**АНОТАЦІЯ**

Метою даної дипломної роботи є розробка інформаційної системи для підтримки діяльності компанії з продажу техніки.

Для досягнення цієї мети було виконано наступне:

* Досліджено процес прийому та обробки замовлень.
* Розроблено дизайн та структуру програми, алгоритм обробки замовлень.
* Розроблено бази даних, необхідні для збереження інформації про замовлення та кількість товарів на складах.
* Продукт по завершенню розробки був протестований, та оптимізований.
* Бази даних були наповнені замовленнями та товарами.

Дана програма містить:

* Зручний, динамічний та привабливий дизайн;
* Інтуїтивно зрозумілий інтерфейс;
* Достатнє інформаційне наповнення, яке забезпечить високу продуктивність роботи;
* Високу швидкість.

**ANNOTATION**

The purpose of the given thesis is developing an information system to support the company for the sale equipment.

To achieve this objective was done the following:

* Investigated the process of receiving and processing orders.
* Developed design and program structure, order processing algorithm.
* Developed a database required to store information about the orders and number of products in warehouses.
* At the end of product development process, the product has been tested and optimized.
* Databases were filled with orders and products.

This program includes:

* Easy, dynamic and attractive design;
* Intuitive interface;
* Adequate content, which ensures high productivity;
* High performance;

Зміст

[ВСТУП 4](#_Toc483855083)

[1. ОПИСОВА ЧАСТИНА 6](#_Toc483855084)

[1.1 Інформаційна система 6](#_Toc483855085)

[1.2 Загальний огляд прийому та обробки замовлень. 8](#_Toc483855086)

[1.2.1 Власний call-центр чи аутсорсинг 8](#_Toc483855087)

[1.2.2 Що врахувати? 8](#_Toc483855088)

[1.2.3 Що повинен знати і вміти оператор call-центру? 9](#_Toc483855089)

[1.2.4 Альтернативні способи обробки замовлень 9](#_Toc483855090)

[1.3 Підбір інструментів для розробки автоматизованої інформаційної системи. 11](#_Toc483855091)

[1.3.1 Мова програмування C Sharp 11](#_Toc483855092)

[1.3.2 Мова розмітки XAML 13](#_Toc483855093)

[1.3.3 Графічна Підсистема Windows Presentation Foundation 15](#_Toc483855094)

[Переваги WPF над WinForms 16](#_Toc483855095)

[1.3.4 Microsoft Visual Studio 2015 17](#_Toc483855096)

[1.3.5 Патерн Model-View-ViewModel 19](#_Toc483855097)

[1.3.6 Патерн Singleton 21](#_Toc483855098)

[1.3.7 Платформа .NET Framework 22](#_Toc483855099)

[2.0 ОРГАНІЗАЦІЙНО-ТЕХНОЛОГІЧНА ЧАСТИНА 24](#_Toc483855100)

[2.1 Постановка задачі 24](#_Toc483855101)

[2.2 Діаграма варіантів використання для інформаційної системи 25](#_Toc483855102)

[2.3 Схема бази даних 26](#_Toc483855103)

# ВСТУП

Логістика (від грецької λογιστική – облік) означає наука про оптимальне управління матеріальними, інформаційними та фінансовими потоками в економічних адаптивних системах.

Слово “логістика” відоме ще за часів Римської імперії, де службовці, що займались розподілом продуктів харчування, називалися “логістами”. Науково логістика стала завдяки розвитку військової справи. Так візантійський цар Леон VI вважав, що завдання логістики є сплачування данину армії, належно постачати їй зброю і військове майно.

Це визначення, не зважаючи на військову специфіку, має принципову схожість із сучасними формулюваннями завдань логістики. Тільки через 1000 років, в час другої світової війни, принципи логістики почали з успіхом втілюватись в життя, коли США, СРСР та інших країни були виконані роботи з вивченням властивостей операцій управління матеріальними потоками. Першим автором праць з логістики вважають французького військового фахівця початку ХХ століття А.Г. Джаміні, котрий визначав логістику як “практичне мистецтво руху військ”.

В наш час великого перемир’я між народами світу, йдуть значні зміни в понятті логістики. На сьогоднішній день логістика використовується у всіх сферах транспортування, зберігання та обробки будь-яких предметів. Однак не зважаючи на те, що комп’ютери виконують левову частку роботи, логістику і по сьогодні виконують вручну. Працівникам необхідно самостійно обробляти замовлення, створювати оптимальні рішення. Саме тому розробка інформаційної системи обробки замовлень є дуже актуальною темою, адже швидке, а головне оптимальне оброблення замовлень забезпечує успіх роботи великих фірм, яким доводиться робити великі затрати наймаючи багатьох фахівців по обробці замовлень.

Метою даної дипломної роботи є розробка інформаційної системи для підтримки діяльності компанії з продажу техніки. Дана програма повинна вміти обробляти замовлення різної техніки з високою швидкістю, оптимально.

Створення і розробка даної програми включає в себе:

* Затвердження первинного технічного завдання та первинних вимог щодо розробки інформаційної системи.
* Визначення алгоритму роботи програми, тобто процес, який дозволить швидко і ефективно розподілити замовлення між складами.
* Дизайн програми, створення нових графічних елементів.
* Написання програмного коду, створення баз даних та інших вузлів для успішної роботи програми.
* Тестування готового продукту.

# 1. ОПИСОВА ЧАСТИНА

## 1.1 Інформаційна система

Інформаційна система (від англійської *Information system)* – сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів.

Таке визначення може бути задовільним тільки при найбільш узагальненій і неформальній точці зору і підлягає подальшому уточненню. Інформаційні системи діють в Україні під назвою “автоматизовані системи(АС)”.

**Класифікація( за ступенем автоматизації)**

***Ручні ІС***

В залежності від ступеня (рівня) автоматизації виділяють ручні, автоматизовані й автоматичні інформаційні системи.

***Автоматизовані ІС***

частина функції (підсистем) керування або опрацювання даних здійснюється автоматично, а частина — людиною.

***Автоматичні ІС***

усі функції керування й опрацювання даних здійснюються технічними засобами без участі людини (наприклад, автоматичне керування технологічними процесами).

**Історія**

Інформаційні системи здавна знаходять (в тому чи іншому вигляді) досить широке застосування в життєдіяльності людства. Це пов'язано з тим, що для існування цивілізації необхідний обмін інформацією — передача знань, як між окремими членами і колективами суспільства, так і між різними поколіннями.

Інформаційні системи існують з моменту появи суспільства, оскільки на кожній стадії його розвитку існує потреба в управлінні. Місією інформаційної системи є переробка інформації, потрібної для ефективного цправління всіма ресурсами організації, створення інформаційного та технічного середовища для управління її діяльністю.

Інформаційна система може існувати і без застосувань комп’ютерної техніки – це питання економічної доцільності.

В будь-якій інформаційній системі управління вирішуються задачі трьох типів:

* задачі оцінки ситуації (деколи їх називають задачами розпізнавання образів);
* задачі перетворення опису ситуації (розрахункові задачі, задачі моделювання);
* задачі прийняття рішень (в тому числі і оптимізаційні).

Найдавнішими і найпоширенішими інформаційними системами слід вважати бібліотеки. І дійсно, здавна в бібліотеках збирають книжки, зберігають їх, дотримуючись певних правил, створюють певні каталоги різного призначення для полегшення доступу до книжкового фонду. Видаютсья спеціальні журнали та довідники, що інформують про нові надходження, ведеться облік видачі.

Найстаріші інформаційні системи повністю базувалися на ручній праці. Пізніше їм на зміну прийшли різні механічні пристрої для обробки даних. Наступним кроком стало впровадження автоматизованих інформаційних систем(АІС), тобто систем, де для забезпечення інформаційних потреб користувачів використовуєтсья електронна обчислювальна машина зі своїми носіями. В наш час – епоху інформаційної револіції – розробляється і впроваджується велика кількість найрізноманітніших автоматизованих інформаційних систем з дуже широким спектром використання.

## 1.2 Загальний огляд прийому та обробки замовлень.

### 1.2.1 Власний call-центр чи аутсорсинг

Можна скористатися послугами аутсорсингової компанії і за рахунок їхніх ресурсів організувати роботу call-центру. Тим не менш, у власної служби підтримки є найважливіша перевага — компетентність. Виключно співробітникам відомі тонкощі конкретної сфери.  
Продаж непопулярного товару тягне за собою досконале знання всіх його тонкощів. Тут простіше створити call-центр з власних фахівців, ніж навчити непрофесіоналів. Низька компетенція спричинить за собою зниження числа замовлень.

Якщо ви володієте і оперуєте персональними даними клієнтів, то аутсорсингова компанія не може бути притягнута. Це можна трактувати як передачу персональних даних третім особам без згоди суб’єкта, що переслідується законом.

### 1.2.2 ****Що врахувати?****

#### Складні послуги

Для їх надання треба кілька контактів з клієнтом. Спочатку обговорення потреби клієнта, потім відправка комерційної пропозиції. Наступний дзвінок буде направлений на обговорення КП. Як правило, останній дзвінок зводиться до домовленості про початок співпраці (виїзд замірника, внесення часткової передоплати і т. П.).

Розподіл зусиль. Наприклад, у вашому call-центрі є 20 співробітників. Половина з них може з 9 до 12 години приймати вхідні дзвінки, а з 12 до 16 передзвонювати за наявними заявками. В цей же час друга половина call-центру в першій половині дня передзвонює клієнтові, а в другій — приймає вхідні.

#### ****Ієрархія****

Краще розділити товари на категорії: наприклад, складні і прості, і для кожної створити групу операторів. Таким чином, досвідчений консультант буде займатися тими завданнями, які відповідають його знань, а новачок або стажер буде розуміти, до чого прагнути. В кінцевому підсумку це призведе до формування компетентної команди з хорошою мотивацією до розвитку.

### ****1.2.3 Що повинен знати і вміти оператор call-центру?****

#### ****Дані про клієнта****

Важливо отримати якомога більше інформації про клієнта (історія покупок, плани про придбання, матеріальне становище, емоційні характеристики тощо. Д.). Ясна річ, що такі речі не з’ясовуються питанням «в лоб», для цього необхідний певний досвід спілкування з клієнтом. Якщо відповіді на питання односкладові, спроба вивести на діалог може завершитися роздратуванням з боку клієнта. Всі дані повинні фіксуватися. Краще, якщо це буде бланк, де потрібно ставити галочки у відповідних колонках. Це заощадить час.

#### Розмова

Є сенс спілкуватися з клієнтом в такій манері, в якій він сам звик. Тобто не квапити, якщо він хоче поговорити, і підтримувати стриманий стиль спілкування, якщо розмова лаконічний. Це значить, що оцінювати потрібно не тільки знання оператора про товар, а й уміння «відчувати» настрій клієнта.  
Активне слухання. Не можна допускати, щоб клієнт говорив в тишу. Уточнюйте, перефразуйте останні слова, підсумовує. Загалом, дайте зрозуміти, що ви розумієте потребу клієнта і знаєте, як йому допомогти. Він буде вам тільки вдячний.

Незнання. Оператор може не володіти будь-якою інформацією достовірно, але це не дає йому права говорити «я не знаю», «може бути», «можливо». Це створить враження, що людина не на своєму місці. Якщо оператор не знає відповіді на питання, можна ввічливо попросити людину залишатися на лінії і відключитися для уточнення інформації. На це є всього кілька секунд, більше клієнт чекати не буде.

### ****1.2.4 Альтернативні способи обробки замовлень****

#### Зворотній дзвінок

Це лаконічна форма на сайті, де клієнту пропонується залишити свій номер телефону і бажаний час для отримання дзвінка. Це звільняє клієнта від очікування, коли звільниться оператор, а також від оплати переговорів. Тим не менш, витрати вам доведеться взяти на себе. Cкриптов форми замовлення зворотного дзвінка можна скачати в інтернеті, наприклад Callme 2.0, CoMagic, CallCons. Є безкоштовні версії і сервіси за гроші.

#### Пошта

Якщо клієнт вирішив зв’язатися з вами через електронну пошту, не змушуйте його змінити спосіб зв’язку. Щоб йому було комфортно з вами спілкуватися, пам’ятайте кілька простих правил:

Не змушуйте чекати відповіді. Через 2 години клієнт і зовсім забуде, що звернувся до вас.

Якщо ви отримали лист, в якому кілька питань, відповідайте по суті і нумерованим списком. Задавайте питання так само.

Пояснюючи, що входить у вартість, вказуйте повну комплектацію (мобільний телефон, зарядний пристрій, карта пам’яті, навушники, запасна батарея і т. П.). Слідкуйте, щоб клієнт отримав все перераховане.  
Вгадував запитання і відразу давайте на них відповіді в листі. Це скоротить кількість листів, який необхідно відправити, і не залишить у клієнта сумнівів.  
Не міняйте тему листа.

Наприкінці листа робіть «висновки» і натякайте на наступний крок, який потрібно від клієнта.

Робіть так, щоб ваш лист було останнім. Подякуйте за увагу, замовлення, оплату тощо.

#### Онлайн консультант

Форма на сайті пропонує поставити запитання онлайн-консультанту або зробити замовлення. Як правило, супроводжується ім’ям і фото. Щоб ефективно взаємодіяти з клієнтом, необхідний зручний сервіс. Таких достатньо багато: SiteHeart відповідає на стандартні запитання заготовленими відповідями і надсилає на пошту контакти потенційно зацікавлених відвідувачів; JivoSite, де можна вести переписку відразу в п’яти чатах; RedHelper, де в режимі реального часу консультант може спостерігати за набором повідомлення користувача і з’ясувати джерело переходу.

Дзвінок з сайту. Навіть якщо у вашого клієнта не виявиться телефону під рукою, він зможе подзвонити прямо з сайту. Ви самі вибираєте, на які номери будуть надходити дзвінки та в яку країну. Ця послуга можлива завдяки сервісам Zingaya, LiveTex, Mango Office, JivoSite та ін. висновки.

## 1.3 Підбір інструментів для розробки автоматизованої інформаційної системи.

### 1.3.1 Мова програмування C Sharp

C Sharp(також відомий під назвою C#) – об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research.

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники та функції-члени класів, атрибути, події, властивості, вийнятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов С++, Delphi, Smalltalk – С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів.

На сьогоднішній день C# активно використовується багатьма фірмами у розробці різного програмного забезпечення, адже дана мова є дуже проста у розумінні, виправляє багато помилок своїх попередників і супроводжується постійною підтримкою новими оновленнями

Версії:

Версія 1.0

Проект C# був початий в грудні 1998 і отримав кодову назву COOL (C-Style Object Oriented Language). Версія 1.0 була анонсована разом з платформою .NET у червні 2010 року. Перша версія С# була дуже схожа на Java 1.4

Версія 2.0 (жовтень 2003)

Появилось багато нових можливостей, як: часткові типи, анонімні методи, узагальнені або параметризовані типи.

Версія 3.0 (вересень 2005)

Додані ключові слова select, from, where, що дозволяють робити запити SQL, XML, LINQ. Також була добавлена ініціалізація об’єкта з властивостями.

Версія 4.0 (наприкінці 2008)

Добавлена можливість використовувати пізного зв’язку для роботи з мовами динамічної типізації (Ruby, Python), з СОМ – об’єктами, з об’єктами зі змінною структурою

Версія 5.0 ( випущена одночасно з Visual Studio 2012)

Добавлені асинхронні методи.

Версія 6.0 (випущена одночасно з Visual Studio 2013)

Оператор nameof, інтерполяція стрічок, ініціалізатори індексів, ініціалізатори властивостей, фільтр вийнятків, умовна операція з null.

Версія 7.0 (випущена одночасно з Visual Studio 2017)

Добавлена можливість створювати змінні загального типу, які прийматимуть кільки вихідних параметрів. Добавлені локальні функції, бінарні літерали.

Приклад простого коду мовою C#:

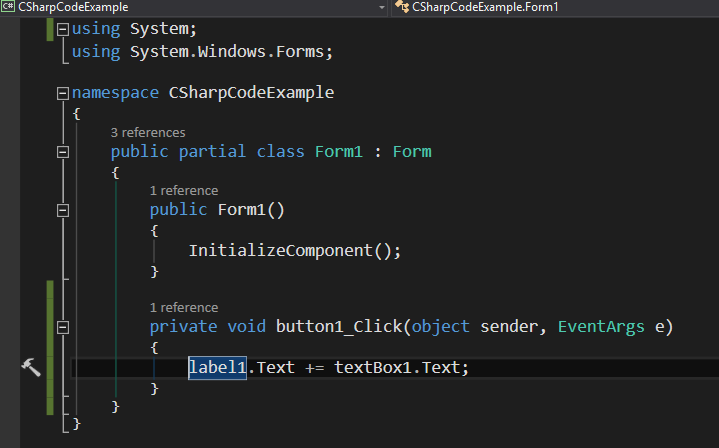


Рис. 1.0 Приклад C# коду

При компіляції результатом роботи програми буде:

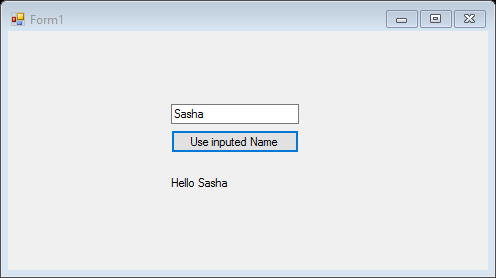


Рис. 2.0 Результат виконання програми

### 1.3.2 Мова розмітки XAML

XAML (від англійської *eXtensible Application Markup Language*) – декларативна мова розмітки. З точки зору моделі програмування .NET Framework мова XAML спрощує створення користувацького інтерфейсу для програми .NET Framework. Можна створити видимі елементи інтерфейсу користувача в декларованій розмітці XAML а потім відокремити визначення користувача інтерфейсу від логіки часу виконання, використовуючи файл коду програмної частини, приєднані до розмітки за допомогою визначень поділюваних класів. Мова XAML безпосередньо представляє створення екземплярів об’єктів в конкретному наборі резервних типів, визначених у збірках. У цьому полягає її відмінність від більшості інших мов розмітки, які, як правило, є інтерпретованими мовами без прямого зв’язку з системою резервних типів. Мова XAML забезпечує робочий процес, що дозволяє декільком учасникам розробляти користувацький інтерфейс і логіку програми, використовуючи потенційно різні засоби.

При поданні у вигляді тексту файли XAML є XML – файли, які зазвичай мають розширення .xaml. Файли можна зберігати в будь-якому кодуванні, що підтримує XML, але зазвичай використовується кодування UTF-8.

Приклад простого коду мовою XAML:

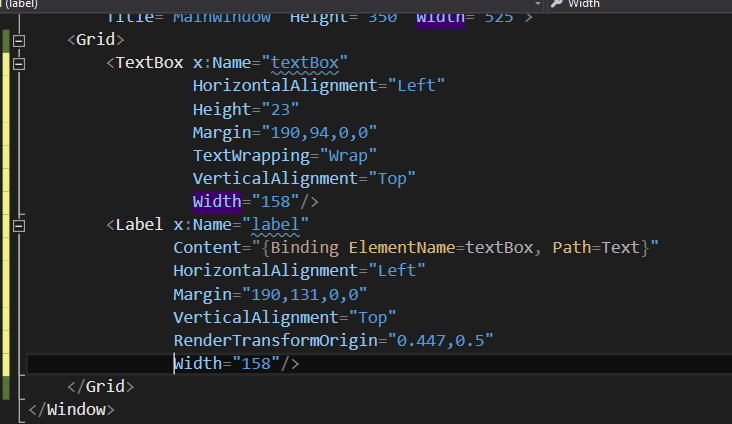


Рис. 3.0 Приклад XAML коду.

При компіляції результатом роботи програми буде:

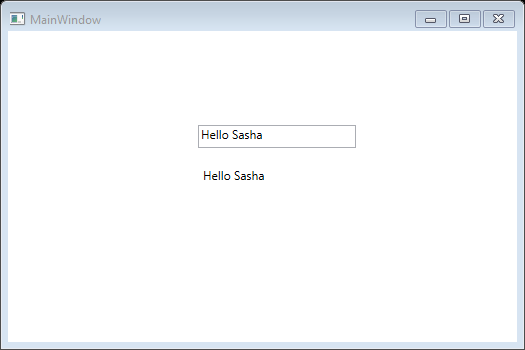


Рис. 4.0 Результат роботи XAML коду.

Якщо порівняти даний приклад з попереднім, то в цьому прикладі я зміг добитись того ж результату без використання кнопок і без жодної строки коду мовою програмування.

### 1.3.3 Графічна Підсистема Windows Presentation Foundation

Windows Presentation Foundation (WPF кодова назва – Avalon) – графічна (презентаційна) підсистема в складі .NET Framework 3.0, що має відношення до XAML. WPF разом з .NET Framework 3.0 вбудована в Windows Vista, а також доступна для інсталяції в Windows XP Service Pack 2.

Вікна (windows) є основними елементами в будь-якому настільному застосунку – настільки “основними”, що в їхню честь навіть була названа операційна система Windows. І хоча WPF є модель для створення навігаційних застосунків, що розподіляють задачі по окремих сторінках, вікна залишаються переважно технологією для створення застосунків.

WPF – революційна технологія, яка була покликана вирішити різні проблеми програмування під Windows і вивести його на абсолютно новий рівень. Дана технологія здатна намалювати чітку лінію яка розділяє написання інтерфейсу та самого програмування.

### Переваги WPF над WinForms

Веб подібна модель розробки

Замість того, щоб фіксувати елменти керування на місці з конкретними координатами, WPF підтримує данімичний і гнучкий потік, який розміщує елементи керування на основі іх заповненості. В результаті виходить користувацький інтерфейс, який може бути адаптованим для відображення високо динамічного заповнення.

Багата модель розробки

Замість малювання пікселів в WPF ви маєте справу з примітивами – базовими фігурами, блоками тексту і іншими графічними інгридієнтами. Крім того, доступні інші можливості, як прозорість, розмитість, підтримка трьох вимірної графіки.

Розвинута текстова модель

Після багатьох років нестандартної обробки текстів WPF почав представляти Windows-додатки, можливість відображення розширеного стилізованого тексту в будь-якому місці користувацького інтерфейсу

Анімація

В WPF нема потреби використовувати таймер для того, щоб змусити форму наново себе намалювати. Замість цього доступна анімація – невід’ємна частина платформи.

Підтримка аудіо та відео

Такі користувацькі інтерфейси як Windows Forms, були досить сильно обмежені в роботі з мультимедіа. Однак WPF включає підтримку і відтворення буль-якого аудіо або відеофайла, який підтримує Windows Media. Що більш цікаво, так це те, що WPF здатний надати вам інструменти, які дозволять вам виконувати різні дії по інтеграції відео вікна на поверхню трьох мірного куба.

Стилі та шаблони

Користувач має змогу самостійно створювати різні стилі які по різному оформлюють різні елементи WPF. Колір елемента, тригери, які реагують на ту чи іншу подію, керування візуальними властивостями елементів. І створення різних шаблонів, які дозволяють докорінно змінювати зовнішність та навіть функціонал елемента WPF без жодної строки коду.

Binding

Ще одна унікальна властивість WPF, яка дозволяє встановлювати зв’язок з програмою без жодної строки коду. Ви можете приєднювати до користувацького інтерфейса різні колекції, команди, інші властивості.

### 1.3.4 Microsoft Visual Studio 2015

Miscrosoft Visual Studio – серія продуктів фірми Microsoft, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів.Напротязі багатьох років займає високі позиції на ринку продуктів по розробці програмного забезпечення. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримують Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .Net Framework, .Net Compact Framework та Microsoft Silverlight.

#### Компоненти

Visual Studio включає в себе багато компонентів :

* Visual Basic
* Visual C++
* Visual C#
* Visual F#
* Visual Studio Debugger

Багато варіантів постачання даних:

* Microsoft SQL Server
* MSDE Visual Source Safe

#### Переваги та недоліки Visual Studio 2015

**Переваги:**

* В Visual Studio 2015 суттєво змінена підтримка багатьох цільових платформ: окрім базової Windows з’явилась можливість будувати проекти на IOS та Android. Для розробників ігор була добавлена підтримка фреймворку Unity. Також дана версія включає в себе .Net Framework 4.6 та підтримку універсальної платформи Windows 10.
* Visual Studio побудована в архітектурі, що підтримує можливість використання доповнень (Add-Ins), - плагінів від сторонніх розробників, до дозволяє розширювати можливість середовища розробки. Серед найпопулярніших доповнень варто виділити: DevPartner Studio, ReSharper, Visual Assist.
* Серед дрібних але значних особливостей варто виділити: стабільність, простота, автоматичне знаходження помилок.

**Недоліки:**

* Великі вимоги за ресурсів комп’ютера;
* Великий набір інструментів може заплутати початківця.

#### Microsoft SQL Server

Miscrosoft SQL Server – комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів – Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керуваннями базами даних.

### 1.3.5 Патерн Model-View-ViewModel

Model-View-ViewModel – це шаблон проектування, що застосовується під час проектування архітектури застосунків (додатків). Публічно вперше був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model. MVVM орієнтований на такі сучасні платформи розробки, як Windows Presentation Foundation та Silverlight від компанії Microsoft.

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель. Модель представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і оброблює більшість, якщо не всю, логіку відображення даних. Модель модель представлення може також реалізовувати патерн медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримуються представленням.

#### Призначення

MVVM використовується для відокремлення моделі та її відображення. Необхідністю цього є надання можливості змінювати їх незалежно одну від одної. Наприклад, розробник працює над логікою роботи з даними, а дизайнер — з користувацьким інтерфейсом.

MVVM була створена з метою поділу праці дизайнера і програміста, що є неможливим, коли Java-розробник намагається побудувати GUI в Swing або розробник на Visual C++ намагається створити користувальницький інтерфейс в MFC. Розробники — кмітливі хлопці і мають безліч навичок, але створення зручних і привабливих інтерфейсів вимагає абсолютно інших талантів, ніж ті, якими вони володіють. Ця робота більше підходить для дизайнерів інтерфейсів. Хороші дизайнери інтерфейсів краще знають, чого бажають користувачі, ніж експерти в області проектування і написання коду. Зрозуміло, буде краще, якщо дизайнер інтерфейсів створить інтерфейс, а розробник напише код, який реалізує логіку цього інтерфейсу, але технології типу Swing або MFC просто-напросто не дозволяють чинити таким чином.

#### Використання

MVVM зручно використовувати замість привичного MVC та йому подібних у тих випадках, де ведеться розробка, присутнє “зв’язування даних”.

В MVC/MVP зміни у користувацькому інтерфейсі не впливають безпосередньо на модель, а йдуть через Контролер/Presenter. У таких технологіях, як WPF та Silverlight, присутня концепція «зв'язування даних», що дозволяє зв'язувати дані із візуальними елементами в обидві сторони.

Архітектура MVVM вирішує цю проблему ясним поділом відповідальності:

* Розробка користувацького інтерфейсу здійснюється дизайнером інтерфейсів за допомогою технології, більш-менш природної для такої роботи (XML);
* Логіка користувацького інтерфейсу реалізується розробником як компонент ViewModel;
* Функціональні зв'язки між користувацьким інтерфейсом та ViewModel реалізуються через біндинги (bindings), які, по суті, є правилами типу «якщо кнопка A була натиснута, повинен бути викликаний метод onButtonAClick() з ViewModel». Біндинги можуть бути написані в коді або визначені декларативним шляхом (Android використовує обидва типи).

Архітектура MVVM використовується в тому чи іншому вигляді усіма сучасними технологіями, наприклад Microsoft WPF і Silverlight, Oracle JavaFX, Adobe Flex, AJAX.

#### Опис

Шаблон MVVM ділиться на 3 частини:

* **Модель** (Model), як і в класичному шаблоні MVC, Модель являє собою фундаментальні дані, що необхідні для роботи застосунку;
* **Вид/(Вигляд)** (View) як і в класичному шаблоні MVC, Вигляд — це графічний інтерфейс, тобто вікно, кнопки тощо;
* **Vодель вигляду** (ViewModel, що означає «Model of View») з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних з Моделі, які мають зв'язуватись. Тобто вона містить Модель, яка перетворена до Вигляду, а також містить у собі команди, якими може скористатися Вигляд для впливу на Модель. Фактично ViewModel призначена для того, щоб: здійснювати зв’язок між моделлю та вікном, відслідковувати зміни та опрацьовувати логіку роботи View.

****

Рис. 5.0 схема роботи патерна MVVM (Model-View-ViewModel)

### 1.3.6 Патерн Singleton

Одинак (від англійської Singelton) – шаблон проектування, відноситься до класу твірних шаблонів. Гарантує, що клас матиме тільки один екземпляр, і забезпечує глобальну точку доступу до цього екземпляра.

#### Мотивація

Для деяких класів важливо, щоб існував тільки один екземпляр. Наприклад, хоча у системі може бути декілька принтерів, може бути тільки один спулер. Повинна бути тільки одна файлова система та тільки один активний віконний менеджер.

Глобальна змінна не вирішує такої проблеми, бо не забороняє створити інші екземпляри класу.

Рішення полягає в тому, щоб сам клас контролював свою “унікальність”, забороняючи створення нових екземплярів, та сам забезпечував єдину точку доступу. Це є призначенням шаблону Одинак.

#### Застосування

Слід використовувати шаблон Одинак коли:

* Повинен бути тільки один екземпляр класу, що легко доступний всім клієнтам;
* Єдиний екземпляр повинен розширюватись шляхом успадкування, та клієнтам потрібно мати можливість працювати з розширеним екземпляром не змінюючи свій код.

#### Структура

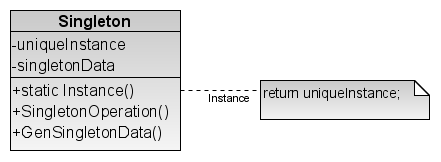


Рис. 6.0 Діаграма класів, що описує структуру шаблону Singleton

**Singleton** – одинак

* Визначає операцію Instance, котра дозволяє клієнтам отримувати доступ до єдиного екземпляру. Instance – це операція класу.
* Може нести відповідальність за створення власного унікального екземпляру.

### 1.3.7 Платформа .NET Framework

Microsoft .NET – програмна технологія, запропонована фірмою Microsoft, як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідеї та принципів, покладених в технології Java. Одною з ідей .NETє сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість.

Як і технологія Java, середовище розробки .NET створює байт-код, призначений для виконання віртуальною машиною. Вхідна мова цієї машини в .NET називається CIL (Common Intermediate Language), також відома як MSIL (Microsoft Intermediate Language), або просто IL. Застосування байт-кода дозволяє отримати крос-платформовість на рівні скомпільованого проекту, а не рівні сирцевого тексту, як наприклад в мові програмування C. Перед запуском збірки в середовищі виконання (CLR) байт-код перетворюється вбудованим середовищем JIT-компілятором (Just in time, компіляція на льоту) в машинні коди цільового процесора.

Слід зазначити, що одним з перших JIT-компіляторів для Java був також розроблений фірмою Microsoft (тепер в Java використовується досконаліша багаторівнева компіляція – Sun HotSpot). Сучасна технологія динамічної компіляції дозволяє досягнути аналогічного рівня швидкодії з традиційними “статичними” компіляторами і питання швидкодії часто залежить від якості того чи іншого компілятора.

# 2.0 ОРГАНІЗАЦІЙНО-ТЕХНОЛОГІЧНА ЧАСТИНА

## 2.1 Постановка задачі

Необхідно розробити та інформаційну систему, завданням якої буде забезпечення обробки замовлень фірми з продажу техніки.

Дана інформаційна система повинна задовольняти наступні вимоги:

* зручний та інтуїтивно зрозумілий інтерфейс;
* кастомізація інтерфейсу;
* зміна мови інтерфейсу;
* перегляд необроблених замовлень;
* перегляд оброблених замовлень;
* перегляд відкладених замовлень;
* перегляд відмінених замовлень;
* зміна стану замовлення;
* оброблення замовлення;
* оброблення замовлення з функцією втручання в процес обробки;
* надсилання E-mail повідомлень на електронну пошту складів для інформування їх про те, що замовлення виконане/скасоване;
* скасування обробленого замовлення;
* запис в файл важливу інформацію у разі критичної помилки програми;
* надання користувачу можливості записати в файл усе що він робив до появи критичної помилки;
* показ результату оброблення замовлень;

Інформаційна система повинна бути реалізована за допомогою платформи .Net Framework, мови програмування C#, набору бібліотек ADO.NET та SQL Server.

## 2.2 Діаграма варіантів використання для інформаційної системи

Проектування та розробка сучасної інформаційної підсистеми вимагає створення графічних діаграм для того, щоб показати властивості підсистем. Для створення діаграм використовують уніфіковану мову моделювання UML. Мова UML використовується у парадигмі [об'єктно-орієнтованого програмування](http://uk.wikipedia.org/wiki/%D0%9E%D0%B1%E2%80%99%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F). Вона містить багато діаграм для графічного позначення різних властивостей і описів підсистеми. Однією із найбільш використовуваних діаграм є діаграма варіантів використання або прецедентів.

Користувач даної інформаційної системи може змінювати інтерфейс, його мову і колір, змінювати стан замовлень і обробляти їх, при цьому зможе напряму редагувати процес обробки замовлень. Виконані замовлення автоматично відправляють електронний лист на електронну пошту сладу. У разі потреби відмінити замовлення користувач може з легкістю це зробити тількома кліками. Лист про відмінені замовлення автоматично надходять на електронну пошту складів попереджаючи про те, що замовлення скасоване. Також інформаційна система має вбудовану підсистему “Collector”, яка виконує функцію запису усіх дій користувача і виконує функцію перехоплення помилок. Також дана підсистема надає користувачу можливість описати все, що користувач робив останнім часом для отримання максимальної кількості інформації. На діаграмі варіантів використання такі відношення позначаються пунктирною лінією з написом “extend”, який доповнює варіант використання іншими варіантами, що спрацьовують при деяких умовах.

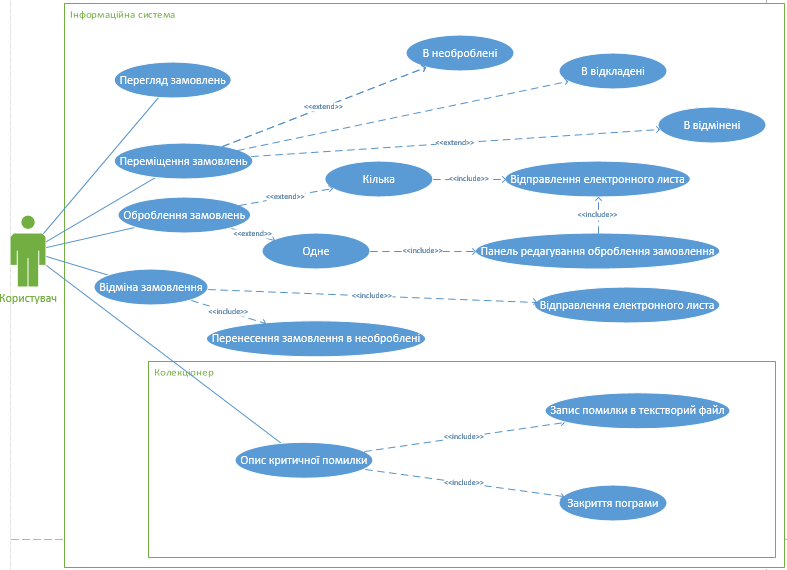


Рис. 2.2 Діаграма варіантів використання інформаційної системи та її підсистеми для інтернет магазину комп’ютерної техніки

## 2.3 Схема бази даних

Структура бази даних для інформаційної системи інтернет магазину з продажі комп’ютерної техніки складається з шести таблиць.

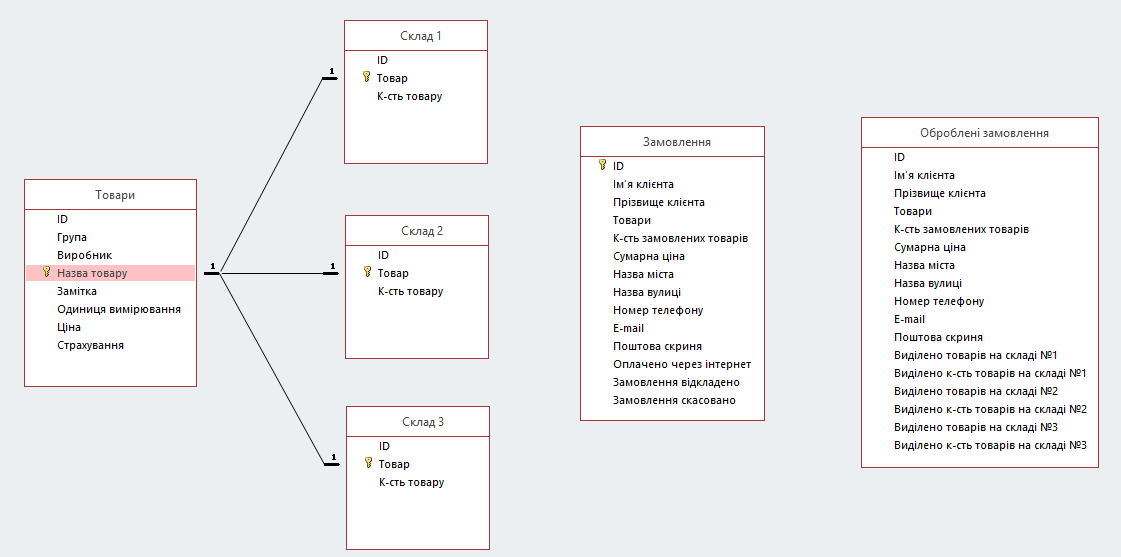


Рис.2.3 Схема бази даних інформаційної системи інтернет магазину з продажі комп’ютерної техніки.

База даних складається з таких таблиць:

* Товари – зберігається інформація про товари;
* Склад 1 – зберігається інформація про товари та їх кількість на складі 1;
* Склад 2 – зберігається інформація про товари та їх кількість на складі 2;
* Склад 3 – зберігається інформація про товари та їх кількість на складі 3;
* Замовлення – зберігається інформація про замовлення та їх стани;
* Оброблені замовлення – зберігаєтсья інформація про оброблені замовлення.

Таблиця “Товари” має таку множину об’єктів:

* ID – ідентифікаційний номер товару;
* Група – група до якої відноситься товар( процесори, відеокарти, інше) ;
* Виробник – назва виробника даного товару;
* Назва товару – загальна назва товару( включаючи його характеристики та групу) ;
* Замітка – короткий опис товару( детальні характеристики, інша додаткова інформація) ;
* Одиниця вимірювання – різні товари мають різні одиниці вимірювання ( в більшості випадків вимірюється в штуках) ;
* Ціна – ціна даного товару ( в доларах) ;
* Страхування – деякі товари мають строк гарантії на випадок поламки;

Таблиця “Склад 1” має таку множину об’єктів:

* ID – ідентифікаційний номер товару на даному складі;
* Товар – назва товару який доступний на даному складі;
* К-сть товару – кількість товару яка доступна на даному складі;

Таблиця “Склад 2” має таку множину об’єктів:

* ID – ідентифікаційний номер товару на даному складі;
* Товар – назва товару який доступний на даному складі;
* К-сть товару – кількість товару яка доступна на даному складі;

Таблиця “Склад 3” має таку множину об’єктів:

* ID – ідентифікаційний номер товару на даному складі;
* Товар – назва товару який доступний на даному складі;
* К-сть товару – кількість товару яка доступна на даному складі;

Таблиця “Замовлення” має таку множину об’єктів:

* ID – ідентифікаційний номер замовлення;
* Ім’я клієнта - Ім’я замовника;
* Прізвище клієнта – прізвище замовника;
* Товари – перелік товарів, які замовив замовник;
* К-сть замовлених товарів – перелік кількості замовлених товарів;
* Сумарна ціна – сумарна ціна замовлення ( в доларах) ;
* Назва міста – місто замовника;
* Назва вулиці – вулиця проживання замовника;
* Номер телефону – номер телефону замовника;
* E-mail – електронна пошта замовника;
* Поштова скриня – номер поштової скрині замовника;
* Оплачено через інтернет – чи оплатив покупець товар через інтернет;
* Замовлення відкладено – чи відкладено замовлення чи ні;
* Замовлення скасовано – чи скасовано замовлення;

Таблиця “Замовлення” має таку множину об’єктів:

* ID – ідентифікаційний номер замовлення;
* Ім’я клієнта - Ім’я замовника;
* Прізвище клієнта – прізвище замовника;
* Товари – перелік товарів, які замовив замовник;
* К-сть замовлених товарів – перелік кількості замовлених товарів;
* Сумарна ціна – сумарна ціна замовлення ( в доларах) ;
* Назва міста – місто замовника;
* Назва вулиці – вулиця проживання замовника;
* Номер телефону – номер телефону замовника;
* E-mail – електронна пошта замовника;
* Поштова скриня – номер поштової скрині замовника;
* Виділено товарів на складі 1 – назви товарів які були відправлені зі складу 1;
* Виділено к-сть товарів на складі 1 – кількість товарів які були відправлені зі складу 1;
* Виділено товарів на складі 2 – назви товарів які були відправлені зі складу 2;
* Виділено к-сть товарів на складі 2 – кількість товарів які були відправлені зі складу 2;
* Виділено товарів на складі 3 – назви товарів які були відправлені зі складу 3;
* Виділено к-сть товарів на складі 3 – кількість товарів які були відправлені зі складу 3;

База даних може містити взаємозв'язані таблиці. Зв'язок встановлюється між двома полями двох таблиць. Зв’язувані поля можуть мати різні імена, але повинні мати однакові типи даних. В табл. 2.3 розглянуто зв’язки між таблицями та полями таблиць інформаційної системи інтернет магазину з продажі техніки, які зображені на схемі бази даних.

Таблиця 2.3

Зв’язки між таблицями інформаційної системи інтернет магазину з продажі техніки

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Таблиця | Тип зв’язку | Опис |
| Товари.Назва товару | Склад 1.ID | 1:1 | Для одного товару може бути лише одна кількість на складі |
| Товари.Назва товару | Склад 2.ID | 1:1 | Для одного товару може бути лише одна кількість на складі |
| Товари.Назва товару | Склад 3.ID | 1:1 | Для одного товару може бути лише одна кількість на складі |

## 2.4 Діаграма класів інформаційної системи

Розроблена інформаційна система для інтернет магазину з продажі техніки складається із 31(26 класів логіки, 5 класів вікон) класу, 6 під проектів, 1 інсталятор, 2 перерахувань(Enum), 2 файлів настройок, 5 вікон, 9 файлів локалізації, 14 малюнків, 1 моделі бази даних . Проект інформаційної системи зображений на рис.2.4.1-2.4.5

Класи та взаємодію між класами можна відобразити за допомогою діаграми класів UML. Клас визначає абстрактні характеристики деякої сутності, включаючи характеристики самої сутності та дії, які вона здатна виконувати .

Клас на діаграмі зображається у вигляді прямокутника, розділеного горизонтальними лініями на три частини. У першій частині вказується назва класу. Як правило, ім'я класу складається з кількох слів. Друга частина містить перелік властивостей класу. Третя частина містить перелік методів, що відображають його поведінку.

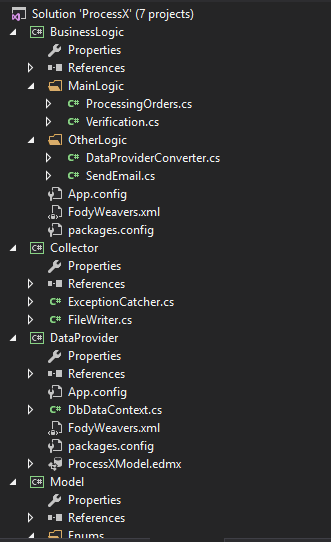


Рис.2.4.1 Проекти ”Бізнес логіки”, ”Колекціонер”,”Провайдер даних”,”Моделі”.

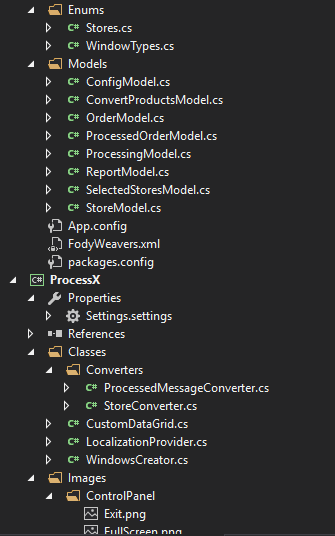


Рис.2.4.2 Проекти ”Моделі”, ”ProcessX”.

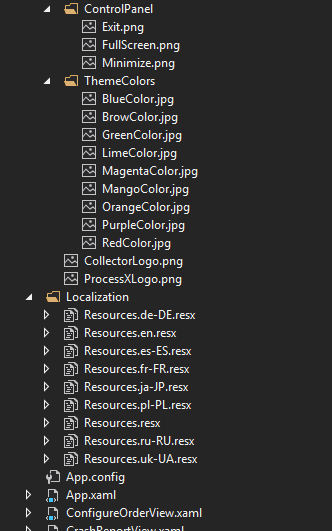


Рис.2.4.3 Проект ”ProcessX”.

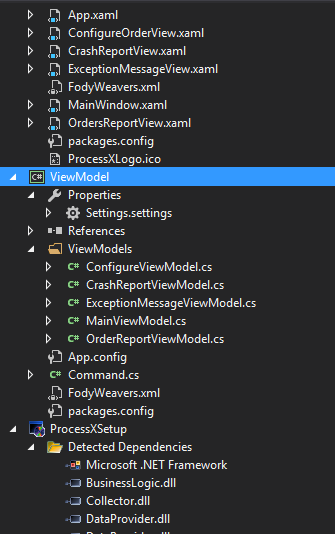


Рис.2.4.4 Проекти”ProcessX”,”Модель інтерфейсу”, ”Інсталятор програми”.

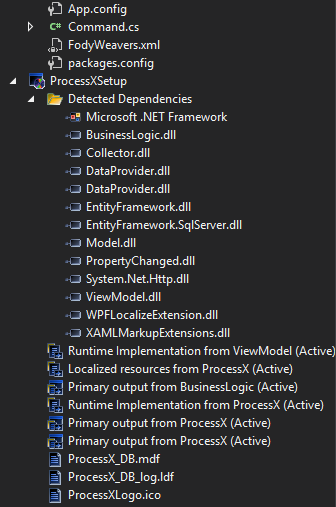


Рис.2.4.5 Проект”Інсталятор програми”

Дана інформаційна система складається з наступних класів:

* ProcessingOrders – клас, який виконує обробку замовлень;
* Verification – клас, який виконує перевірку на наявність товару;
* DataProviderConverter – клас, який конвертує дані з бази даних у зручний для читання формат;
* SendEmail – клас, який виконує функцію надсилання листів;
* ExceptionCatcher – клас, який виконує роль перехоплення помилок програми;
* FileWriter – клас, який записує в текстовий файл усі дії програми;
* DbDataContext – клас, який виконує функцію селекції, модифікації, видалення даних з таблиць бази даних;
* ConfigModel – клас, який виконує роль моделі для збереження даних про кількість товарів на складах і про товари та те, як програма їх розподілила між складами;
* ConvertProductsModel – клас, який виконує роль моделі, для збереження даних про назву товару та кількість яка була видана;
* OrderModel – клас, який виконує роль моделі, для збереження даних таблиці “Замовлення”;
* ProcessedOrderModel – клас, який виконує роль моделі, для збереження даних таблиці “Оброблені замовлення”;
* ProcessingModel – клас, який виконує роль моделі, для збереження даних про те, як програма обробила замовленя( назва та кількість товару, яка була виділена для даного складу);
* ReportModel – клас, який виконує роль моделі, для збереження даних про замовлення( чи було замовлення виконане чи ні);
* SelectedStoresModel – клас, який виконує роль моделі, для збереження даних про кількість товару на складах;
* StoreModel – клас, який виконує роль моделі, для збереження даних таблиці “Склад 1”, “Склад 2”, “Склад 3”;
* ProcessedMessageConverter – клас, який виконує роль конвертера, для конвертації даних оброблених та не оброблених замовлень;
* StoreConverter - клас, який виконує роль конвертера, для конвертації даних складів;
* CustomDataGrid – клас, який виконує роль нового та покращеного елементу інтерфейсу під назвою “Data Grid”;
* LocalizationProvider – клас, який повертає слово з файлів локалізації через ключове слово;
* WindowsCreator – клас, який займається створенням вікон;
* ConfigureViewModel – клас, який виконує роль моделі інтерфейсу по конфігурації обробленого замовлення;
* CrashReportViewModel – клас, який виконує роль моделі інтерфейсу по опису критичної помилки;
* ExceptionMessageViewModel - клас, який виконує роль моделі інтерфейсу по опису помилок;
* MainViewModel – клас, який виконує роль моделі інтерфейсу для головного вікна, і надає стилі дизайну іншим вікнам;
* OrderReportViewModel – клас, який виконує роль моделі інтерфейсу для відображення результатів обробки замовлень;
* Command – клас, який виконує роль помічника для виконання команд;
* App – клас, який запускає головне вікно програми, використовується для перехоплення помилок при запуску програми;
* ConfigureorderView – клас вікна, який виконує базові функції вікна;
* CrashReportView - клас вікна, який виконує базові функції вікна;
* ExceptionMessageView - клас вікна, який виконує базові функції вікна;
* MainWindow - клас вікна, який виконує базові функції вікна;
* OrdersReportView - клас вікна, який виконує базові функції вікна;

Дана інформаційна система складається з наступних проектів:

* DataProvider – проект, який виконує роль провайдера бази даних;
* BusinessLogic – проект, який виконує роль бізнес логіки;
* ViewModel – проект, який виконує роль моделі інтерфейсу;
* ProcessX – головний проект, який містить інтерфейс і виконує роль прийому та показу даних;
* Model – проект, який містить моделі, які використовують в інших проектах;
* Collector – проект, який виконує роль підсистеми, для запису роботи програми в файл та для перехоплення критичних помилок;
* ProcessXSetup – проект, який створює інсталятор програми;

Дана інформаційна система складається з наступних перерахувань:

* Stores – містить в собі список складів;
* WindowTypes – містить в собі список вікон;

Дана інформаційна система складається з наступних файлів налаштувань:

* ProcessX.Settings – зберігає в собі налаштування мови;
* ViewModel.Settings – зберігає в собі налаштування кольору теми, кольору фону, кольору країв, кольору тексту панелі вкладок, розмір шрифту;

Дана інформаційна система складається з наступних файлів локалізації:

* Resources.de-DE – містить локалізацію німецькою;
* Resources.en – містить локалізацію англійською;
* Resources.es-ES - містить локалізацію іспанською;
* Resources.fr-FR - містить локалізацію французькою;
* Resources.ja-JP - містить локалізацію японською;
* Resources.pl-PL - містить локалізацію польською;
* Resources - містить локалізацію за замовчуванням;
* Resources.ru-RU - містить локалізацію російською;
* Resources.uk-UA – містить локалізацію українською;

### 2.4.1 Модель таблиць бази даних

В сучасних інформаційних системах використовують бази даних. Для роботи із ними необхідно щоб система розуміла з якими даними їй прийдеться працювати, тому необхідно використовувати класи з інформацією та описом цих даних.

Діаграма моделі бази даних з інформацією про таблиці бази даних зображена на рис. 2.4.6 Код класу з інформацією про таблиці бази даних наведено в   
додатку А.

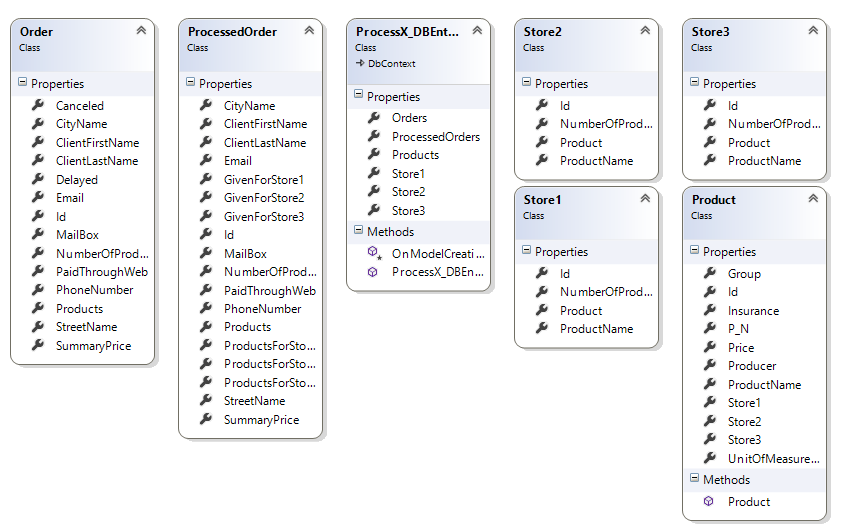


Рис. 2.4.6 Діаграма класів моделі бази даних

Модель ProcessXModel складається з декількох класів:

* Order – в класі містяться дані про замовлення;
* ProcessedOrder – в класі містяться дані про оброблені замовлення;
* Store1 – в класі містяться дані про товари та їх кількість на складі;
* Store2 – в класі містяться дані про товари та їх кількість на складі;
* Store3 – в класі містяться дані про товари та їх кількість на складі;
* Product – в класі містяться дані про товари;
* ProcessX\_DBEntities – клас, який складається з вище перерахованих класів;

### 2.4.2 Клас для роботи з базою даних

Для роботи з БД було створено клас з назвою DbDataContext. В цьому класі реалізовано методи, що вибирають, редагують, записують та видаляють з БД потрібні дані для інформаційної системи за допомогою технології ADO.Net. Діаграма класу зображена на рис. 2.4.7, код класу наведений у додатку Б.

Клас DbDataContext має такі методи:

* SelectAllNotProcessedOrders – методі селекції всіх не оброблених замовлень з таблиці “Замовлення”;
* SelectAllDelayedOrders – метод селекції всіх відкладених замовлень з таблиці “Замовлення”;
* SelectAllProcessedOrders – метод селекції всіх оброблених замовлень з таблиці “Оброблені замовлення”;
* SelectAllCanceledOrders - метод селекції всіх скасованих замовлень з таблиці “Замовлення”;
* SendNotProcessedToDelayed – метод, який відправляє не оброблені замовлення в затримані;
* SendProcessedToDelayed – метод, який відправляє оброблені замовлення в затримані;
* SendDelayedToNotProcessed – метод, який відправляє затримані замовлення в не оброблені;
* SendOrderToCanceled – метод, який відправляє замовлення в скасовані;
* SendCanceledToDelayed – метод, який відправляє скасовані замовлення в відкладені;
* SendOrderToProcessed – метод, який відправляє замовлення в таблицю “Оброблені замовлення”;
* SelectNumberOfProducts – метод селекції кількості конкретного товару;
* RemoveNeededProducts – метод видалення конкретної кількості конкретного товару в конкретний склад;
* AddNeededProduct – метод додавання конкретної кількості конкретного товару в конкретний склад;

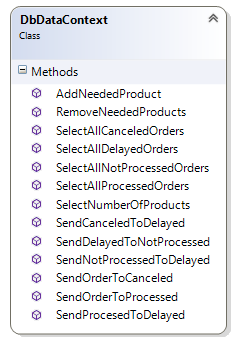


Рис. 2.4.7 Діаграма класу роботи з базою даних інформаційної системи інтернет магазину з продажі техніки.

### 2.4.3 Класи бізнес логіки

Кожна програма містить у собі “бізнес логіку”, котра виконує основну логіку програми. Давий проект складається з головної бізнес логіки та допоміжніх класів.

Клас ProcessingOrders є головним класом бізнес логіки, котрий шукає способи виконати замовлення оптимальним шляхом. Даний клас повертає список котрий містить інформацію про те чи замовлення було виконане чи ні. UML діаграма класу зображена на рис. 2.4.8, а код класу наведений в додатку В.

Клас Verification є другим головним класом бізнес логіки після класу ProcessingOrders, він виконує роль перевірки того, чи може замовлення бути виконаним з даним розподілом товарів. UML діаграма класу зображена на рис. 2.4.9, а код класу наведений в додатку Г.

Клас DataProviderConverter є допоміжнім класом бізнес логіки, він виконує функцію конвертації замовлень в зручний для читання формат, оформлює замовлення, котре буде перенесене в іншу таблицю та виконує роль посередника між провайдером даних та моделлю інтерфейсу. UML діаграма класу зображена на рис. 2.4.10, а код класу наведений в додатку Ґ.

Клас SendEmail є допоміжнім класом бізнес логіки, він виконує функцію надсилання електронних листів на електронну пошту кожного складу коли замовлення було оброблено або відмінено. UML діаграма класу зображена на рис. 2.4.11, а код класу наведений в додатку Д.

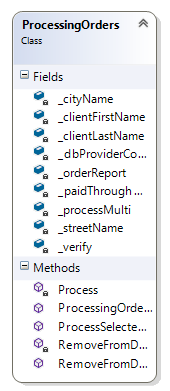


Рис. 2.4.8. UML діаграма класу ProcessingOrders інформаційної системи інтернет магазину з продажі техніки.

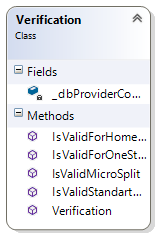


Рис. 2.4.9. UML діаграма класу Verification інформаційної системи інтернет магазину з продажі техніки.

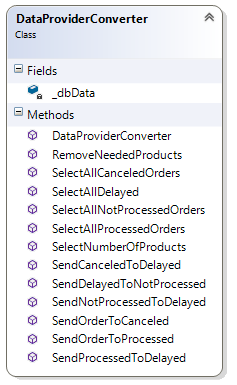


Рис. 2.4.10. UML діаграма класу DataProviderConverter інформаційної системи інтернет магазину з продажі техніки.

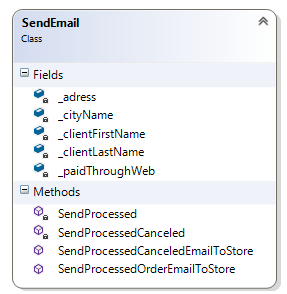


Рис. 2.4.11. UML діаграма класу SendEmail інформаційної системи інтернет магазину з продажі техніки.

### 2.4.4 Класи Моделі інтерфейсу

Кожна програма повинна містити у собі “моделі інтерфейсу”, котрі виконують основні операції програми. Давий проект складається з моделей інтерфейсу та допоміжніх класів.

Клас ConfigureViewModel є класом, котрий виконує роль моделі інтерфейсу. Він містить у собі дані для редагування процесу обробки замовлення, де працівник може напряму впливати на процес обробки замовлень. UML діаграма класу зображена на рис. 2.4.12, а код класу наведений в додатку Е.

Клас CrashReportViewModel є класом, котрий виконує роль моделі інтерфейсу. Він містить у собі дані для запису користувачем усіх дій, які він виконував перш ніж програма видала критичну помилку. UML діаграма класу зображена на рис. 2.4.13, а код класу наведений в додатку Є.

Клас ExceptionMessageViewModel є класом, котрий виконує роль моделі інтерфейсу. Він містить у собі дані для виведення на екран інформації про критичну помилку. UML діаграма класу зображена на рис. 2.4.14, а код класу наведений в додатку Ж.

Клас MainViewModel є головним класом , котрий виконує роль моделі інтерфейсу. Він містить у собі всі дані про стиль інтерфейсу( колір теми, колір фону , розмір шрифту та інші стилі) і приймає головні операції користувача та викликає класи бізнес логіки. UML діаграма класу зображена на рис. 2.4.15, а код класу наведений в додатку З.

Клас OrderReportViewModel є класом, котрий виконує роль моделі інтерфейсу. Він містить у собі дані про те чи замовлення виконано чи ні. UML діаграма класу зображена на рис. 2.4.16, а код класу наведений в додатку І.

Клас Command є допоміжнім класом моделей інтерфейсу. Він наслідує методи інтерфейсу ICommand, котрі дозволяють прив’язувати команди до інтерфейсу прогами. UML діаграма класу зображена на рис. 2.4.17, а код класу наведений в додатку Ї.

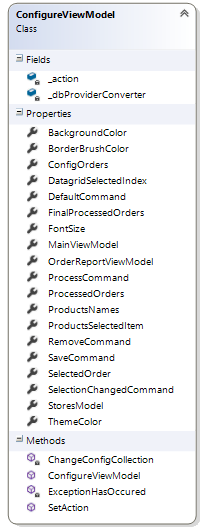


Рис. 2.4.12. UML діаграма класу ConfigureViewModel інформаційної системи інтернет магазину з продажі техніки.

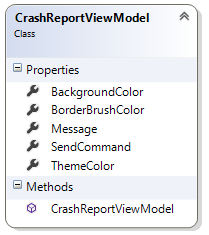


Рис. 2.4.13. UML діаграма класу CrashReportViewModel інформаційної системи інтернет магазину з продажі техніки.

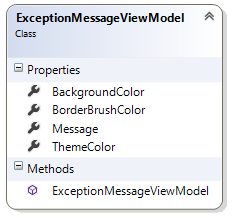
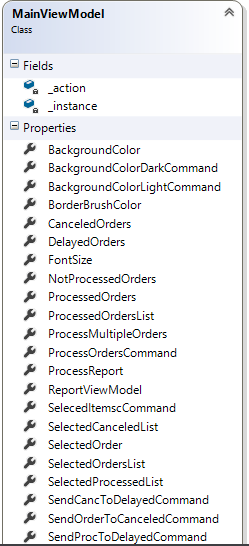


Рис. 2.4.14. UML діаграма класу ExceptionMessageViewModel інформаційної системи інтернет магазину з продажі техніки.



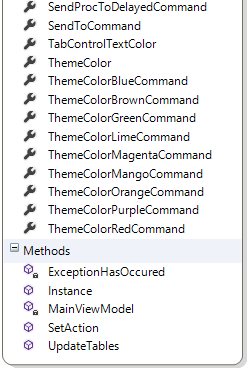


Рис. 2.4.15. UML діаграма класу MainViewModel інформаційної системи інтернет магазину з продажі техніки.

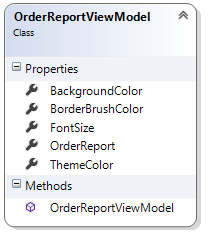


Рис. 2.4.16. UML діаграма класу OrderReportViewModel інформаційної системи інтернет магазину з продажі техніки.

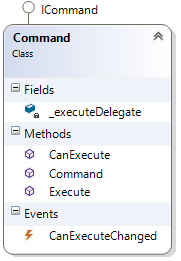


Рис. 2.4.17. UML діаграма класу Command інформаційної системи інтернет магазину з продажі техніки.

### 2.4.5 Класи інтерфейсу та додаткові класи програми

Класи Конвертори:

* ProcessedMessageConverter є класом конвертером, кортий конвертує дані про те чи замовлення оброблене чи ні, заодно передаючи повідомлення у відповідній локалізації. UML діаграма класу зображена на рис. 2.4.18, а код класу наведений в додатку Й.
* StoreConverter – є класом конвертером, котрий конвертує назву складу у більш просту і зручну для використання. UML діаграма класу зображена на рис. 2.4.19, а код класу наведений в додатку К.

Інші класи:

* CustomDataGrid є модифікацією стандартного елементу інтерфейсу Data Grid. Він містить покращення даного класу, які дозволяють прив’язувати до нього списки за допомогою мови розмітки XAML. UML діаграма класу зображена на рис. 2.4.20, а код класу наведений в додатку Л.
* LocalizationProvider є класом провайдером, котрий надає інформацію залежно від вибраної локалізації та ключового слова. UML діаграма класу зображена на рис. 2.4.21, а код класу наведений в додатку М.
* WindowsCreator є класом, котрий виконує роль створення вікон. UML діаграма класу зображена на рис. 2.4.22, а код класу наведений в додатку Н.