збереженням властивостей об’єктів і відносин між ними.

Одним з вирішенням такої задачі є використання реляційної бази даних. Але навіть такий підхід є дещо нераціональним. Адже потрібно провести безліч об’єднань таблиць (join). Також, щоб добратися до такий даних потрібно виконати багато запитів. А це призводить до нагромадження коду.

Написання ADO.Net коду для доступу до даних є виснажлива і монотонна робота. Microsoft створила ORM фреймворк з назвою «Entity Framework» (EF) для автоматизації діяльності пов'язана з базою даних для програм [4].

Microsoft дав таке визначення Entity Framework: Microsoft ADO.NET Entity Framework є об'єктно реляційний відображення (ORM) фреймворк, який дозволяє розробникам працювати з реляційними даними як з об'єктами, усуваючи необхідність для тривіального коду доступу до даних, що розробнику, як правило, потрібно написати самому. Використання Entity Framework робить запити, використовуючи Linq, а потім отримавши і перетворивши дані в сильно типізовані об'єкти. Реалізація ORM платформи Entity Framework надає послуги, такі як відстежування змін, lazy(ліниве) завантаження, і переклад запиту (на різні типи СУБД), так що розробники можуть зосередитися на своєму конкретного додатка бізнес-логіки, а не основам доступу до даних [4].

Entity Framework є корисним у трьох сценаріїв. По-перше, якщо у вас вже є існуюча бази даних, або необхідно, щоб проектування бази даних було спочатку. По-друге, щоб зосередитися на класах предметної області, а потім створити базу даних з класів моделей. По-третє, створити свою схему бази даних на візуальному конструкторі, а потім створити класи.



‘

Рисунок 1.1 – Представлення варіантів роботи EF

Відповідно до рисунка 1.1, EF створює класи доступу до даних для існуючої бази даних, так що можна використовувати ці класи для взаємодії з базою даних, а не ADO.Net безпосередньо.

EF також може створювати базу даних з класів моделей, таким чином, можна зосередитися на бізнес логіці.EF пропонує модель конструктора, де можна проектувати моделі бази даних, а потім EF створює базу даних і класи на основі БД моделі [4].

Оскільки використаний підхід Code First то для коректної роботи і створення таблиць потрібно використати анотацію (атрибути), звичайно можна без них але тоді можлива втрата даних коли будуть необхідні ключі з різних таблиць.

Також необхідно включити функцію автоматичної міграцію даних, коли поля в моделі даних змінюються, за допомогою команди в консолі диспетчера пакетів NuGet «Enable-Migrations –true». Для автоматичного оновлення бази даних при зміні моделі, це можна робити і вручну. Також це команда створить папку Migrations з класом Configuration де є метод Seed який виконується кожного разу після оновлення бази даних за допомогою команди «Update-Database».

При використані класів даних (також відомий як класи сутностей) можна застосовувати атрибути до класу або членів, які визначають правила перевірки, і встановити відносини між класами. EF містить класи, які використовуються в якості атрибутів даних. Застосовуючи ці атрибути до класу або полів, централізувати визначення даних і не доведеться повторно застосувати ті ж атрибути в декількох місцях.

Власне даний спосіб найбільш простий для розуміння. Досить призначити необхідний атрибут класу або властивості і це відповідним чином відіб’ється при генерації таблиць в базі даних.

У коді для використання атрибутів необхідно підключити простір імен System.ComponentModel.DataAnnotations.

‘

[Table (string name, Schema = string)] використовується для класів і визначає довільне ім’я (name) для створюваної таблиці. Іменований параметр Schema дозволяє вказати схему, використовувану для її генерації [4].

[Column (string name, Order = int, TypeName = string)] атрибут застосовується для характеристики класів Моделі і визначає ім’я колонки (name), пов’язаної з ним. Крім того, іменовані параметри дозволяють: Order – задати її порядковий номер у таблиці, TypeName – уточнити тип, який буде використовуватися в таблиці для даної властивості [4].

[Key] дозволяє позначити одну або декілька властивостей, які будуть розглядатися як первинний ключ для створюваної таблиці. Атрибут застосовується лише до простих типів [4].

[Required] вказує на те, що дана властивість і відповідне поле в базі даних обо’язково повинні містити значення, тобто не можуть бути рівні null.

[MaxLength (int length)] даний атрибут встановлює максимальну довжину масиву (length) і може бути використаний тільки з властивостями типу byte [] або string. параметра length [4].

[ForeignKey (string name)] вказує навігаційну властивості пов’язана з ним властивість name, яке є зовнішнім ключем [4].

[DatabaseGenerated (type)] атрибут зазначає властивість, для якого значення в пов’язаної з ним колонці таблиці повинно обчислюватися самою базою даних. Його параметр databaseGeneratedOption може встановлювати один з наступних варіантів генерації значень: DatabaseGeneratedOption.Identity тільки при додаванні рядка (такий варіант, як правило, за замовчуванням використовується для первинних ключів в таблицях), DatabaseGeneratedOption.Computed при додаванні або оновленні рядка, DatabaseGeneratedOption.None – генерація значення для даної рядки відключена [4].

[JsonIgnore] вказує що при серіалізації об’єкта властивість не буде перетворена в Json, можливе застосування до декількох властивостей. Для використання необхідно підключити Newtonsoft.Json який завантажується окремо [4].

‘

[DataType(DataType)] більш конкретизує тип стовпця в базі даних, за замовчування використається тип найближчий до типу. Якщо тип вказано використовується той тип і застосовує, автоматично, максимальний розмір поля [4].

Клас DbContex є базовим класом який містить всі основні методи і властивості, які використовуються для відстежування змін, які пов’язані з контексом даних. Також в класі повинно бути поле DbSet<>, за яким і буде відслідковуватися ці зміни.

DbContext зазвичай використовується в похідному класу, який містить DbSet <TEntity> властивість яка містить, як TEntity клас моделі. Ці набори автоматично ініціалізують коли створюються екземпляри похідного класу Така поведінка може бути змінено шляхом застосування SuppressDbSetInitializationAttribute прописати або весь отриманий класу або окремих властивостей класу. Модель може бути створенами декількома способами. При використанні Code First підхід, то DbSet <TEntity> використовуються властивості для побудови моделі по конвенції (атрибути які описувалися вище). Метод OnModelCreating можна перевизначити для налаштування цієї моделі.

Підключення до бази даних (включаючи ім’я бази даних) може бути зроблено в декілька способів. Якщо без параметрів DbContext конструктор викликається з похідного класу, то ім’я похідного клсу використовується, щоб знайти рядок підключення у файлі app.config або web.config. Якщо рядок з’єднання не знайдена, то ім’я передається в DefaultConnectionFactory зареєстровано в класі Database. Підключення потім використовує ім’я класу в якості імені бази даних у рядку з’єднання за замовчуванням. (Цей рядок з’єднання за замовчуванням вказує SQLEXPRESS на локальній машині, якщо не перевизначена DefaultConnectionFactory.)

Замість того щоб використовувати отримане ім’я класу, ім’я з’єднання база даних також може бути вказаний явно, передаючи ім’я одного з конструкторів DbContext, який приймає рядок. Назва також може бути переданий у вигляді «name= Myname», в цьому випадку ім’я має бути

знайдено в конфігураційному файлі або буде виняток. Зверніть увагу, що підключення знаходиться у файлі app.config або web.config може бути звичайний рядок підключення до бази даних (не виключна зв'язок рядок Entity Framework) в цьому випадку буде використовувати DbContext Code First. Однак, якщо з'єднання знаходиться у файлі конфігурації є особливий рядок підключення Entity Framework, то DbContext буде використовувати Database/Model First. Існуючих або явно створено DbConnection також може бути використаний замість імені бази даних (підключення). DbModelBuilderVersionAttribute може бути застосований до класу, наслідованого з DbContext встановити версію бази даних, використовуваних в класі, коли він створює модель. Якщо атрибут не застосовується, то остання версія конвенцій використовуватиметься [4].

Тепер розглянемо основні властивості, методи класу, які будуть використані тут. Метод SaveChanges дозволяє змусити зберегти всі зміни зроблені в моделі. Метод Entry дозволяє вказати стан об'єкта (використовується для збережені), коли встановлено флаг змін.

ChangeTracker дозволяє настроювати як клас буде реагувати на зміну даних які прив'язані через DbSet. Також метод DetectChanges дозволяє сказати класу щоб всі дані, які були внесені потребують збереження.

Configuration дозволяє настроїти роботу з сутностями (об'єктами класу). Як буде відбуватися, отримання даних з бази даних (LazyLoadingEnabled) чи будуть отримані всі сутності (класи) пов'язані з ним, за замовчуванням false. AutoDetectChangesEnabled дозволяє визначити чи EF буде автоматично оновляти базу даних. ValidateOnSaveEnabled дозволяє вказати чи буде перед збереженям виконана перевірка моделі, дозволяє уникнути sql помилок.

Database створює екземпляр бази даних для цього класу, що дозволяє створення, видалення, перевірку на існування члені в базі даних. Основні використані методи ExecuteSqlCommand дозволяє виконати SQL запит до бази даних SqlParameter дозволяє перетворювати об'єкти на типи які приймає база даних.

‘

Також клас DbSet має декілька важливих методі для маніпулювання даними. Які не додають/видаляють дані одразу, а тільки після виклику SaveChanges, тобто дані які були видалені до виклику, будуть існувати в базі даних.

Add додає заданий об’єкт в контексті (DbSet) ставлячи позначку стану Added. Якщо зміни зберігаються, об’єкти з Added статусом зберігаються в базу даних. Після збереження змін, змінюється стан об’єкта Unchanged.

Remove визначає даний об’єкт зі статусом Deleted. Коли зміни будуть збережені, об’єкт видаляється з бази даних. Об’єкт повинен існувати в контексті в іншому стані, перш ніж цей метод викликається.

Find використовує значення первинного ключа, щоб знайти сутність. Якщо об’єкт не знаходиться в контексті, то запит буде виконаний і перетворений за даними в джерелі даних, і повертається NULL, якщо об'єкт не виявлений в контексті, або в джерелі даних. Зверніть увагу, що пошук також повертає об’єкти, які були додані в контексті, але ще не були збережені в базі даних.

DbSet також застосовується для фільтрування даних за допомогою Linq.

1.5.5 OData

Мета протоколу відкритих даних (далі, як OData) є забезпечення протоколу REST основі для операцій CRUD (створення, читання, оновлення та видалення) для ресурсів у вигляді сервісів передачі даних. «Послуга передачі даних» є кінцевою точкою, де є дані від однієї або декількох «колекції» кожного з нуля або більше елементів, які складаються з типізованих пар ім’я-значення. OData випускається корпорацією Майкрософт в рамках OASIS (Організація з поліпшення стандартів структурованої інформації). Так що кожен, хто хоче, може побудувати сервер з використання OData [4].

Оскільки краще, зберігати дані у структурованому форматі, будь то на електрону обчислювальній машині, мобільному пристрох, завжди потрібні

стандартизовані інтерфейси для роботи з даними в цьому форматі. Якщо дані (реляційні) SQL надає набір операцій над даних. Навіть дані, які є реляційні не часто піддаються для використання у звітності за допомогою SQL. OData є веб-еквівалент ODBC, OLEDB, ADO.NET і JDBC.

Технічно OData протокол визначає, як стандартизувати типізований, ресурсів-орієнтований інтерфейс, надаючи колекцію записів, які повині відповідати певному наборі правил. Крім того, він висвітлює модель даних для визначення типізованих або нетипізовані значення на вході (наприклад, кількість записів), а також дозволяє отримувати тільки ті записи і дані, які ми хочемо. OData повинна підтримувати принаймні один з формат Json.

Документ метаданих описує типи, набори, функції і дії зрозумілі службою OData. Клієнти можуть використовувати цей документ метаданих щоб зрозуміти, які опції доступні в службі.

OData складається з динамічних ресурсів. URL-адреси для багатьох з цих ресурсів може бути обчислена з інформації, що міститься в документі метаданих.

Розглянемо основні параметри. Для застосування параметрів необхідно вказати «?» символ, для комбінації символ «&», кожна опція починається з символу «$» [4].

Expand розгорнути відповідні ресурси відповідно до отриманих ресурсів, наприклад, Categories/$expand=Products додасть дані продуктів відповідно до кожного запису Категорія [4].

Select обмежити вибірку даних тільки полями які необхідні, наприклад, $select=Category Name,Description [4].

Orderby один або декілька розділених комами виразів з додатковим «asc» (за замовчуванням) або «desc» (спадаючий), залежно від того, як значення упорядковано наприклад, $orderby=CategoryName desc.

Top повернення записів з верхньої частини списку, наприклад, $top=4.

Skip скільки записів пропустити, наприклад, $skip=4 [4].

Count логічне значення true або false запитом. Кількість записів які відповідають запиту, наприклад, $count=true.

Search вираз пошуку довільного тексту, що відповідає будь-якому значенню в будь-якому полі, елемент буде включений до вибірки, $search=blue OR green.

Filter логічний вираз для конкретний елемент повинен бути включений до вибірки, наприклад $filter=CategoryName eq 'Produce'. Для опису фільтру використовується OData вираз [4].

Основні застовані OData вираження відповідають звичайним операціям порівняння. Рівний eq («назва поля» eq «значення») працює як з числами так і з рядками. Більший або рівне ge («назва поля» ge «значення») працює тільки з числами і датами [4]. Substring (Substring («назва поля», «значення»)) використовує функція для порівняння входження рядка. Приклад методу контролера представлений в лістингу 1.2.

Лістинг 1.2 – Метод який використовує OData

```
        [HttpGet]
        [Route("institutions")]
       public Object FindWithQuery(ODataQueryOptions<Institution>
options)
            {
           PageContainer page = manager.FindWithQuery(options);

                page.QueryCount = Request.GetInlineCount();
                return page;
             }
```

Даний метод описує отримання даних з бази даних використовуючи OData, для фільтрування даних.

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ТА РОЗРОБКА БІЗНЕС-ЛОГІКИ

2.1 Розробка бази даних

2.1.1 Побудова UML-діаграми класів для взаємодії з базою даних

Провівши аналіз предметної області, ми можемо виявити основні сутності предметної області. Оскільки мета роботи - це створення системи ведення обліку та статистики продаж, то перша сутність, яку можна виявити - це запит на товар (Product) (рис. 2.1).

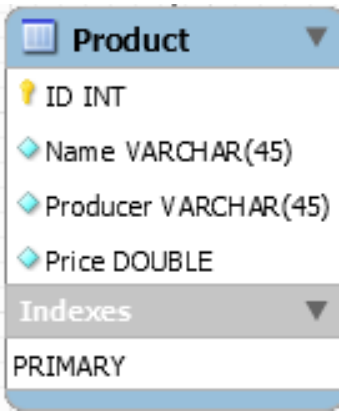


Рисунок 2.1 – таблиця сутності p.

Таблиця складається лише з найнеобхіднішої інформації про запит: Name – назва товару, Producer – виробник товару, Price – ціна товару.

Для дотримання нормальних форм [2], кількість була винесений у окрему сутність – number_product (рис. 2.2).

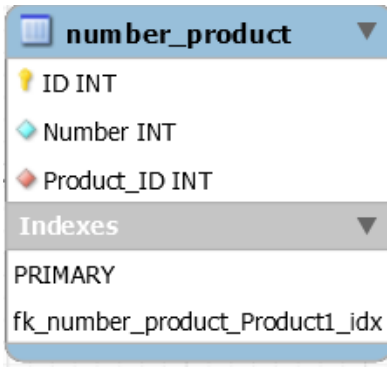


Рисунок 2.2 – таблиця сутності number_product.

‘

Таблиця складається з : Number – кількість товару, Product_ID – id товару, якому відповідає певна його кількість.

Для зберігання даних про товар, який було продано була створена сутність – Product_sold (рис. 2.3).

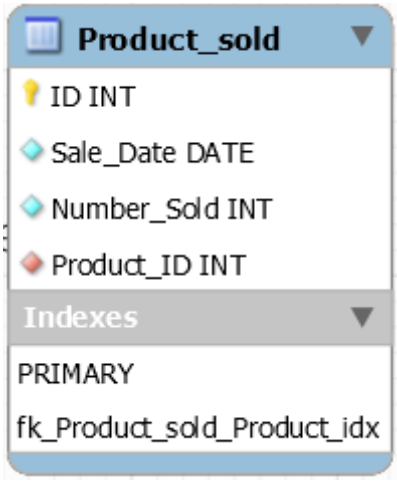


Рисунок 2.3 – таблиця сутності Product_sold.

Таблиця складається з : Sale_Date – дати продажу товару, Number_Sold- кількості проданого товару , Product_ID – id товару, якому відповідає проданий товар.

Для відокремлення даних про користувачів системи була створенна сутність – User (рис. 2.4).

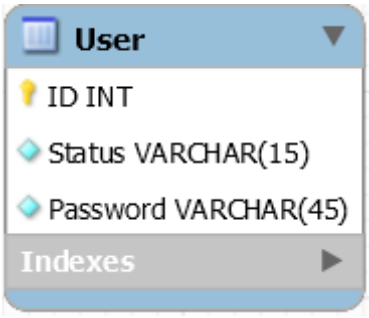


Рисунок 2.4 – таблиця сутності User.

Таблиця складається з: Status – статус користувача, Password – пароль користувача.

Під час аналізу предметної області були виявлені наступні сутності із такими відношеннями:

Сутність Product має відношення з number_product.

Оскільки одному товару може належати тільки одна кількість і навпаки, то відношення між сутностями було визначено як «один до одного» (рис. 2.5).

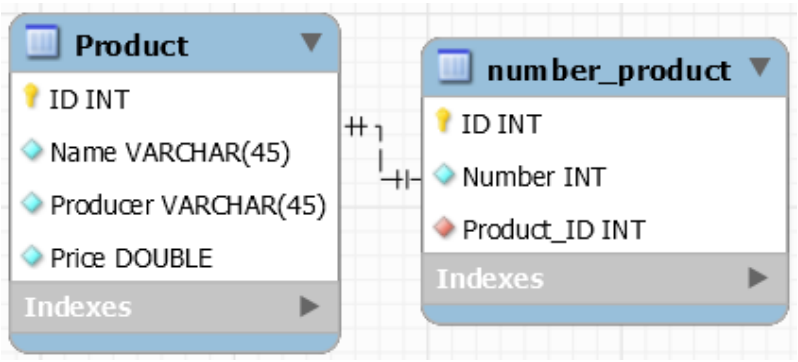


Рисунок 2.5 – відношення між сутностями Product та number_product.

Для реалізації даного відношення у таблиці number_product поле Product_ID було позначено як зовнішній ключ для таблиці Product.

Сутність Product має відношення з сутністю Product_sold.

Оскільки один товар може бути кілька разів проданий, то відношення між ними було встановлено як «один до багатьох» (рис 2.6).

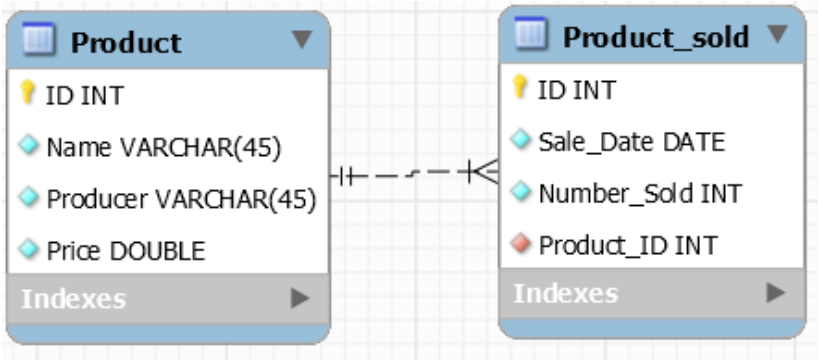


Рисунок 2.6 – схема відношень таблиць Product та Product_sold.

Для реалізації даного відношення у таблиці Product_sold поле Product_id було позначено як зовнішній ключ для таблиці Product.

Після виявлення усіх сутностей і зв'язків між ними, можна побудувати загальну схему бази даних (рис. 2.7).

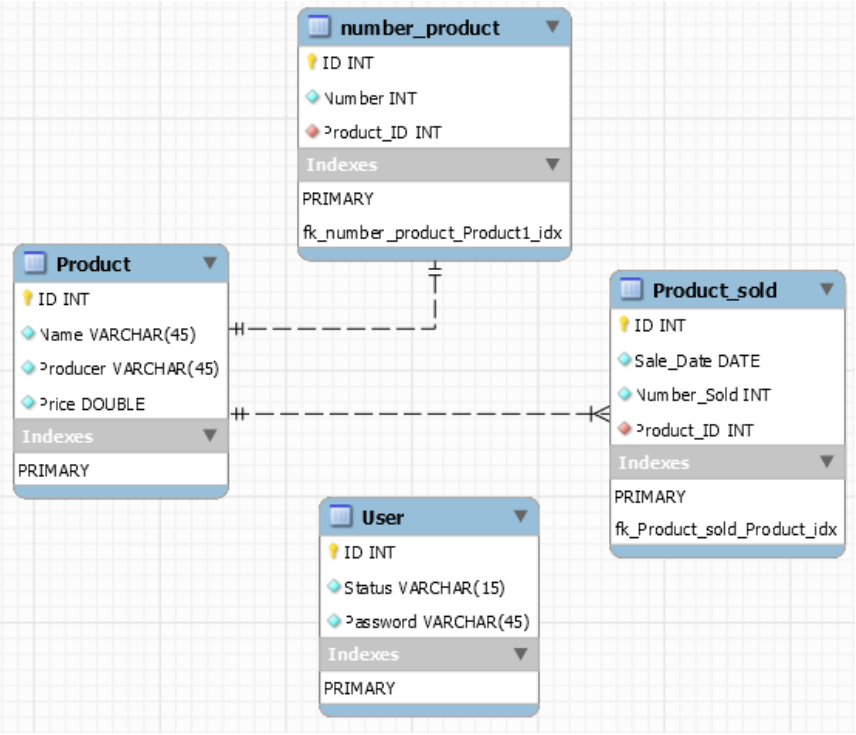


Рисунок 2.7 – повна схема сутностей та їх зв’язків.

2.1.2 Класи використані для бази даних

Класи для авторизації і автентифікації ASP.NET забезпечує стандартним набором класів для цього, цей пакет називається IdentityUserS. Який складається з чотирьох класів: IdentityUsers, IdentityUsersRole, IdentityUsersLogin, IdentityUsersClaim, для нас важливі перші два, які є базовими для авторизації, інші два відповідають за зовнішнє логінування. Дані класи описані в метаданих, тому тут будуть надані тільки основні поля для користувача.

Перший клас IdentityUserS відповідає таблиці AspNetUser, який включає основні дані для логіну і реєстрації, такі як: Id головний ідентифікатор представлений типом даних стрічка, відповідає за ім’я користувача, якщо таке не надано, EmailConfirmed властивість логічного типу, яка відповідає за перевірку на існування однакового emeil, PaswordHash пароль користувача у вигляді хешу, UsersName – ім’я користувача для входу.

Даний клас відповідає за збереження даних про користувача.

‘

Наступний клас IdentityUsersRole, якому відповідає таблиці AspNetRole, включає основні дані для ролей користувачів, такі як Id та Name

.

Для позначення даних, що будуть завантажуватися тільки при необхідності, використовують ключове слово virtuol означає lazy-завантаження даних, використовується для навігаційних властивостей.

Наступна частина описує класи для збереження даних, вони всі унаслідуються від базового класу EnityBase. Код його наведений в лістингу 2.1.

Лістинг 2.1 – Базовий клас для моделей

```
public class EnityBase
 {
     [Key]
   [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
     public long Id { get; set; }

     [Timestamp]
     public byte[] Version { get; set; }
 }
```

Цей клас має два поля Id, яке є генерованим тоді, коли йде збереження нової сутності і виконує роль первинного ключа, тому властивість Version використовується для логіки при апдейті моделі вона кожного разу переписується при успішному збережені даних.

Клас Instution застосовується для збереження даних про робочу кастину і деякі додаткові функції для програми при виборі робочої установи. Також тут присутні властивості для виводу кількості товару . Код класу наведений в лістингу 2.2.

Лістинг 2.2 – Код класу для торгівельної структури

```
[Table("Instutions")]
 public class Institution : EntityBase
 {
     public string FullNames { get; set; }

     [StringLength(120)]
     [DataType(DataType.Text)]
     public string Acronym { get; set; }
```

‘

```
            [DataType(DataType.Html)]
            public string Description { get; set; }

            [DataType(DataType.ImageUrl)]
            public string ImageUrl { get; set; }

            [JsonIgnore]
            public virtual ICollection<Account> Accounts { get;
set; }
        }
```

Цей клас має такі властивості: FullNames повна назва, що використовується для пошуку, Acronym - назва для відображення в пошуку, Descriptions поле для опису буде відображена при перегляді деталей.

Клас Comments застосовується для збереження даних про коментарі написані користувачем. Код класу наведений в лістингу 2.3.

Лістинг 2.3 – Код класу коментарів

```
[Table("Comments")]
public class Comment : EntityBase
{
    [Required]
    [MaxLength]
    [DataType(DataType.Html)]
    public string Description { get; set; }

    [ForeignKey("Account")]
    public long OwnerId { get; set; }

    [JsonIgnore]
    public virtual Account Account { get; set; }

    [Required]
    [ForeignKey("Course")]
    public long CourseId { get; set; }

    [JsonIgnore]
    public virtual Course Course { get; set; }

    public DateTime CreatedDate { get; set; }
}
```

Цей клас має такі властивості: Descriptions коментар в форматі html, , OwnerId зовнішній ключ для акаунтів, CourseIds зовнішній ключ для курсу, до якого коментар, CreatedDate дата створення коментаря.

‘

Клас Accounts застосовується для збереження даних про користувачів більш розширено.

Код класу наведений в лістингу 2.4.

Лістинг 2.4 – Основні властивості класу акаунтів

```
[Table("Accounts")]
public class Accounts : EntityBase{
    [Required]
    [JsonIgnore]
    public string UsersId { get; set; }

    [Required]
    [DataType(DataType.EmailAddress)]
    public string Email { get; set; }

    [Required]
    public string UsersName { get; set; }

    [Required]
    public UsersGroup GroupType { get; set; }

    [ForeignKey("Instutions")]
    [Required]
    public long InstitutionIds { get; set; }

    public Institution Institution { get; set; }

    [Required]
    [StringLength(120)]
    public string SurName { get; set; }

    [Required]
    [StringLength(120)]
    public string Name { get; set; }

    [StringLength(120)]
    public string MiddleName { get; set; }

    [DataType(DataType.Html)]
    public string Descriptions { get; set; }

    [DataType(DataType.PhoneNumber)]
    public string Number { get; set; }

    [Required]
    public string GroupName { get; set; }
}
```

‘

Цей клас має такі властивості: UsersId зовнішній ключ для класу користувачів, Email користувача буде оновлети також в користувача, GroupType тип користувача може бути продавцем або адміністратором, InstitutioId зовнішній ключ для установи , UsersName - ім’я користувача при вході, SureName прізвище користувача, Names ім’я користувача, MiddleName по-батькові користувача, Number номер, GroupName поточна група, Description коротка інформація про товар.

Лістинг 2.5 – Код класу курси

```
[Table("Courses")]
public class Course : EntityBase
{
    [Required]
    [StringLength(120)]
    public string Title { get; set; }

    [Required]
    [DataType(DataType.MultilineText)]
    [StringLength(250)]
    public string Annotation { get; set; }

    [Required]
    [DataType(DataType.Html)]
    public string Description { get; set; }

    [DataType(DataType.ImageUrl)]
    public string ImageUrl { get; set; }

    [Required]
    [ForeignKey("Professor")]
    public long ProfessorId { get; set; }

    [JsonIgnore]
    public virtual Account Professor { get; set; }

}
```

Цей клас має такі властивості: Titles назва курсу користувачів, ImageUrl посилання на зображення для відображення курсу в списку, ProfessorId зовнішній ключ для класу користувачі, Description поле для відображена при перегляді деталей про курси формату html, Annotation коротка інформація про курс.

‘

Клас Advertisments застосовується для збереження даних про оголошення, які створюють користувачі для курсів. Код класу наведений в лістингу 2.6.

Лістинг 2.6 – Код класу оголошень

```
[Table("Advertisements")]
 public class Advertisements : EntityBase{
     [Required]
     public string Title { get; set; }

     [Required]
     [DataType(DataType.Html)]
     public string Description { get; set; }

     [Required]
     [ForeignKey("Owner")]
     public long OwnerId { get; set; }

     [JsonIgnore]
     public virtual Account Owner { get; set; }

     [ForeignKey("Course")]
     public long? CourseId { get; set; }

     [ForeignKey("Institution")]
     public long? InstitutionId { get; set; }

     [Required]
     public Priotity Priority { get; set; }

     [Required]
     public DateTime RelevantDate { get; set; }
 }
```

Цей клас має такі властивості: Title заголовок оголошення користувачів, OwnerId зовнішній ключ для класу користувачів, Description поле для відображена при перегляді деталей про оголошення формату html, CourseId зовнішній ключ для курсу потрібен або група або курс, GroupId зовнішній ключ для групи потрібен або група або курс, Priority пріоритет оголошення відображається по різному в залежності від нього, RelevantDate дата доки оголошення є актуальним.

Клас ApplicationsCours застосовується для збереження даних про заявки до курсів і є проміжним класом між акаунтами і курсами. Код класу наведений в лістингу 2.7.

‘

Лістинг 2.7 – Код класу заявки для товрів

```
[Table("ApplicationsCourse")]
public class ApplicationsCourse : EntityBase
{
    [Required]
    [ForeignKey("Course")]
    public long CourseId { get; set; }

    [JsonIgnore]
    public virtual Course Course { get; set; }

    [Required]
    [ForeignKey("Student")]
    public long StudentId { get; set; }

    [JsonIgnore]
    public virtual Account Student { get; set; }

    public DateTime Date { get; set; }

    [Required]
    public RequestStatus Status { get; set; }

    public CourseStatus  UserStatus { get; set; }
}
```

Цей клас має такі властивості: Status статус заявки StutId зовнішній ключ для класу користувачів, Student навігаційна властивість, Date час подання заявки, CountId зовнішній ключ для курсу, UsersStatus внутрішній статус для заявки (випускник, ігнорування при вибірці, студент).

Клас ColumnHeader (заголовки таблиць) застосовується для збереження даних заголовки таблиці журналу для конкретної групи, дозволяє записувати будь-який заголовок. Код класу наведений в лістингу 2.8.

Лістинг 2.8 – Код класу заголовки таблиць

```
[Table("ColumnHeaders")]
public class ColumnHeader : EntityBase
{
    [Required]
    [ForeignKey("Counte")]
    public long CourseId { get; set; }

    [JsonIgnore]
    public virtual Course Course { get; set; }
```

‘

```
        [Required]
        [ForeignKey("Group")]
        public long GroupId { get; set; }

        [JsonIgnore]
        public virtual Group Group { get; set; }

        [Required]
        public DateTime ColumnDate { get; set; }

        [Required]
        public String Title { get; set; }

        [JsonIgnore]
        public ICollection<Mark> Marks { get; set; }}
```

Цей клас має такі властивості: CountId зовнішній ключ для курсу, GroupId зовнішній ключ для групи, ColumnDate дата стовпця (йде сортування), Title заголов стовпця, Marks список всіх оцінок.

Клас Mark застосовується для збереження даних про відмітки товарів певних категорій. Код класу наведений в лістингу 2.9.

Лістинг 2.9. Код класу відмітки.

```
[Table("Marks")]
public class Mark : EntityBase
{
    [ForeignKey("Column")]
    public long ColumnHeaderId { get; set; }
    [JsonIgnore]
    public virtual ColumnHeader Column { get; set; }

    [ForeignKey("Course")]
    public long CourseId { get; set; }

    [JsonIgnore]
    public virtual Course Course { get; set; }

    [ForeignKey("Student")]
    public long StudentId { get; set; }

    [JsonIgnore]
    public virtual Name Student { get; set; }

    [Required]
    public int Value { get; set; }
}
```

‘

Цей клас має такі властивості: Value поточна відмітка будь-якого формату, StutId зовнішній ключ для класу користувачів, ColumnsHeaderId зовнішній ключ для заголовку стовпця, CountId зовнішній ключ для курсу, StuId зовнішній ключ , Name навігаційна властивість.

Клас Names застосовується для збереження даних про назви товарів. Код класу наведений в лістингу 2.10.

Лістинг 2.10 – Код класу імен

```
[Table("Names")]
public class Name : EntityBase
{
    [ForeignKey("Student")]
    public long? StudentId {get; set;}

    [JsonIgnore]
    public virtual Account Student { get; set; }

    [StringLength(350)]
    public String FullName { get; set; }
}
```

2.2 Розробка бізнес-логіки

2.2.1 C#

У комп’ютерному програмуванні, інтерфейс прикладного програмування (API), це набір підпрограм, протоколів та інструментів для створення додатків. API утворює програмний компонент з точки зору його операцій читання, запису даних. API визначає функції, які не залежать від їх відповідних реалізацій, що дозволяє визначення та реалізації змінювати її без шкоди один одному. Добре написане API дозволяє легше розробляти програму, надаючи всі блоки. Програміст, ставить блоки разом [4].

API часто розповсюджують у вигляді бібліотеки, яка включає специфікації для процедур, структур даних, класів об’єктів і змінних. В інших випадках, зокрема SOAP і REST послуг, API просто специфікація віддалених викликів, для користувачів API.

Web API представляє собою інтерфейс прикладного програмування (API), як для веб-сервера та веб-браузера.

‘

Серверний частина Web API являє собою програмний інтерфейс до певної системи визначених запитів відповідей, як правило, виражається в Json або XML, який передається через веб – найбільш часто за допомогою веб-сервера на основі HTTP. Гібридні додатки є веб-додатки, які поєднують в собі використання декількох таких Web API. Webhooks є серверні веб-інтерфейси, які беруть в якості вхідних URI, який призначений для використання як дистанційну адресу або як зворотній виклик, коли сервер виступає в якості клієнта перенаправити URI та ініціювати подія на іншому сервер, який обробляє цю подію таким чином забезпечуючи зв’язок.

У той час як Web API в цьому контексті іноді вважається синонімом для веб-служби, Web 2.0 веб-додатків відійшли від SOAP веб-сервісів до більш згуртованих колекцій RESTful веб-ресурсів. Ці RESTful веб-інтерфейси доступні через стандарт методи HTTP різними HTTP клієнтів, включаючи браузери і мобільні пристрої.

На стороні клієнта Web API є програмний інтерфейс для розширення функціональності в рамках веб-браузера або іншого клієнта HTTP. Спочатку це було у вигляді плагінів, проте більшість нових розробок використовують прив’язку за допомогою JavaScript [6].

Стек .NET вже включає в себе ряд технологій, які забезпечують можливість створення HTTP (серверних додатків), і це було зроблено так, з самого початку платформи .NET. Від простих HTTP обробників і модулів в середовищі ASP.NET, до платформ на високому рівні, як ASP.NET MVC, Web Forms, ASP.NET AJAX і REST (які технічно не ASP.NET, але можна інтегрувати з ним), завжди можна обробити практично будь-який вид запиту HTTP, і дати відповіді з ASP.NET. Зручність платформи ASP.NET є те, що вона має все що потрібно побудувати майже будь-який тип HTTP програми починаючи з API, низького рівня закінчуючи високо рівневими HTML генераторами [4].

ASP.NET Web API диференціює себе від попередніх рішень Microsoft в області HTTP застосунків, що він був побудований з нуля над протоколом HTTP і його семантики обміну повідомленнями. На відміну від REST або

ASP.NET AJAX, це нова платформа, яка працює в контексті існуючого фреймворку. Перерава ASP.NET WEB API є те, що він поєднує в собі кращі риси платформ, які були до нього, щоб забезпечити комплексну і дуже зручну HTTP платформу. Тому що вона заснована на ASP.NET і використовує багато понять з ASP.NET MVC, Web API повинні бути знайомим і зручним для більшості розробників ASP.NET.

Ось деякі з особливостей, які Web API передбачає:

1) хороша підтримка для URL маршрутизації, щоб створювати чисті URL, використовуючи знайому семантику маршрутизації стилю MVC;
2) Різний вміст відповіді на основі заголовків запиту для серіалізації;
3) підтримка безлічі форматів виводу, включаючи Json, XML, та інші;
4) підтримка за замовчуванням REST, не є обов'язковими;
5) легко розширювана підтримку форматування, щоб додати нові вхідні (вихідні) типи;
6) підтримка для більш просунутих функцій HTTP за допомогою HttpResponseMessage і HttpRequestMessage;
7) класи і строго типізованих списки для опису операцій HTTP;
8) розширюваний, заснований на MVC, як модель для розширення форматерів і фільтрів;
9) легко тестовані, поняття тестування, схожі на MVC.

Web API призначений для обробки будь-яких вхідних HTTP і створювати вихідні запити, використовуючи повний спектр функціональності HTTP доступу в гнучкій основі.

Переглядаючи список вище можна бачити, що багато функціональності дуже схожа на ASP.NET MVC, тому багато розробників ASP.NET повинні відчувати себе цілком комфортно з поняттями Web API. Маршрутизація та базова структура Web API дуже схожі на те, як MVC працює забезпечуючи багато з переваг MVC, але з акцентом на доступ HTTP і маніпуляції в методах контролера, а не на генерації HTML в MVC.

‘

Там набагато покращена підтримка різного формату контенту на основі HTTP заголовкові з полем (accept) з структури, що дозволяє виявити який формат даних буде використаний для відправки клієнту. Це здається такою маленькою і очевидною річчю, але це дійсно важливо. Сьогоднішні сервери часто використовуються декількома клієнтами (додатками) і можливість вибрати правильний формат даних, що найкраще підходить для клієнта.

У той час як попередні рішення були в змозі зробити це, використовуючи різні змішані особливосте ASP.NET, Web API поєднує в собі всі ці функціональні можливості в одному фреймворк HTTP на стороні сервера, що розуміє HTTP семантику і тонко направляє в правильному напрямку. І коли потрібно налаштувати або зробити щось своє, є багато способів і можливо перевизначення для більшості поведінок, які дозволяють підключити власну функціональність з відносно невеликим зусиллям.

Це і є основні переваги через які використовуються Web API в цьому проекті.

2.2.2 Основні налаштування API

Для налаштування ASP .Net Web API, використовується наступні файли: RouteConfig, FilterConfig, BundleConfig, WebApiConfig, StartupAuth. Також є необхідний файл web.config, первинний файл конфігурації для будь-якої веб-служби. Оскільки з’єднання реалізовані у вигляді веб-служб, їх файли web.config повинні мати правильну настройку для того, щоб з’єднання функціонували.

Також варто згадати про CORS технологія сучасних браузерів, яка дозволяє надати веб-сторінці доступ до ресурсів іншого домену. До недавнього часу основним способом подолання обмежень, накладених в same-origin-policy щодо XSS запитів, було використання JSONP. Сам JSONP має непереборні обмеження дозволяє тільки отримання даних GET методом, тобто відправка даних через POST метод залишається недоступною.

Файл RouteConfig не потрібний для WEB API. Він включений за умовчанням в нових проектах Web API, тому що цей шаблон також включає

в себе веб-сторінку MVC (для сторінки індексу, який з’являється, коли відкривається проект в браузері).

Файл FilterConfig глобальні фільтри дій застосовуються до всіх запитів у веб-додатках. Наприклад, можна використовувати глобальні фільтри для перевірок безпеки. В даному проекті не використовуються.

Файл BundleConfig реєструє CSS і JS, так що вони можуть бути об’єднані і зменшені в файли. Для даного проекту не використовується.

Файл WebApiConfig створюється, якщо використовуєте Web API. Це можна використовувати для налаштування конкретних маршрутів, будь-яких налаштувань Web API та обслуговування Web API. Тут вже були внесені деякі зміни код в лістинг 2.11.

Лістинг 2.11 – Внесені зміни до файлу WebApiConfig

```
config.EnableCors(new EnableCorsAttribute("*", "*", "*"));
 config.Services.Replace(typeof(IContentNegotiator),
new JsonContentNegotiator(new JsonMediaTypeFormatter()));
```

В даному коді до стандартних функцій API, додається підтримка OData, також тут через EnableCors дозволяється запити, від інших URL, також йде заміна стандартного серіалізатора для JSON.

Код який наведений в лістингу 2.25, вказаує на використання CamelCaseProperty тобто всі назви будуть з малої букви, і Formatting.Indented який дозволяє перетворити вигляд Json в дерево, інакше в рядок. Далі йде стандартне оголошення для перевизначення формату, і додавання до відповіді загаловку, що вказує на те що це Json.

Файл StartupAuth реєструє зовнішніх постачальників перевірки автентичності. По простому дозволяє вказати зовнішні служби для авторизації.

Тепер розглянемо основні анотації використанні в проекті: Authorize, HttpGet, HttpPost, HttpPut, HttpDelete, HttpPatch, AllowAnonymous, Route(«path»). Для застосування анотацій необхідно вказати їх перел методм (класом) в квадратних дужках ([HttpPut]) [4].

Варто зауважити що кожна анотація яка починається з Http відповідає методу, який описаний, тобто вона дозволяє настроїти ресурс для прийому тільки певного типу запиту. Це спрощує роботу і дозволяє використовувати однакові URL для багатьох функцій, просто вказавжи різний тип запиту.

Authorize вказує фільтр авторизації, який перевіряє запит, чи користувач авторизований.

AllowAnonymous дозволяє доступ до ресурс не авторизованим користувачам.

Route дозволяє визначити шлях до ресурсу, і параметри які будути передаватися з ним.

2.2.3 Авторизація

Авторизація для Web API відбувається за наступною схемою рисунок 2.2.

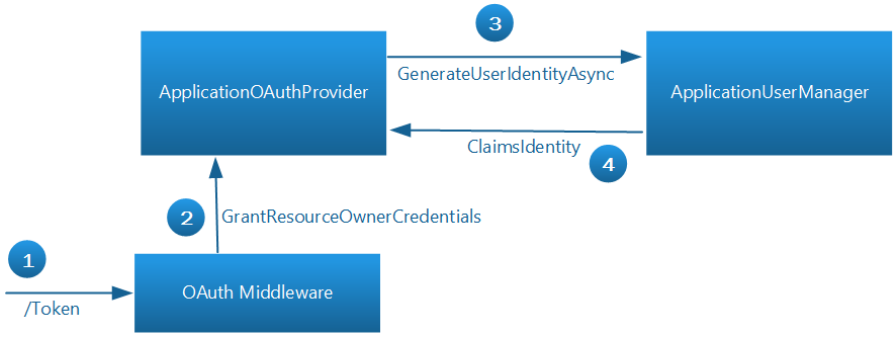


Рисунок 2.2 – Авторизація в Web API

Як видно з рисунку 2.2 ось основні кроки, коли додаток хоче отримати токен: щоб отримати токен доступу, додаток відправляє запит на «/token» (перший крок). OAuth викликає метод GrantResourceOwnerCredentials в провайдері (другий крок). Провайдер використовує ApplicationUserManager для перевірки облікових даних і створення токену (третій крок). Якщо це успішно, провайдер створює квиток перевірки автентичності, яка використовується для генерації токену (четвертий крок) [8].

Нова особливість безпеки для MVC/API Web заснований на Owin аутентифікації. Переваги в тому, що функція безпеки можуть бути розділені

іншими компонентами, які можуть бути розміщені на Owin. Так команда Katana зробила велике зусилля, щоб підтримати інтегрований Owin в ASP.NET, він може також безпечні додатки, розміщеного на IIS, в тому числі ASP.NET MVC, Web API, веб-форми.

Owin визначає стандартний інтерфейс між .NET веб-серверів і веб-застосунків. Мета інтерфейсу Owin є поділ серверів і застосунків, заохочувати розвиток простих модулів для .NET веб-розробки, і, будучи відкритим стандартом, стимулювати розробку відкритим вихідним кодом екосистеми .NET та інструментами веб-розробки.

База OAuth 2.0 дає можливість стороньому додатку отримати обмежений доступ до послуги HTTP сервера. Замість того, щоб, використовувати облікові дані власника ресурсу, клієнт отримує токен доступу. Токен доступу видаються сторонніми клієнтам, сервера авторизації з дозволу власника ресурсу [8].

Перевірка справжності форм використовує квиток додатка, що засвідчує користувача і зберігає його в кукі. Коли користувач вперше отримує доступ до ресурсу, що вимагає дозволу, він буде перенаправляти користувача на сторінку входу. Після користувач вводить облікові дані, код програми буде перевіряти ім’я користувача та пароль і побудує токен, включаючи ім’я користувача, ролі і т.д., також токен шифрується Після надходження токену до сервера, він валідується, і йде перетворення його в об’єкт зрозумілий програмі і видимий всюди [4].

Реєстрація нового користувача доволі проста річ достатньо надіслати дані на відповідне API яке внесе дані до бази даних, при цьому пароль бути зашифрований. Тому краще зупинитися на авторизації де було дописано код, до стандартної авторизації.

Для отримання маркеру (далі токен) необхідно відіслати запит типу POST на ресурс «/token» який поверне токен якщо дані надані правильно, для конфігурування атворизації в файлі StartupAuth необхідно додати код наведений в лістингу 2.12.

‘

Лістинг 2.12 – Код для конфігурації сервера авторизації

```
      PublicClientId = "self";
        OAuthOptions = new OAuthAuthorizationServerOptions
            {
    TokenEndpointPath = new PathString("/token"),
    Provider = new ApplicationOAuthProvider(PublicClientId),
    AccessTokenExpireTimeSpan = TimeSpan.FromDays(24),
                AllowInsecureHttp = true,

            };
```

TokenEndpointPath є URL-шлях до кінцевої точки сервера авторизації.

Постачальник вказує провайдера, який підключається до проміжного Owin і обробляє події при отримані запитів. На рисунку 2.2 наведено як працює авторизація.

Оскільки було додано декілька нових парамертрів, варто описати їх додавання і отримання в застосунку. Приклад як відбуваються отримання токену, додавання до нього додаткових значень наведено в лістингу 2.13.

Лістинг 2.13 – Код додавання до токену необхідних значень

```
                oAuthIdentity.AddClaim(new Claim(
                    ClaimTypes.Name, user.UserName)
                    );
                oAuthIdentity.AddClaim(new Claim(
                    ClaimTypes.NameIdentifier, user.Id)
                    );
                oAuthIdentity.AddClaim(new Claim(
                    ClaimTypes.Sid, account.Id.ToString())
                    );
                oAuthIdentity.AddClaim(new Claim(
                    ClaimTypes.Role,
account.GroupType.ToString())
                    );
```

Код наведений в лістингу 2.13 додає до токену значення такі як ім’я користувача, ідентифікатор користувача, роль користувача. Для цього необхідно дописати цей код в GrantResourceOwnerCredentials який і генерує токен. Для отримання цих даних був створений статичний клас UserIndentityWraper який видимий в усіх контролерах. Для цього наведемо

приклад, варто зауважити що перед цим йде перетворення його в те що розуміє програма, код наведений в лістингу 2.14.

Лістинг 2.14 –Отримання даних з токену

```
((ClaimsPrincipal)Thread.CurrentPrincipal)
                  .Claims
                  .Where(c => c.Type == ClaimTypes.Name)
                  .Select(c => c.Value)
                  .SingleOrDefault();
```

Як видно з коду наведеного в лістингу 2.14. Всі дані які додані містяться в Thread.CurrentPrincipal, з якого можна добути ті зміні які описані в лістингу 2.13, для кожного з цих написаний схожий код.

2.2.4 Контролери в Web API

В даному проекті використовується 8 контролерів, які використовують схожі методи, і імена для прикладу розглянемо один з контролерів, найпростіший, розглянумо кожен метод окремо.

Перший метод служить для отримання даних код якого наведений в лістингу 2.15.

Лістинг 2.15 – Код для отримання даних

```
[HttpGet]
     [Route("course/{course}/comment/{id}")]
     public Comment Get(long id)
     {
         return CommentManager.Get(id);
     }
```

Даний метод використовує значення ід отриманого з URL, за допомогою прив’язки параметрів, для цього потрібно передати в функцію відповідну назву параметра використовуючи те саме ім’я змінної як і в URL. Метод повертає значення коментаря

Наступний метод це внесення даних до бази даних дані надходять в форматі JSON. Код метод наведено в лістингу 2.16.

Лістинг 2.16 – Код методу отримання даних

```
‘
        [HttpPost]
            [Route("course/{courseId}/comment")]
            public HttpResponseMessage Post(Comment entity, long
courseId)
            {
                if
(ApplicationsCourseManager.IsUserHadRight(UserIndentityWraper.Ac
countId, courseId))
                {
                    entity.CourseId = courseId;
                    entity.OwnerId                                =
UserIndentityWraper.AccountId;

                    CommentManager.Save(entity);

                    return
Request.CreateResponse(HttpStatusCode.Created, entity);
                }

                return
Request.CreateResponse(HttpStatusCode.Forbidden, "User must had
course or be owner");

            }
```

Даний метод демонструє отримання даних, для завдання отримання даних потрібно вказати як параметр об’єкт (його клас), який очікується, перед цим йде перевірка чи користувач може додати коментар до курсу за допомогою IsUserHadRight. Якщо все добре то йде внесення в базу даних нового запису, інакше відправляється 403 (Forbidden).

Наступний метод є складніший оскільки необхідна перевірка, чи такий запис існує, і користувач має право його редагувати власне це метод оновлення даних код наведений в лістингу 2.17.

Лістинг 2.17 – Методу оновлення даних

```
         [HttpPut]
            [Route("course/{courseId}/comment/{id}")]
            public  HttpResponseMessage  Put(long  id,  Comment
entity, long courseId)
            {
                if      (CommentManager.Get(id).OwnerId      ==
UserIndentityWraper.AccountId)
                {
                    CommentManager.Update(entity);

                    return
Request.CreateResponse(HttpStatusCode.OK);
```

```
‘
                }

                return
Request.CreateResponse(HttpStatusCode.Forbidden,  "User  had  no
permission");

            }
```

Даний метод, наведений в лістингу 2.17, спочатку необхідно перевірити чи коментар, належить цьому користувачу за допомогою класу UserIndentityWraper і властивості accountId яка повертає ід поточного користувача. Якщо все добре то відбувається оновлення запису в базі даних, інакше відправляється 403 (Forbidden).

Наступний метод видаляє запис з бази даних код якого наведений в лістингу 2.18.

Лістинг 2.18 –Методу видалення даних

```
[HttpDelete]
        [Route("course/{courseId}/comment/{id}")]
        public HttpResponseMessage Delete(long id)
        {
            if      (CommentManager.Get(id).OwnerId      ==
UserIndentityWraper.AccountId)
            {
                CommentManager.Delete(id);

                return
Request.CreateResponse(HttpStatusCode.OK);
            }

            return
Request.CreateResponse(HttpStatusCode.Forbidden,  "User  had  no
permission");}
```

Даний метод, наведений в лістингу 2.18, спочатку необхідно перевірити чи коментар, належить цьому користувачу за допомогою класу UserIndentityWraper і властивості accountId яка повертає ід поточного користувача. Якщо все добре то відбувається видалення запису з бази даних, інакше відправляється 403 (Forbidden).

3 РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

3.1 Технології застосовані для розробки програми

3.1.1 Вибір технологій програмування

Для відображення матеріалів у проекті використовується об’єктно-орієнтовна мова C#.

Об’єктно-орієнтовна мова - це комп'ютерна мова програмування, яка формується навколо поняття об’єкта. Об'єктно-орієнтовані мови були розроблені для полегшення розробки, налагодження, повторного використання та обслуговування програмного забезпечення. Розуміння об'єктів та об'єктно-орієнтованих мов вимагає знання еволюції мов комп'ютерного програмування та структур даних..

C# — об'єктно-орієнтована мова програмування від компанії Microsoft, що має на меті поєднати обчислювальну потужність C++ з простотою програмування Visual Basic та кросплатформовість. C# заснований на C ++ і містить функції, подібні до функцій Java. C# призначений для роботи з Microsoft .Net.

.Net — це набір інструментів, що складаються з компонентів часу виконання, бібліотеки та компілятора, а також надає можливість створювати додатки, які працюють на Windows, Mac OS X та Linux.

MySQL - це найпопулярніша база даних із відкритим кодом у світі. Завдяки своїй перевіреній продуктивності, надійності та простоті використання MySQL стало провідним вибором баз даних для веб-додатків, включаючи Facebook, Twitter, YouTube, Yahoo! і багато іншого. Oracle стимулює інновації MySQL, надаючи нові можливості для роботи в Інтернеті, хмарі, мобільних і вбудованих додатках нового покоління.

3.1.2 Опис архітектури програмної системи

Програма складається з декількох частин:

- Взаємодія з базою даних та основна логіка програми
- Графічний інтерфейс користувача

У першу чергу розпочнемо логічний опис основної частини, яка забезпечує взаємодію з базою даних та основний функціонал системи.

Система класів складається з шести класів (рис 3.1): MyProduct, Database_manag, User, Sold_product, Number_product, Defective_product.

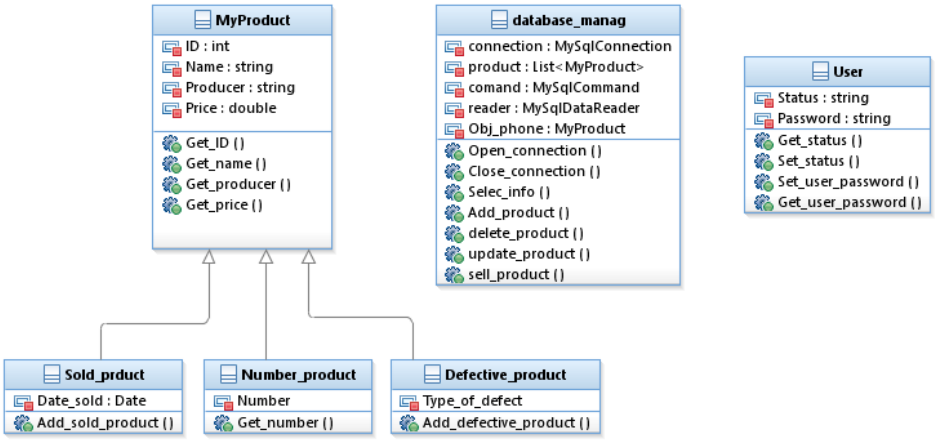


Рисунок 3.1 – Діаграма класів відповідальних за роботу системи

MyProduct – клас, який відповідає за збереження інформації про кожен товар(id товару, назви товару, виробника товару і ціни товару).

Клас database_manag відповідає за з'єднання з базою даних і маніпуляцією інформації , яку отримує з цієї бази(отримання даних, редагування даних, додавання нових даних, видалення даних).

Клас User зберігає дані про користувача(статус, пароль) і дозволяє отримати пароль з бази даних, який відповідає продавцеві або адміністраторові.

Sold_product – клас, який відповідає за формування списку проданих товарів, кожен проданий товар має відповідний час проажу і унаслідується від класу MyProduct.

Number_product – клас, який відповідає за збереження кількості кожного товару і унаслідується від класу MyProduct.

Sold_product – клас, який відповідає за формування списку бракованих товарів і унаслідується від класу MyProduct.

3.1.3 Опис концептуальної моделі бази даних

Для пришвидшення виконання певних елементів програми, було прийнято рішення помістити частину логіки на сторону сервера. На сервері було розміщено п'ять вбудованих процедур та два тригера.

Вбудовані процедури мають мету пришвидшити та полегшити виконання певних алгоритмів пов’язаних з маніпуляціями над даними з бази даних, тому вони виконують операції, які користувач програми, в теорії, буде використовувати досить часто.

Тригери мають мету забезпечити цілісність даних, які записуються у базу даних та, при необхідності, певним чином редагують записи при додаванні їх у базу даних.

Вбудована процедура select_all_File – забезпечує отримання усієї інформації про товар.

Вбудована процедура Select_sell_product – забезпечує отримання списку усіх проданих товарів.

Вбудована процедура insert_product – забезпечує додавання товару до бази даних.

Вбудована процедура delete_product – забезпечує видалення товару з бази даних.

Вбудована процедура Update_product – забезпечує редагування товару в базі даних.

Тригер price_add – тригер у таблиці product, який викликається перед спробою додавання даних в таблицю. Контролює коректність ціни, що додається, при спробі додавання ціну, яка менша 0, тригер змінює його ціну на 0;

Тригер Number_add – тригер у таблиці number_product, який викликається перед спробою додавання даних в таблицю. Контролює коректність кількості товару, що додається, при спробі додавання товару, кількість якого менша 0.

3.2 Інтерфейс програми

Для тестування буде використовуватись уже розгорнута на комп'ютері система, тестувалася на коректне відображення компонентів та правильну роботу ключових елементів.

Виконавши вхід у систему під іменем адміністратора ми бачимо його основний функціонал (рис. 31).

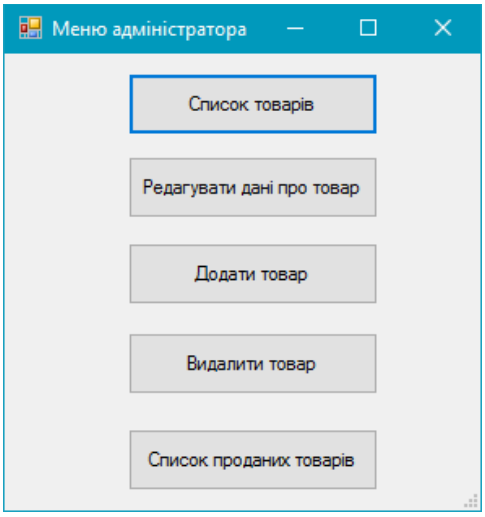


Рисунок 3.1 – вікно з функціоналом адміністратора

Нажавши на клавішу “Список товарів”, ми переходимо до вікна, де відображається таблиця наявних товарів, якщо змінити “Наявний товар” на “Товар, що закінчився” то відкриється список товару, що закінчився (рис. 3.2).

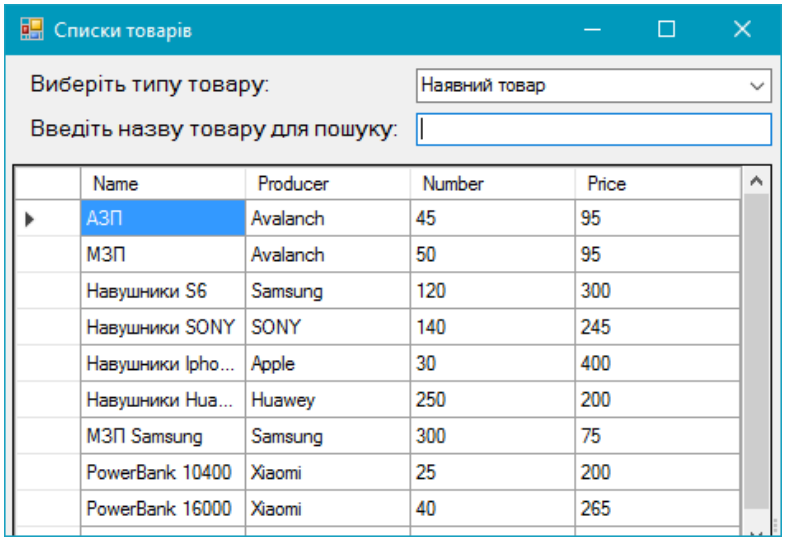


Рисунок 3.2 – Вікно списку товарів

Закривши вікно “Список товарів”, ми можемо перейти до вікна “Додати новий товар”, де надається можливість додати новий товар до нашої бази, якщо такого товару немає в списку, то система додасть новий, якщо він вже наявний, то програма це врахує і додасть тільки нову кількість і ціну товару (рис. 3.3).

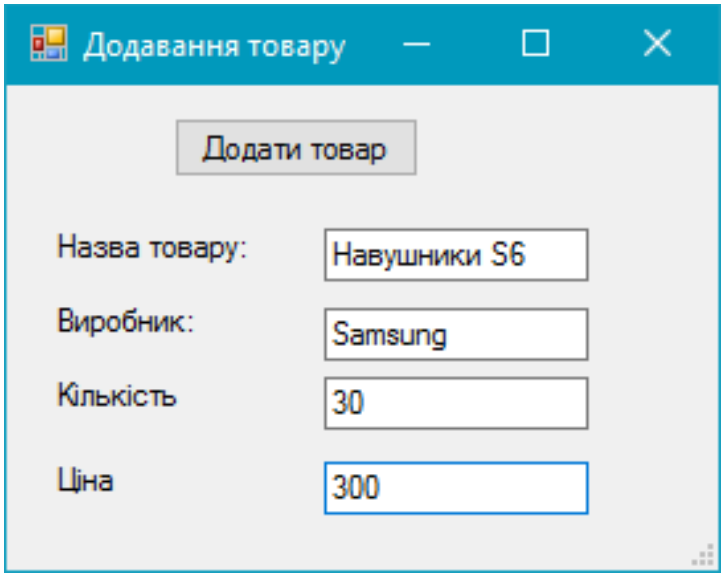


Рисунок 3.3 – Вікно додавання нового товару

Повернувшись до попереднього меню, ми можемо відкрити вкладку “Редагувати дані про товар”. Після відкриття ми можемо обрати потрібний нам товар у списку й нажати пошук, після чого програма заповнить відповідні строки з поточними даними про товар(назва, виробник, кількість і ціну), після чого їх можна буде змінити на нові (рис.3.4).

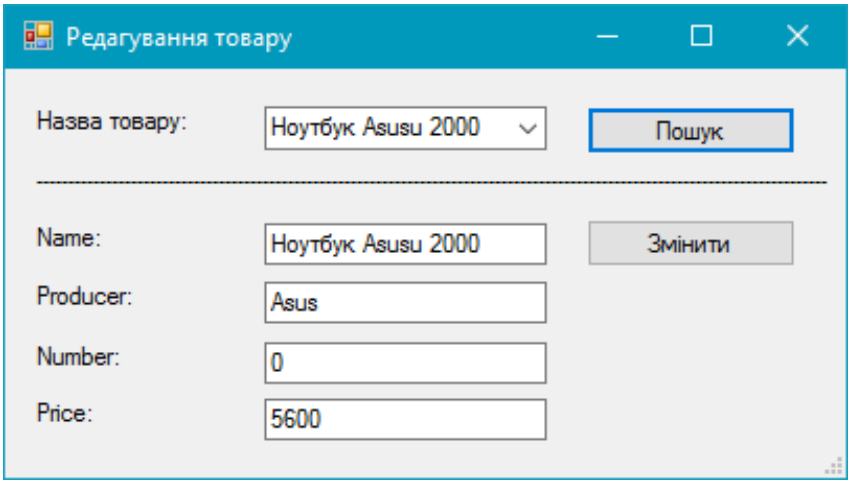


Рисунок 3.4 – Вікно редагування товару

Повернувшись до функціоналу адміністратора, ми можемо нажати клавішу видалити товар. Після відкриття вікна потрібно обрати товар зі списку і нажати клавішу “Видалити”, після чого товар буде видалений (рис. 3.5).

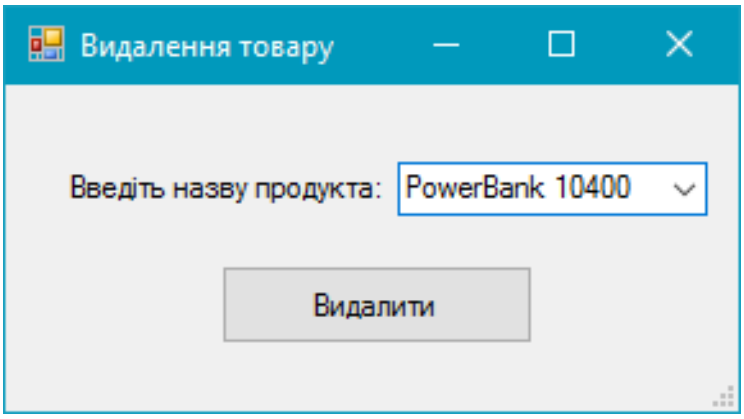


Рисунок 3.5 – Вікно видалення товару

Нажавши кнопку “Список проданих товарів”, ми переходимо до вікна, де відображаються товари за різний період часу(день, тиждень, місяць) (рис.3.6).

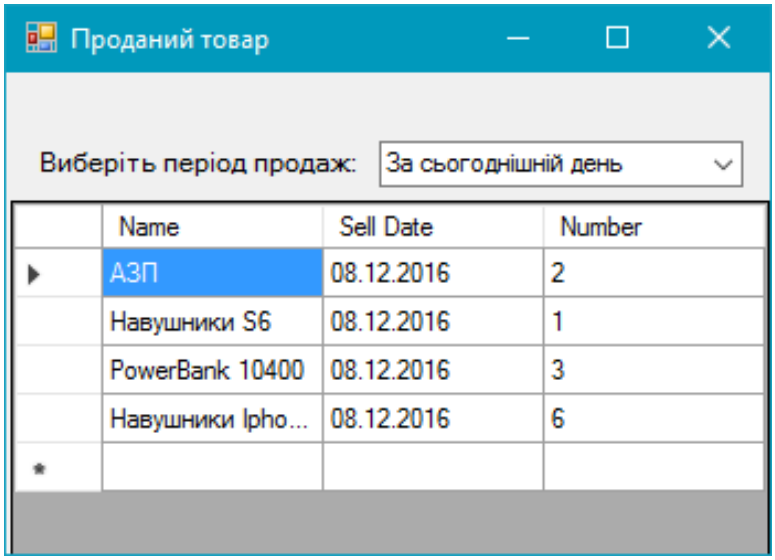


Рисунок 3.6 – Вікно списку проданих товарів

Виконавши вхід у систему під іменем продавця ми бачимо його основний функціонал, кнопка “Список наявних товарів” відкриє аналогічне вікно списку усіх товарів, як і в адміністратора, дивитися вище (рис. 3.1).

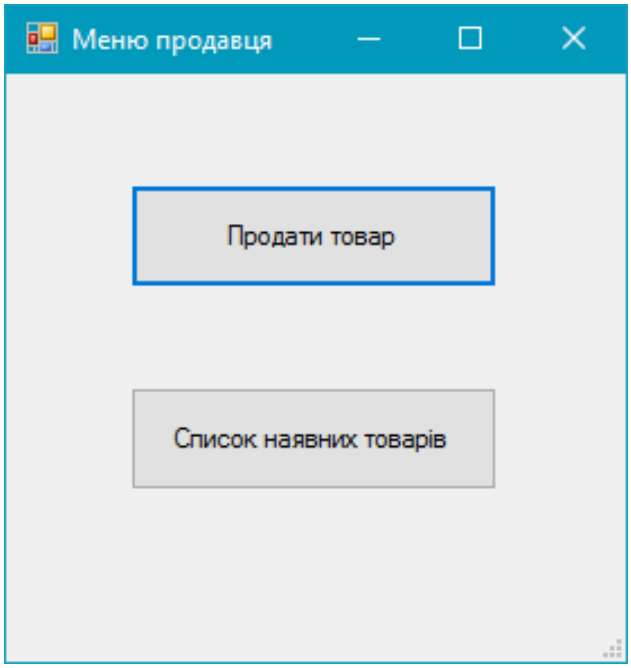


Рисунок 3.7 – вікно функціоналу продавця

Натиснувши на кнопку “Продати товар” ми перейдемо до вікна, у якому продавцеві надається можливість продавати товар. Заповнивши відповідну форму з назвою товару і його кількоскті, потрібно нажати клавішу “Продати” і система автоматично порахує загальну суму до оплати за вказаний товар (рис. 3.8).

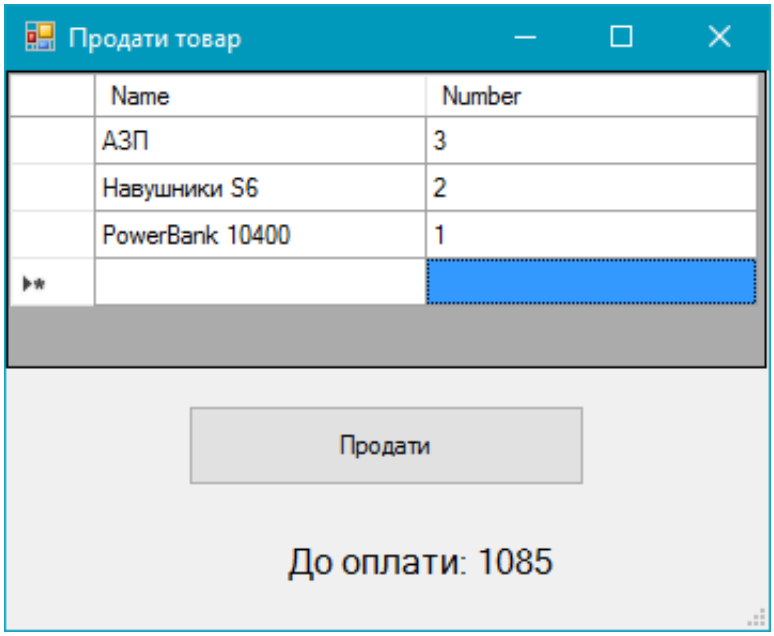


Рисунок 3.8 – вікно продажу товару

4 ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

4.1 Розрахунок норм часу на виконання науково-дослідної роботи

Метою даного розділу є встановлення економічної доцільності розробки програмного забезпечення із використанням технології .NET

Оцінка вартості комп'ютерних програм базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням амортизації. При цьому для оцінки нематеріальних активів використовують міжнародні стандарти оцінки розрахунку вартості об'єктів інтелектуальної власності, розроблені TIAVIS (The International Assets Vabulation Standarts Commitie). Витрати на реалізацію проекту – це витрати на розробку та реалізацію, що в перспективі будуть покриті доходом від реалізації споживачам. При поширенні проекту розмір доходів буде значно більшим за витрати.

Враховуючи залежність якості кінцевого продукту від кваліфікації програмістів, розробники часто йдуть на додаткові заходи їх стимулювання,

які найчастіше втілюються у певній системі преміювання, акційні відрахування за кожну продану одиницю програми.

Разом з тим до створення ПЗ можуть бути залучені також позаштатні програмісти як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору, найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все залежить від того, чи зареєстрований виконавець підприємцем. Також необхідно врахувати витрати на оподаткування відповідно.

Важливим етапом розробки ПЗ є його тестування, яке виконується тестувальником за певну винагороду. Якщо тестувальники не перебувають з підприємством у трудових відносинах, оплата виконується на основі договору підряду на виконання робіт з тестування ПЗ.

Слід зауважити, що з нового 2013 року встановлено тимчасове, з 01.01.2013 р. до 01.01.2023 р., звільнення від оподаткування ПДВ операцій з постачання програмної продукції [18]. Податок на прибуток встановлено у розмірі 5%. Однак слід зауважити, що невідповідність закону «авторське право і суміжні права» положенням податкового кодексу може значно ускладнити процедуру визнання підприємця як суб’єкта сектора програмного забезпечення.

Вважатимемо, що усі програмісти працюють у штаті підприємства-розробника і їм встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість (днів) і основна заробітна плата кожного учасника техпроцесу представлено у таблиці 4.1.

Таблиця 4.1 – Розрахункова вартість технологічного процесу

Посада	Місячний оклад	Денна зар. плата	Об’єктно-орієнтований підхід		Процедурний підхід	
			Днів	Сума	Днів	Сума

Керівник	10800,00	480,00	7	3360,00	7	3360,00
Програміст	7140,00	340,00	23	7820,00	27	9180,00
Тестер	3570,00	340,00	2	680,00	2	680,00
Дизайнер	7060,00	360,00	8	3040,00	8	3040,00
Додаткова зар. плата 20%			25	2980,00	29	3252,00
Фонд оплати праці 36,77%				6574,48		7174,57
Всього витрат на зар. плату				24454,48		26686,57
Військовий збір 1,5%				366,82		400,30
Єдиний соціальний внесок 3.6%				643,68		702,43
Доходи фізичних осіб 15%				2585,45		2821,44
Всього				28050,43		30610,74

У таблиці 4.1 всі суми наведені в гривнях.

З цієї суми утримуються обов’язкові відрахування на заробітну плату: єдиний соціальний внесок, який складає 3,6% від суми нарахованої заробітної плати [18] та податок на доходи фізичних осіб, який складає 15% від суми нарахованої заробітної плати, зменшеної на суму єдиного внеску на загальнообов’язкове соціальне страхування та податкової соціальної пільги [18]. Нарахування на фонд оплати праці. Зокрема, видання програмного забезпечення – 36,77%.Також 1,5% військовий збір. Розрахунки в таблиці 4.1.

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни.

Таблиця 4.2 – Матеріальні витрати

Найменування ресурсу	Кількість	Ціна одиниці	Загальна сума
Флешки	2	150,00 грн.	300,00 грн.
Папір для записів	100	0,15 грн.	15,00 грн.
Картридж для принтера	1	50,00 грн.	50,00 грн.
Всього			365,00 грн

Отже, загальна сума матеріальних витрат становить 365 гривень.

Розрахунок витрат на електроенергію. Витрати на електроенергію одиниці обладнання визначаються за формулою:

$$З_в = WTS \quad (4.1)$$

де W – необхідна потужність, кВт;

T – кількість годин роботи обладнання;

S – вартість кіловат-години електроенергії, S = 2.50 грн/кВт·год.

$З_{в1} = 215$ грн. $З_{в2} = 248,82$ грн.

Розрахунок суми амортизаційних відрахувань. Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 %, вартість яких перевищує 1000 грн. і визначається:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}} \quad (4.2)$$

де $C_Б$ – балансова вартість обладнання, грн;

N_A – норма амортизаційних відрахувань в рік, %;

$T_{ГОД}$ – річний робочий фонд часу, год;

$T_{ФАК}$ – фактичний час роботи обладнання по написанню програми, год.

Таблиця 4.3 – Перелік необхідного обладнання

Найменування	Кількість	Ціна	Сума
Комп'ютер	2	8000,00 грн.	16000,00 грн.
Принтер	1	1200,00 грн.	1200,00 грн.
Середовища розробки	-	4000,00 грн.	4000,00 грн.
База даних	1	2000,00 грн.	2000,00 грн.
Хостинг (1 рік оренда)	2	450,00 грн	900,00 грн.
Операційна система	2	Безкоштовно	безкоштовно
Всього більше 1000 грн			23200,00 грн.

Всього витрат на амортизацію	1380,96	1601,91
Всього	25480,96	25701,91

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

Залежно від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників. Нехай вона буде дорівнювати 45%, яка 6705 грн для об’єктно-орієнтованого і процедурного 7317 грн..

Проведемо розрахунок вартості створюваного програмного продукту. Вартість продукції включає у собі собівартість і планований прибуток.

Собівартість продукції – це сума грошових витрат підприємства (фірми) на виробництво і збут одиниці продукції, виконання робіт та надання послуг.

Повна собівартість програмного продукту дорівнює сумі усіх витрат на його виробництво: 33114,55 грн. об’єктно-орієнтований, 36213,30 грн процедурний загальна собівартість

Приймемо прибуток на рівні 30% (9934,37 грн., 10863,99 грн.). Для нових інноваційних продуктів, що користуються високим попитом на ринку, ринкову вартість B_{p1} =43048,88 грн. для об’єктно-орієнтованого, B_{p2}= 47077,29 грн. для процедурного.

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (Е) полягає у відношенні результату виробництва до затрачених ресурсів.

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку.

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($Т_{ок}$):

$$T_{ок} = \frac{1}{E} \quad (4.4)$$

У нашому випадку $Т_{ок1}$= $Т_{ок2}$ =1/0,30=3,33 років, що є нормальним, оскільки допустимим вважається термін окупності до 5 років.

Даний розрахунок виконаний у розрахунку на 1 екземпляр програмного продукту без врахування його тиражування.

Виходячи із експертних оцінок і складності програми, приймемо величину витрат на супровід і модернізацію програмного забезпечення, створеного за процедурним методом 40%(14485,32 грн.) від початкових витрат, а за об'єктно-орієнтованим – 20%(6623 грн.).

Таблиця 4.4 – Загальні витрати

	Процедурний	Об'єктно-орієнтований
Зарплата основна, грн	16260,00	14900
Зарплата додаткова, грн.	3252,00	2980
Фонд заробітної плати, грн.	19512,00	17880
Відрахування на ФОП, грн.	7174,57	6574.48
Разом на оплату праці, грн.	26686,57	24454,55
Матеріальні витрати, грн.	359,00	359,00
Електроенергія, грн.	248,82	215,00
Амортизація, грн.	1601,91	1380,96
Накладні витрати, грн.	7317,00	6705,00
Разом на ін.витрати, грн.	9526,73	8659,96
Собівартість	36213,30	33114,51
Прибуток	10863,99	9934,37
Вартість розробленого ПЗ	47077,29	43048,88

Економічна ефективність	0,30	0,30
Термін окупності, років	3,33	3,33
Модернізація і супровід	14485,32	6623
Загальні витрати на придбання	61562.61	49671,88
Економія витрат для споживача	-	-11890,73
Дохідність проекту для споживача	59100,11	48585,97
Економія	-	10518,14

Загальна вартість пропонованих робіт становить 61562,61 грн для процедурного і 49671,88 грн. для об’єктно-орієнтованого. Оскільки ефективність для обох типів проекту, становить 0,3, вкладені кошти окупляться за 3,33 року. Варто вибрати об’єктно орієнтований підхід через меншу затрату коштів.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦ

5.1 ОХОРОНА ПРАЦІ

Працівники установ, що використовують програмну систему з продажу електроніки повинні дотримуватися норм внутрішнього трудового розпорядку, виконувати правила особистої гігієни та гігієни приміщення, володіти знаннями з техніки безпеки користування комп’ютерною технікою, пожежної безпеки. Для забезпечення норм охорони праці та безпеки використання програмної системи користувачі повинні проходити первинний інструктаж з охорони праці на робочому місці.

Роботодавець повинен проінформувати працівників під розписку про умови праці та наявність на їхніх робочих місцях небезпечних і шкідливих виробничих чинників (фізичних, хімічних, біологічних, психофізіологічних), які виникають під час роботи з екранними пристроями та ще не усунуто, а також про можливі наслідки їх впливу на здоров'я працівників.

Також роботодавець повинен забезпечити навчання й перевірку знань працівників із питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними, а також у випадках модифікації й організації роботи обладнання.

Для внутрішнього оздоблення приміщень з персональними компютерами, де використовують програмне забезпечення з продажу електроніки слід обирати світлі нейтральні кольори стін. Покриття підлоги та поверхня має бути рівною, неслизькою, з антистатичними властивостями. Ці умови оздоблення необхідно враховувати, адже система вимагає забезпечення максимально комфортного середовища. Це сприятиме максимальній ефективності отримання результатів.

Згідно ГОСТ 12.1.005-88 “Повітря робочої зони. Загальні санітарно-гігієнічні вимоги”, ДСН 3.3.6.042-99 температура навколишнього середовища повинна бути в межах +18°С - +22°С, відносна вологість повітря близько 55% швидкість руху повітря – 0,1-0,2 м/сек. Рівні позитивних і негативних іонів у повітрі мають відповідати санітарно-гігієнічним нормам №2152-80. Допустима інтенсивність шуму на робочих місцях з ЕОМ має відповідати вимогам ДСанПіН 3.3.2-007-98: оптимальна — до 45 дБ, гранична — до 60 дБ. Надмірний шумовий вплив та вібрації негативно впливають на якість виконання та викликають дискомфорт для користувачів.

Потрібно створити сприятливі умови для зорової роботи, які б мінімізували втому очей, виникнення професійних захворювань та сприяли підвищенню продуктивності праці. Тому освітлення повинне відповідати вимогам ДБН В.2.5-28:2018 «Природне і штучне освітлення». Основною вимогою є необхідність створення на робочій поверхні освітленості, що відповідає характеру зорової роботи і знаходиться в межах встановлених норм. Освітлення у приміщенні має бути суміщеним. Відтак, недостача денного природнього освітлення компенсується необхідною для приміщення кількістю штучного освітлення. Як джерело штучного освітлення в приміщеннях, де встановлено комп’ютерну техніку, бажано використовувати

люмінесцентні лампи. Освітленість робочого місця у горизонтальній площині на висоті 0,8 м від рівня підлоги повинна бути не менш 400 лк. Для захисту від прямих сонячних променів повинні бути передбачені сонцезахисні пристрої, жалюзі, штори. Безпосередньо при виконанні роботи з обладнанням можливе використання місцевого освітлення в комбінації з загальним.

Одним із важливих параметрів охорони праці в ході експлуатації ЕОМ є ергономіка користування. Розробляючи програму, важливо врахувати фізіологічні властивості користувачів. Важливо ретельно підійти до питання вибору розмірів іконок та рисунків, кольорової гами. Так, для кращого сприймання та розрізнення поданої інформації на дисплеї, рекомендовано виконати текст в темних тонах на світлому фоні. Таким чином збільшується акцент на поданій інформації.

При розробці програмного продукту з продажу електроніки та в ході його експлуатації користуються лініями електромереж. Для побудови системи потрібно використовувати лише якісні та сертифіковані пристрої та засоби. Персональні комп'ютери і периферійні пристрої повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. В них, окрім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Конструкція вилки має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Усі електроприлади, згідно з ДНАОП 0.00-1.21-98, повинні бути заземленні за допомогою нульового захисного провідника.

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу.

Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією.

Приміщення, де працюють користувачі з готовою програмною ситемою з продажу електроніки мають бути оснащені системою автоматичної пожежної сигналізації і вогнегасниками відповідно до вимог чинного законодавства України.

5.2 ПІДВИЩЕННЯ СТІЙКОСТІ РОБОТИ ОБ'ЄКТІВ ГОСПОДАРСЬКОЇ ДІЯЛЬНОСТІ У ВОЄННИЙ ЧАС

На основі вивчення факторів, які впливають на стійкість роботи об'єктів, і оцінки стійкості елементів і галузей виробництва проти уражаючих факторів ядерної, хімічної і біологічної зброї, стихійних лих і виробничих аварій, необхідно завчасно організувати і провести організаційні, інженерно-технічні й технологічні заходи для підвищення стійкості роботи.

Здійснення організаційних заходів передбачає завчасну підготовку всіх структур цивільного захисту, служб і формувань до надзвичайних ситуацій.

Вжиттям технологічних заходів підвищується стійкість роботи об'єктів шляхом змінювання технологічних процесів, режимів, можливих в умовах надзвичайних ситуацій.

Інженерно-технічні заходи мають забезпечити підвищену стійкість виробничих споруд, технологічних ліній, устаткування, комунікацій об'єкта до впливу вражаючих факторів під час надзвичайних ситуацій.

При проведенні цих заходів необхідно враховувати конкретні умови об'єкта народного господарства. Проте є загальні організаційні інженерно-технічні заходи, які мають проводитись на всіх об'єктах.

1. Забезпечення захисту людей та їх життєдіяльності. Створення на об'єкті надійної системи оповіщення про загрозу нападу противника, радіоактивне забруднення, хімічне і біологічне зараження, загрозу стихійного

лиха і виробничої аварії. Організація розвідки і спостереження за радіоактивним забрудненням, хімічним і біологічним зараженням; гідрометеорологічне спостереження за рівнем води, напрямком і швидкістю вітру, рухом і поширенням хмари радіоактивного забруднення, СДЯР і ОР.

Створення фонду захисних споруд ЦО, запасів засобів індивідуального захисту і забезпечення своєчасної видачі їх населенню.

Завчасна підготовка до масової санітарної обробки населення і знезаражування одягу, організація взаємодії з установами охорони здоров'я для медичного обслуговування населення у надзвичайних ситуаціях.

Підготовка до евакуації населення, розміщеного в зонах можливих руйнувань і катастрофічного затоплення. Завчасна підготовка місць евакуації, організація прийому евакуйованого населення на територію населених пунктів.

Постачання населення продуктами харчування, питною водою, предметами першої необхідності; комунальне побутове обслуговування населення з урахуванням проведення евакуаційних заходів, забезпечення захисту продовольчих запасів.

Навчання населення способам захисту, надання першої допомоги, практичним діям в умовах надзвичайних ситуацій, морально-психологічна підготовка населення для виживання.

Забезпечення чіткої інформації про обстановку та правила дій і поведінки населення в надзвичайних ситуаціях мирного і воєнного часу.

2. Захист цінного й унікального устаткування. Захистити цінне і унікальне устаткування можна завдяки проведенню інженерно-технічних заходів, щоб зменшити небезпеку пошкодження і руйнування цінного й унікального устаткування, станків з програмним керуванням, шліфувальних, токарних, розточувальних, зубофрезерних, пресових станків, автоматичних конвеєрних ліній та іншого устаткування.

Варіантами такого захисту є розміщення зазначеного устаткування в заглиблених приміщеннях а також використання спеціальних захисних

пристосувань, закріплення станків на фундаментах, застосування контрфорсів для підвищення стійкості проти перекидання обладнання.

3. Стійкість роботи галузі рослинництва. Планування і проведення заходів захисту сільськогосподарських рослин, урожаю в різних надзвичайних ситуаціях.

Встановлення надійної взаємодії зі станцією захисту рослин, радіологічною і агрохімічною лабораторією для організації спостереження за зараженістю посівів сільськогосподарських культур та ґрунтів, відбір необхідних проб та їх аналіз.

Впровадження у виробництво високоурожайних, стійких проти небезпечних хвороб і шкідників сільськогосподарських культур.

Підготовка техніки і хімічних засобів захисту сільськогосподарських культур від біологічних засобів ураження.

Розробка заходів збирання урожаю в умовах обмеженості забезпечення людьми, технікою, паливом і мастилами, порушення міжгалузевих зв'язків, технології доведення урожаю до кондиції.

Організація зберігання і переробки урожаю в господарстві при порушенні зв'язків із заготівельними й переробними організаціями та підприємствами.

Розробка і підготовка до впровадження спрощених технологій вирощування сільськогосподарських культур, підготовка до зміни сівозмін і перепрофілювання рослинництва.

Забезпечення ефективного використання сільськогосподарських угідь в умовах радіоактивного забруднення, зараження хімічними і біологічними засобами.

Підготовка всіх засобів для захисту працюючих у рослинництві в різних умовах надзвичайних ситуацій.

4. Стійкість роботи тваринництва. Підготовка до проведення ветеринарно-санітарних заходів, спрямованих на зниження втрат тварин від сучасних засобів ураження. Завчасна підготовка приміщень для утримання

тварин. Розробка заходів захисту тварин на пасовищах. Створення запасів кормів і організація забезпечення водою.

Організація ветеринарної розвідки в господарстві, відбір необхідних проб та їх аналіз.

Створення індивідуальних засобів захисту для елітного поголів'я худоби.

Розробка заходів евакуації тварин із зон можливих руйнувань, катастрофічного затоплення, районів хімічного зараження, підготовка місць для евакуації тварин. Планування заходів захисту кормів, джерел водопостачання і тваринницьких ферм.

Організація забезпечення основних виробничих процесів у тваринництві електроенергією від автономних джерел електропостачання, у разі відключення від центральної енергомережі.

Підготовка до постійної готовності спеціальної техніки для обробки тварин, а також пристосування для цієї мети іншої техніки, наявної в господарстві.

Організація ветеринарної обробки, утилізації і забою уражених тварин, тимчасового зберігання м'ясної продукції при порушенні господарських зв'язків із заготівельними організаціями і підприємствами.

Розробка найпростіших технологій переробки і зберігання продукції тваринництва в разі неможливості відправки переробним підприємствам і реалізації.

Організація забезпечення працюючих у тваринництві колективними та індивідуальними засобами захисту.

5. Підвищення стійкості мереж комунального господарства. Для забезпечення стійкості роботи об'єктів повинні проводитись інженерно-технічні заходи на мережах комунального господарства з метою захисту джерел тепла із заглибленням у ґрунт комунікацій. Котельні слід розміщувати в спеціальному окремо розміщеному приміщенні.

Якщо об'єкт одержує тепло з міської теплоцентралі, необхідно провести заходи для забезпечення стійкості трубопроводів і розподільних пристроїв, підведених до об'єкта.

Теплова мережа має будуватися за кільцевою системою з прокладанням труб у спеціальних каналах зі з'єднанням паралельних ділянок. Ці пристосування необхідно розміщувати в оглядових колодязях, на території, що не завалюється при руйнуванні будівель.

Система каналізації має будуватись окремо: одна для дощових, друга для промислових і господарських вод. На об'єкті має бути не менше двох виводів з підключенням до міських каналізаційних колекторів, а також виводи і колодязі з аварійними засувками на об'єк-тових колекторах з інтервалом 50 м на території, що не завалюється, для аварійного скидання неочищеної води в найближчі штучні та природні заглиблення.

На деяких промислових об'єктах е системи для забезпечення технології виробництва: для подання кисню, аміаку, стиснутого повітря та інших рідких і газових реактивів. Для цих систем розробляють заходи для попередження виникнення вторинних факторів зброї, стихійних лих та виробничих аварій і катастроф.

6. Забезпечення стійкості роботи паливно-енергетичного комплексу і водопостачання. Створення резерву енергетичних потужностей за рахунок автономних пересувних електростанцій, а також місцевих джерел електроенергії. Підготовка автономних електростанцій до роботи за спеціальним режимом (графіком) для забезпечення технологічних процесів виробництва, для яких неможливі тривалі перерви в електропостачанні.

З метою попередження аварій на електричних мережах необхідно установити автоматичну систему відключення при виникненні перенапруги. Повітряні лінії* електропостачання замінити на підземно-кабельні.

Створення необхідних запасів (резервів) паливно-мастильних матеріалів та інших видів палива й організація їх безпечного зберігання.

Щоб не допустити зупинки підприємства через дефіцит палива, необхідно підготуватись для роботи на різних видах палива: нафта, вугілля, газ. Для підвищення стійкості забезпечення водою слід провести такі заходи. Необхідно створити основні і резервні джерела водопостачання. Як резервне джерело краще мати артезіанську свердловину, яку необхідно підключити до системи водопостачання.

ВИСНОВКИ

Розроблено модель даних, яка повністю задовольняє мету розробки ПЗ, описано основні класи для роботи з даними, які відповідають предметній області. Використані методи для створення таблиць пришвидшують і спрощують розробку з обліком та продажом електроніки. Наведені основні аргументи вибору бази даних, середовища та моделі розробки, які суттєво впивають на процес весь процес реалізації проекту. Описано переваги і недоліки використаних методів, можливі ризики в майбутньому, що дозволить їх попередити та мінімізувати.

Описано основні принципи роботи класів, для отримання даних з бази даних з використанням технології Entity Framework, описано основні етапи створення WEB API, зроблено обґрунтування вибору технологій створення API. Створене API повністю задовольняє вибраній темі роботи, і виконує поставлені функції розробки.

Наведено основні принципи розробки програмного забезпечення з використанням технології .NET, описано основні методи роботи для нього.

Описано як розробляється графічний інтерфейс користувача та його взаємодія з функціоналом ПЗ, всі користувацькі відповідають функціям передбачених в системі. Інтерфейс користувача розроблено з урахуванням ергономіки, яка спрощує роботу з програмою та робить її більш гнучкою інтуїтивною для сприйняття.

Виконано обґрунтування проведених досліджень та розраховано основні економічні показники, які виявилися більш економічно вигідними для об’єктно-орієнтованого підходу, ніж для процедурного, що є абсолютно очікуваним.

Враховані вимоги ергономіки до організації робочого місця оператора ПК, описано основні методи для розміщення елементів на столі. Розробка технічних засобів щодо захисту горючих речовин від статичної електрики дозволило звести імовірність пожежі до мінімуму.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : http://uk.wikipedia.org.
2. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : http://www.omega.ru/oracleinfo.html
3. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : http://uk.wikipedia.org/wiki/Microsoft_SQL_Server.
4. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : https://msdn.microsoft.com
5. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : https://en.wikipedia.org/wiki/Language_Integrated_Query.

6. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : https://en.wikipedia.org/wiki/Application_programming_interface.

7. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : https://uk.wikipedia.org/wiki/HTTP.

8. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : http://www.asp.net/web-api/overview/security/individual-accounts-in-web-api.

9. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : https://angularjs.org.

10. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : http://www.asp.net/web-api/documentation/security/individual-accounts-in-web-api–

11. Посібник початківця з оптимізації для пошукової системи Google [Електронний ресурс] – Режим доступа : URL : http://static. googleusercontent.com/external_content/untrusted_dlcp/www. google.com.ua/uk/ua/intl/uk/webmasters/docs/search-engine-optimization-starter-guide-uk.pdf .

12. Розміщення елементів на веб-сторінці і навігація по сайту [Електронний ресурс] – Режим доступа : URL : http://siteua.info/8.htm.

13. Angular [Електронний ресурс] – Режим доступа : URL : https://docs.angularjs.org.

14. Grunt Tutorial [Електронний ресурс] – Режим доступа : URL : http://gruntjs.com – Загл. с экрана.

15. JavaScript Tutorial [Електронний ресурс] – Режим доступа : URL : http://www.w3schools.com/js/default.asp – Загл. с экрана.

16. Жидецький, В.Ц. Основи охорони праці. [Текст] Джигирей В.С., Мельников О.В. – Вид. 2–у, стеритипне. – Львів: Афіша, 2000. – 348с.

17. Вільсон, О.Г. Охорона праці. [Текст] – Центр дистанційної освіти КНУБА, 2002р.

18. Методичні вказівки для виконання розділу дипломної роботи щодо техніко-економічного обґрунтування вибору проектного рішення розробки та оцінки якості програмного забезпечення [Текст] / Упор. Петрик М.Р., Кінах Я.І., Головатий А.І., Рогатинська Л.Р. – Тернопіль: Вид-во ТНТУ ім. І. Пулюя. – 2013. – 34 с.

додатки

ДОДАТОК А - ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем і програмної інженерії

Кафедра програмної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою програмної інженерії

“___ “_____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської роботи

на тему: «Розробка програмного забезпечення для реалізації продажу електроніки з використанням технології .net»

Заоборний Сергій Олегович

Керівник роботи:

к.т.н., доцент Михалик Д. М.

“___” _____ 2019 р.

_____

Виконавець:

студент групи СПм-61

Заоборний Сергій Олегович

“___” _____ 2019 р.

_____

м. Тернопіль – 2019

ЗМІСТ

Вступ

1. Підстави до розробки

2. Призначення до розробки

3. Вимоги до програмного продукту

3.1 Функціональні характеристики

3.2 Параметри технічних засобів

3.3 Інформаційна та програмна сполучність

4. Стадії розробки

5. Програмна документація

6. Порядок контролю та приймання

1 ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану на 2019 рік, та згідно наказу на виконання магістерської роботи студента магістра.

Тема проекту: «Розробка програмного забезпечення для реалізації продажу електроніки з використанням технології .net».

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Програма для реалізації електроніки дозволяє оптимізувати процес продажу товару, зберігати його в електронній базі даних, що зменшує кількість помилок при проведені обліку, а також створює статистичні показники за конкретно визначений період часу.

Метою магістерської роботи є розробка програмного забезпечення для реалізації продажу електроніки. Необхідно проаналізувати предметну область та дослідити програмні системи з аналогічним функціоналом.

Одним з ключових моментів є забезпечення зручності системи та інтуїтивно зрозумілий інтерфейс, а також надійності бази даних, у якій будуть зберігатися усі дані про товар та статистику його продаж.

За результатами виконаної роботи необхідно розробити програмне забезпечення, яке дозволить пришвидшити та спростити роботу персоналу магазину.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональні характеристики

Програмне забезпечення має виконувати наступні дії:

- додавати новий товар у БД;
- редагувати товар у БД;
- видаляти товар з БД;
- підтримувати можливість виходу та входу в систему, запам’ятовувати користувача в системі;
- автоматично формувати статистику проданого товару за визначений період;
- формувати звітність за певний проміжок часу;

3.2 Ппараметри технічних засобів

1) ПК з 2 ГБ оперативної пам‘яті, встановленою системою Windows 7 і вище, MacOS, Не менше 5 ГБ вільного місця на жорсткому диску. Процесор з тактовою частотою від 1.4 GHz і більше.
2) наявність встановленої бібліотеки .Net framework 3.5 та 4.5.

3.3 Інформаційна та програмна сполучність

Програмний продукт повинен коректно функціонувати в операційних системах Windows 7 та новіших, на яких доступне для встановлення бібліотека .Net framework 3.5 та 4.5. Розроблювана бібліотека класів повинна бути пристосована до використання у інформаційних системах та програмних засобах. Розробку виконувати з використанням бібліотек та технологій мови C# в середовищі програмування Visual Studio 2015 з використанням технології .Net.

4 СТАДІЇ РОЗРОБКИ

В ходів реалізації роботи проект повинен пройти крізь наступні стадії розробки:

- аналіз предметної області;
- проектування архітектури;
- реалізація архітектури;
- реалізація графічного інтерфейсу;
- тестування результатів розробки; - оформлення супровідної документації;
- здача роботи.

5 ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмного продукту повинні бути розроблені наступні документи:

- Пояснювальна записка;
- Технічне завдання;
- Презентаційний матеріал;
- Додатки.

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблений програмний продукт має виконувати всі вимоги, що складаються з перерахованих у п. 3.1 характеристик.

Приймання проводиться спеціально створеною екзаменаційною комісією в термін до:

“__” грудня 2019р.

УДК 004.415.2

С. О. Заоборний – магістрант
Д. М. Михалик – кандидат технічних наук, доцент
Тернопільський національний технічний університет ім. І. Пулюя, Україна

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ ПРОДАЖУ ЕЛЕКТРОНІКИ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ .NET

S. O. ZAOBORNY – graduate student
D.M. Mikhalik – Ph.D, Assoc. Prof.
Ternopil Ivan Pul'uj National Technical University, Ukraine

DEVELOPMENT OF SOFTWARE TO REALIZE ELECTRONICS SALES WITH THE USE OF .NET TECHNOLOGY

Ведення обліку товару в паперовому вигляді - це відлуння минулого. Сучасні фахівці використовують у своїй роботі інформаційний комплекс спеціалізованого ПЗ. Його впровадження в процес обліку та торговельної діяльності дозволяє створювати статистику в електронному вигляді і ефективно виконувати інші процеси торгівлі. Це економить час і кошти. Всі дані передаються до віртуальної бази даних для подальшого ведення обліку та статистики проданого товару[3].

Продаж товарів – це договір, за яким продавець передає або погоджується передати право власності на товар покупцеві в обмін на гроші або послуги. Якщо право власності має перейти у майбутній час, договір називається договором про продаж[3].

Система управління складом - це програмно-апаратна система управління складом, яка забезпечує комплексну автоматизацію управління складськими та логістичними процесами[3].

Авторизація - це механізм захисту для визначення рівнів доступу або привілеїв користувача / клієнта, пов'язаних із системними ресурсами, включаючи файли, послуги, комп'ютерні програми, дані та функції додатків[1].

Ідентифікація – це процес розпізнавання користувача певною системою за допомогою наперед встановленого ідентифікатора, наприклад ім'я або ID користувача[1].

Автентифікація – процедура перевірки приналежності користувачеві інформації в системі[1].

C# - об'єктно-орієнтована мова програмування від компанії Microsoft, що має на меті поєднати обчислювальну потужність C++ з простотою програмування Visual Basic та кросплатформовість[2].

.NET — це набір інструментів, що складаються з компонентів часу виконання, бібліотеки та компілятора, а також надає можливість створювати додатки, які працюють на Windows, Mac OS X та Linux[5].

Кросплатформність — властивість програмного забезпечення працювати більш ніж на одній програмній або апаратній платформі[6].

MySQL - це найпопулярніша база даних із відкритим кодом у світі. Завдяки своїй перевіреній продуктивності, надійності та простоті використання MySQL стало провідним вибором баз даних для веб-додатків, включаючи Facebook, Twitter, YouTube, Yahoo! і багато іншого[7].

Література

1.Тишин, В. И. Паскаль. Основы программирования [Електронний ресурс]. Ч. 1 : Программирование в интегрированных средах / В. И. Тишин. — Брянск, 2002. — 424 с. - Режим доступу: http://lib.sumdu.edu.ua/library/DocDownload?docid=154603

2.Алексенко, О. В. Технології програмування та створення програмних продуктів [Текст] : конспект лекцій для студ. напряму підготовки 6.050101 "Комп'ютерні науки" усіх форм навчання / О. В. Алексенко. — Суми : СумДУ, 2013. — 133 с. - URL: http://lib.sumdu.edu.ua/library/DocDownload?docid=372950

3.Академик [Електронний ресурс]: Словари и энциклопедии на Академике. - Режим доступу: http://dic.academic.ru/dic.nsf/enc_colier/3133/ЦИФРЫ

4.Планета информатики. учебник по информатике. [Електронний ресурс]: Принципы фон Неймана. - Режим доступу: http://inf1.info/machineneumann

5.ДОМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ [Електронний ресурс]: МОВИ ПРОГРАМУВАННЯ. - Режим доступу: http://www.itdom.info/Tehnol/2.htm

6.Программирование [Електронний ресурс]: Язык программирования Паскаль(Turbo Pascal). Обучающие уроки. - Режим доступу: http://life-prog.ru/view_cat.php?cat=1

7.Язык Pascal. Программирование для начинающих [Електронний ресурс]: Все материалы учебника по языку программирования Pascal (Паскаль). - Режим доступу: http://pas1.ru/pascaltextbook