Держаний торговельно-економічний університет

Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

НА ТЕМУ:

«Програмний модуль підприємства з продажу автомобілів з пробігом»

Студента 4 курсу, 7 групи, спеціальності 121 «Інженерія програмного забезпечення» освітньої програми «Інженерія програмного забезпечення»	_____ підпис студента	Глижін Владислав Сергійович
Науковий керівник кандидат технічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки	_____ підпис керівника	Власенко Лідія Олександрівна
Гарант освітньої програми кандидат технічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки	_____ підпис гаранта	Рзаєва Світлана Леонідівна

КИЇВ – 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення та кібербезпеки

Освітній ступінь бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«14» листопада 2022 р.

Завдання

на випускний кваліфікаційний проєкт студентові

Глижіну Владиславу Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Програмний модуль підприємства з продажу автомобілів з пробігом»

Затверджена наказом ректора від «6» грудня 2022 р. №3288

2.Строк здачі студентом закінченого проєкту 5 червня 2023

3. Цільова установка та вихідні дані до проєкту

Метою випускної кваліфікаційної роботи є розробка швидкого, інтуїтивного та легкого у роботі програмного модулю для менеджерів компаній що займаються торгівлею авто з пробігом.

Об’єктом дослідження є процес створення модулю реалізації вторинного автопарку.

Предметом дослідження є програмний модуль повного циклу реалізації вживаних автомобілів розроблений на мові C# для операційної системи Windows.

4. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення
1.2 Аналіз вимог до програмного забезпечення
1.3 Феномен імпорту вживаних авто в Україну
1.4 Аналіз існуючих варіативних рішень
1.5 Вибір середовища та інструментів розробки
1.6 Технічне завдання
1.7 Висновок до Розділу 1

РОЗДІЛ 2 РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ

2.1 Розробка архітектури програми
2.2 Елемент Grid
2.3 Елемент Data template
2.4 Елемент Button
2.5 Події в XAML
2.6 Visibility в Windows Presentation Foundation
2.7 Model-View-ViewModel
2.8 Опис бази даних
2.9 Логічна модель бази даних
2.10 Фізична модель бази даних
2.11 Нормалізація бази даних
2.12 Висновок до Розділу 2

РОЗДІЛ 3 ТЕСТУВАННЯ ТА ОПИС ПРОГРАМНОГО МОДУЛЮ

3.1 Опис процесів тестування
3.2 Інструкція користувача по роботі з системою
3.3 Висновок до Розділу 3

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

5. Календарний план виконання проєкту

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	Вибір теми випускного кваліфікаційного проєкту	21.09.2022	21.09.2022
2.	Розробка та затвердження завдання на проєкт	14.11.2022	14.11.2022
3.	Вступ та перелік літературних джерел	23.12.2022	23.12.2022
4.	*Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ*	27.01.2023	27.01.2023
5.	*Розділ 2. РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ*	03.03.2023	03.03.2023
6.	*Розділ 3. ТЕСТУВАННЯ ТА ОПИС ПРОГРАМНОГО МОДУЛЮ*	14.04.2023	14.04.2023
7.	Висновки	28.04.2023	28.04.2023
8.	Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)	17.05.2023	17.05.2023
9.	Підготовка автореферату та презентації доповіді	26.05.2023	26.05.2023
10.	Попередній захист випускного кваліфікаційного проєкту	29.05.2023 – 02.06.2023	
11.	Зовнішнє рецензування випускного кваліфікаційного проєкту	05.06.2023	05.06.2023
12.	Здача прошитого випускного кваліфікаційного проєкту на кафедру	05.06.2023	05.06.2023
13.	Публічний захист випускного кваліфікаційного проєкту	06.06.2023 19.06-22.06.2023	

6. Дата видачі завдання «14» листопада 2022р.

7. Науковий керівник випускного кваліфікаційного проєкту Власенко Л.О.
(прізвище, ініціали, підпис)

8. Гарант освітньої програми Рзаєва С.Л.
(прізвище, ініціали, підпис)

9. Завдання прийняв до виконання студент Глижін В.С.
(прізвище, ініціали, підпис)

10. Відгук керівника випускного кваліфікаційного проєкту

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Науковий керівник випускного кваліфікаційного проєкту

_____

(підпис, дата)

Відмітка про попередній захист_____

(підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента _____Глижін В.С.___

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____Рзаєва С.Л.___

(прізвище, ініціали)

Завідувач кафедри_____Криворучко О.В.

(прізвище, ініціали)

«_____»_____20_____р.

АНОТАЦІЯ

Випускна кваліфікаційна робота на тему: «Програмний модуль підприємства з продажу автомобілів з пробігом».

Під час виконання випускної кваліфікаційної роботи були здобути теоретичні та практичні навички програмування мовою C#; був проведений аналіз предметної області, внаслідок чого розроблений програмний модуль задля продажу автомобілей. Програмне забезпечення складається з застосунку, що працює на платформі Windows, має інтуітивний дизайн та функціонально не перевантажений в порівнянні з подібними рішеннями.

Загальний обсяг роботи складає 46 сторінок, з них 3 розділів, 6 таблиць, 21 рисунок, список використаних джерел та додатки.

Ключові слова: мова програмування C#, прикладний програмний інтерфейс, автомобілі, аукціони, розмитнення, програмний модуль.

ABSTRACT

Graduation qualification work on the topic: "Software module of the enterprise for the sale of cars with mileage".

During the completion of the final qualification work, theoretical and practical programming skills in the C# language were acquired; an analysis of the subject area was carried out, as a result of which a software module was developed for the sale of cars. The software consists of an application running on the Windows platform, has an intuitive design and is not functionally overloaded compared to similar solutions.

The total volume of the work is 31 pages, including 5 chapters, 4 tables, 20 figures, a list of used sources and appendices.

Keywords: C# programming language, application programming interface, cars, auctions, customs clearance, software module.

ЗМІСТ

					ДТЕУ 121 07-8 БР		
Зм.	Аркуш	№ докум	Підпис	Дата			
Зав. каф		Криворучко О.В.		23.12.22	Програмний модуль підприємства з продажу автомобілів з пробігом	Стадія	Аркуш	Аркушів
Керівник		Власенко Л.О.		23.12.22		З	2	46
Гарант		Рзаєва С.Л.		23.12.22		Факультет інформаційних технологій 4 курс, 7 група		
Розробив		Глижін В.С.		23.12.22	Зміст			

ВСТУП

Актуальність: сьогодні нам доступні технології що дозволяють полегшувати наше повсякденне життя. Вони стали нашими інструментами праці, що допомагають вирішувати проблеми й виконувати задачі набагато швидше й ефективніше аніж без них. З ними, наше життя стає легшим, в деякому сенсі. Розвиток технологій неминуче набирає свої темпи, й це нормально, адже саме технології зіграли велику роль в тому, як люди живуть в сучасному світі. Вплив інновацій на нас зараз найвпливовіший, ніж ми могли будь-коли уявити.

Враховуючи важливість цих подій слід долучатися до розвиту технологій, адже вони є повсюди, у будь-яких сферах життя людини.

Завдання випускної кваліфікаційної роботи полягає в дослідженні області комерційних відносин людей у сфері продажу вживаних автомобілей. За останні роки, під впливом внесень правок у порядок розмитнення авто, ця сфера розвивається швидше ніж будь-коли за часи незалежної України.

Метою випускної кваліфікаційної роботи є розробка швидкого, інтуїтивного та легкого у роботі програмного модулю для менеджерів компаній що займаються торгівлею авто з пробігом.

Об’єктом дослідження є автомобільний ринок вживаних авто й, зокрема, обсяги ново розмитнених авто, що прибули в Україну за останні роки та їх вплив на ринок загалом.

Предметом дослідження є програмне забезпечення компаній зі сферою дій пов’язаних з пошуком, купівлею, перевезенням, розмитненням та інших послуг пов’язаних з вживаними автомобілями.

Під час виконання проекту створено буде модуль для перегляду авто, що можуть бути запропоновані для комерційної реалізації.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Структурне програмування та об’єктно-орієнтоване програмування (ООП) – це дві важливі парадигми що сформували розвиток сучасних мов програмування.

Структурне програмування це парадигма програмування в основі котрої закладено уявлення про будову програм у ієрархічній структурі. У 1970-х – на початку 1980-х структурне програмування було домінуючим. Такі мови як Pascal, C та Fortran, що підтримували таку форму кодування, мали велику популярність. Однак, такий дизайн програм мав одну й ту саму проблему – було дуже важко використовувати один й той самий код в двох, навіть більш-менш схожих програмах, через це, майже для кожного додатку створювали своє унікальне рішення.

Протягом останніх двох десятиліть було розроблено й розповсюджено інший підхід програмування, де замість розподілу проблеми на завдання було зосереджено увагу на структура даних. Це прийшло з усвідомленням того, що функціональність системи має тенденцію змінюватись більше, ніж дані, на які вона діє. Даним та способам їх структурування надавалося першочергове значення, а код був організований у модулях навколо них.

Поєднуючи структуру даних з їх внутрішніми функціональними можливостями, ці модулі стали «самодостатніми» сутностями, й отримали змогу бути підключеними до нових програм, тим самим вдосконалюючи багаторазове використання та скорочення часу на кодування. Крім того, структуруючи програму як окремі модулі, що відповідають за свої власті задачі, було покращено «читаємість» коду і його підтримка в майбутньому. Ці модулі називалися об'єктами і підхід програмування назвали об'єктно-орієнтованим програмуванням, часто скорочуваним як ООП. [9]

1.2 Аналіз вимог до програмного забезпечення

Мови програмування, що мають статус об'єктно-орієнтованих, можуть мати й підтримувати такі особливості програмування як: інкапсуляція, поліморфізм та успадкування. Ці правила роблять код простішим для повторного використання та його розширення, мінімізуючи ймовірність появи помилок. Щоб зрозуміти як ці функції працюють буде корисно розділити програмістів на дві групи. Перша з груп займається забезпеченням фундаментальних підпрограм або бібліотек які утворюють будівельні блоки більших застосувань. Тоді другою групою програмістів є ті, хто використовує ці фундаментальні бібліотеки у своїх додатках.

Інкапсуляція це – механізм що відноситься до приховування інформації, до якої Творець об'єкту не хоче, щоб користувач мав доступ. Користувач може отримати доступ лише до тих методів й даних, до котрих може звернутися за допомогою викликів функцій, наданих творцем. Це має ряд переваг. Користувач бачить набагато чистіший інтерфейс, за допомогою якого може використовувати об'єкти, а не турбуватися про непотрібні деталі реалізації. Список функції, надані творцем для доступу до об'єкта називається Інтерфейсом програмування програм (API). Більш важливою перевагою є те, що якщо Творець об'єктного модуля хоче змінити дані

модуля, це можна зробити не турбуючись про поломки коду, який використовує об’єкт. Творець повинен забезпечити лише те, що функції в API все ще працюють коректно. [10]

Успадкування - це спосіб прийняття загального роду об’єкта і використання його відповідно до конкретних потреб. Наприклад кожне парне число є числом, тому воно має властивості числа та, власне парного числа. Натомість поруч може існувати непарне число, котре також може мати властивості числа, однак вже властивості парного числа воно мати вже не зможе. У термінології ООП, «число» буде називатися базовим класом, а парне і непарне числа – похідними класами. За допомоги успадкування можна створювати ієрархії даних.

З успадкуванням тісно пов’язане поняття поліморфізму. Це одне з найбільш вагомих переваг об’єктно-орієнтованого програмування. Якщо буде потрібно надіслати повідомлення об’єкту, він повинен мати у своєму розпорядженні метод, створений для відповіді на це повідомлення. В ієрархії успадкування всі підкласи успадковують від своїх суперкласів. Однак, оскільки кожен підклас є окремою сутністю, кожному з них може знадобитися надати окрему відповідь на одне і те саме повідомлення. [10]

Розглянемо приклад, в якому Circle, Square і Star успадковуються від Shape. Це ставлення часто називається відношенням «є екземпляром», оскільки коло – це форма, як і квадрат. Коли підклас успадковує від суперкласу, він отримує всі можливості, якими володіє цей суперклас. Таким чином, Circle, Square і Star є розширеннями Shape. На рис. 1.1 ім’я кожного з об’єктів представляє метод Draw для Circle, Star і Square відповідно. При проектуванні системи Shape дуже корисно було б стандартизувати те, як ми використовуємо різноманітні форми. Так ми могли б вирішити, що, якщо нам буде потрібно намалювати фігуру будь-якої форми, ми викличемо метод з ім’ям Draw. Якщо ми станемо

дотримуватися цього рішення щоразу, коли нам потрібно буде намалювати фігуру, то буде потрібно викликати тільки метод Draw, незалежно від того, якою вона буде форми. В цьому полягає фундаментальна концепція поліморфізму - на індивідуальний об’єкт, будь то Circle, Star або Square, покладається обов’язок з малювання фігури, яка йому відповідає. Це загальна концепція в багатьох сучасних додатках, наприклад призначених для малювання і обробки тексту [1].

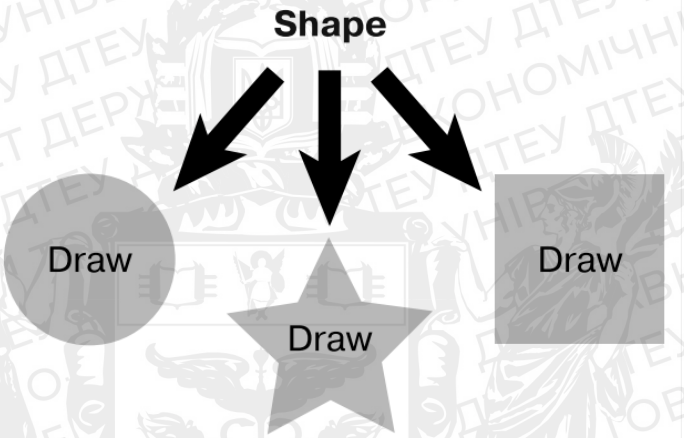

Рисунок 1.1 Приклад успадкування

Джерело: побудовано на основі джерела [1]

1.3 Феномен імпорту вживаних авто в Україну

В Україні починаючи з 2015 року стрімко стали набувати популярності вживані, злегка пошкоджені, однак досить вигідні до придбання, порівнюючи з ринком України, автомобілі з онлайн аукціонів Сполучених Штатів Америки (США) таких як Copart[3], Manheim[4], та IAA-Insurance[5].

Як правило, з США везуть автомобілі, що втрапили в дорожньо-транспортну пригоду (ДТП) або були пошкоджені після повеней чи інших катаклізмів. [8]

Справа в тому, що американці часто беруть автомобілі в лізинг або купують їх за допомогою кредиту. Й в випадках, коли з ними щось

трапляється ремонт автомобіля в США може бути занадто коштовним й не вигідним для страхових компаній. Саме в цих випадках транспортні засоби й виставляються на аукціонах де їх й купують іноземці.

Так, авто середнього класу і доброї комплектації можна придбати за $4-5 тис. USD, однак варто зауважити що це лише менша частина витрат на це авто. Аби отримати справний автомобіль на котрому можна пересуватись дорогами України, до вартості авто слід додати: витрати на послуги аукціону, транспортування автомобіля з аукціону до порту, а звідти вже до нашої неньки, далі в черзі йде розмитнення авто, його ремонт й, нарешті, сертифікація (перевірка чи справний автомобіль) й, фінальна, постанова на облік.

Зважаючи на весь цей шлях, котрий подолає бажане авто й час, який на це буде затрачено і ось відповідь на питання «Чому ж українці зважуються на цей шлях ?». Все досить просто – такі авто в сумі коштують дешевше автомобілей, що продаються на українському автомобільному ринку.

Приведемо трохи статистики:

Таблиця 2.1

СТАТИСТИКА ІМПОРТОВАНИЙ АВТО В УКРАЇНУ

Рік	Кількість автомобілей імпортованих в Україну (шт.)	Сума імпортованих автомобілей саме з США (Млн. USD)
2022	412966	343,62
2021	517450	572,65
2020	531400	664,88
2019	544043	518,50
2018	230050	316,76
2017	154410	229,29
2016	87768	–
2015	62313	–

Джерело: таблиця побудована автором, дані взяті з офіційного сайту митниці України [6] та джерела [7]

Ми бачимо що кількість автомобілей, що вперше отримують українські номерні знаки все росте, так само як і відсоток тих автомобілей, що припливли до нас саме з США.

Найбільш популярним ціновим сегментом є діапазон від 17 до 35 тисяч доларів. Частка його перевищила 50%. Серед дорогих брендів, які купували українці, - Aston Martin DBS, Bentley Bentayga, Continental GT і Mulsanne; Rolls-Royce Cullinan і Phantom, Mercedes S-Class Maybach і інші. Як ми бачимо з таблиці 2.1 й загальна вартість пригнаних авто за рік стабільно росте.

Як важливий факт, слід відзначити тенденцію зниження цін на вживані автомобілі. Очевидно, що це є наслідком збільшення кількості автомобілей на ринку. Адже лише за березень 2021р., в Україну було ввезено транспортних засобів на 420 млн. доларів. Що є більшою половиною від річної суми за цілий минулий рік рис. 1.2.

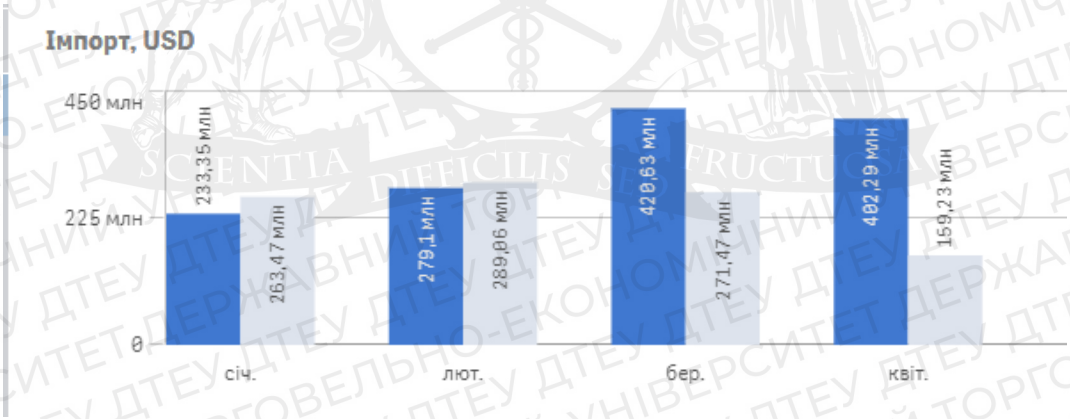

Рисунок 1.2 – Імпорт авто в Україну

Джерело: [6]

1.4 Аналіз існуючих варіативних рішень

Проаналізувавши сайти й соціальні мережі компаній www.auctionexport.com, atl.ua, auto_usa_lutsk та інших, що займаються підбором, пригоном та іншими роботами зв'язаними з автомобілями зі

Сполучених Штатів Америки, видно очевидну закономірність, всі вони користуються соціальними мережами й інколи створюють власні WEB-сайти, про додатки для операційної системи Windows й зовсім згадок немає. Це зумовлено тим, що до кожного клієнта, котрий зацікавлений у придбанні собі чи на перепродаж вигідного авто з США прикріплюється менеджер, що й виконує пошук цікавих автомобілей за характеристиками що надав йому клієнт.

Таким чином програмний модуль вирішує це завдання для компаній такої сфери послуг. Адже орієнтовні клієнти, встановивши програмний модуль, відкидають на перших етапах підбору й «прицінки» авто спілкування з менеджерами. Вони мають змогу продивитися варіанти автомобілей, що знаходяться на аукціонах, підібрати й продумати яке саме авто хочуть саме вони, а не те, що їм будуть нав'язувати менеджери.

За останнє десятиліття люди все більше відходять від потреби спілкування одне з одним: створюються каси самообслуговування, в великих компаніях створюються спеціалізовані робочі додатки де люди можуть створювати завдання, залишати коментарі й спілкуватися способом передачі повідомлень, таким чином робітник нічого не проґавить й буде бачити свою чітко поставлене завдання.

Так само й клієнти можуть сформулювати свій власний, чіткий запит на автомобіль, а потім, вже спілкуючись з професіоналами, вони допоможуть підібрати найвигідніший лот.

Другим типом людей, котрим буде корисне це ПЗ, це ті, хто вже визначився з маркою, моделлю, бажаним ступенем пошкоджень автомобілей й лише чекають коли ідеальне, саме для них, авто потратить на аукціон. В цьому кейсі людина має швидку змогу до моніторингу нових надходжень бажаних авто, й при наявності такого, буде дано завдання менеджеру до купівлі або ж змагання на аукціоні за цей лот.

Говорячи про спільні функціональні особливості розробленого модулю в порівнянні з браузерними аналогами є розподілення доступних можливостей зображених на рис. 2.3.

Рисунок 1.3 – UML діаграма прецедентів

Джерело: побудовано автором

1.5 Вибір середовища та інструментів розробки

Мова програмування C# є однією з найпопулярніших мов у наших сьогоднішніх реаліях. Згідно з опитуванням WEB-порталу DOU.ua, C# займає другу позицію за популярністю серед усіх комерційних проектів (рис

2.4), а за використанням серед комерційних проектів націлених на розробку ПЗ під персональні комп'ютери (ПК) посідає перше місце (рис. 2.5).

Рисунок 1.4 – Популярність мов програмування

Джерело: [2]

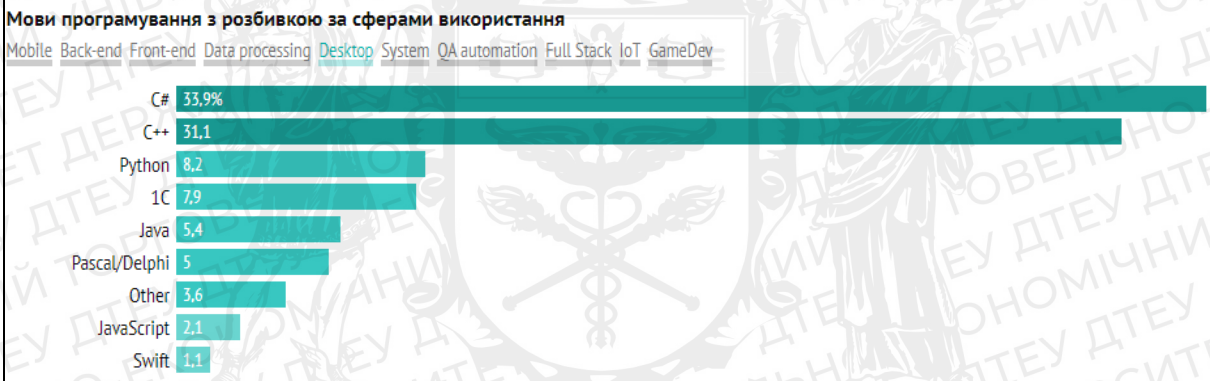

Рисунок 1.5 – Популярність мов програмування серед комерційних проектів під ПК

Джерело: [2]

Друге, що використовувалось при розробці ПЗ це Windows Presentation Foundation (WPF) - структура для створення додатків Windows, що дозволяє користувачам розробляти розширені користувальницькі інтерфейси, які мають 3D-анімацію та насичені кольори з меншою складністю коду. WPF це векторний механізм візуалізації, який використовує апаратне прискорення сучасних відеокарт, що робить інтерфейс користувача швидшим та масштабованим. В неї входить й

функціональність, котрій би заздрив попередник цієї технології – WinForms що надає безліч потужних функціональних можливостей які можна використовувати для спрощення та пришвидшення розробки додатків й налагодження їх. Від спільного використання ресурсів у програмі до створення власних властивостей, що можна використовувати в анімаціях, все це про Windows Presentation Foundation.

Третя ж опора створення даного програмного модулю це інтегроване середовище розробки (IDE) Visual Studio 2022. Була вибрана саме це програмне забезпечення адже: воно швидко працює з МП C# й відкладка додатків виконується набагато швидше, порівняно з іншими IDE, розробка ПЗ пришвидшується за допомоги функціоналу, інтерактивного інтерфейсу й, звісно, обов'язкове до згадування хост-додаток «Resharper» що вказує на помилки, підказує найбільш ймовірні рішення проблем й тд.

1.6 Технічне завдання

Типова структура технічного завдання:

1. Загальні відомості:

1.1. Найменування системи: Програмний модуль підприємства з продажу автомобілів з пробігом.

1.2. Планові терміни початку та закінчення робіт: початок робіт планується на 28.01.2023, а реліз проєкту на 28.05.2023

1.3. Порядок оформлення і пред'явлення результатів робіт: вказано в календарному плані виконання проєкту

1.4. Головний бенефіціар та потенційні користувачі системи є компанії, що надають послуги з підбору та привезення авто з-за кордону «під ключ», компанії, що вже мають певну кількість авто для продажу в ролі перших, та потенційні покупці вживаних авто в якості других.

2. Мета та призначення створення системи

2.1. Призначення системи: як призначення програмного продукту обрано спрощення, а як наслідок й пришвидшення роботи менеджерів з продажу автомобілей шляхом оптимізації процесів роботи з базою даних пропозицій, що наразі є в наявності

2.2. Метою створення системи є розробка швидкого, інтуїтивно зрозумілого у роботі програмного модулю для менеджерів компаній, що займаються торгівлею авто з пробігом.

3. Вимоги до системи:

3.1. Вимоги до системи в цілому:

3.1.1. Вимоги до структури та функціонування системи, перелік підсистем: програмний модуль має включати в себе графічний інтерфейс, базу даних.

3.1.1. Вимоги до режимів функціонування системи: система має працювати без підключення до мережі інтернет.

3.1.2. Вимоги до діагностування системи: Дані, що були введені та записані у вигляді торгових пропозицій, мають зберігатися в базі даних, й коректно відображатися.

3.1.3. Вимоги до режимів управління системою: буде розроблено два види доступу до системи: ступінь доступу «адміністратор» та «користувач».

4. Вимоги до програмного забезпечення: програмний модуль має стабільно працювати, на комп’ютерах з операційною системою Windows в незалежності від сценаріїв його використання.

5. Вимоги до технічного забезпечення: програмний модуль має працювати на персональних комп’ютерах, що оснащені: процесором з ARM64 або x64 архітектурою та двома або більше ядрами, мінімум 4GB оперативної пам’яті, відеокартою підтримуючою від WXGA (1366px на 768px) роздільної здатності.

6. Вимоги до методичного забезпечення: має містити інструкцію користувача по роботі з системою.

1.7 Висновок до Розділу 1

Таким чином було сформульоване чітке завдання, до розробки програмного забезпечення. Досить корисним у цій частині роботи було спілкування з представником області котру було досліджено. Адже люди з зовні не зможе осягнути й передбачити проблеми й засади на котрі натикається користувач. Таким чином ділова бесіда з потенційним юзером й адміністратором додатку що слід розробити, показала внутрішні нововведення що слід додати до програми.

Другою не менш важливою частиною цього розділу було саме аналіз спектру програмних забезпечень що слід вибрати саме для цього програмного забезпечення. Як факт було використано такі програмні додатки та імплементовані бібліотеки такі як: середа розробки Visual Studio 2022, мова програмування C# та система для побудови клієнтських додатків Windows Presentation Foundation.

Можна з впевненістю сказати, що об’єкт був рушійною силою в індустрії програмування протягом дуже довгого часу і залишатиметься таким в осяжному майбутньому. Докази, що підтверджують це твердження, досить переконливі. Сьогодні практично всі основні методології розробки програмного забезпечення засновані на об’єктах. В результаті практично всі мови програмування, мови сценаріїв і проекти додатків є об’єктно-орієнтованими. А в сучасних IT-проектах об’єктно-орієнтоване програмування й його механізми посідають ключові позиції. У підтвердження цього можна привезти статистику найпопулярніших мов програмування, див. рисунок 1.2. Опитування було проведене на WEB-

порталі DOU.ua [2] серед програмістів України й країн що входили в зону Співдружності Незалежних Держав в цілому.

Можна одразу побачити що всі вагомі й найпоширеніші мови програмування саме об’єктно орієнтовані, що й доводить потужності цієї парадигми програмування.

РОЗДІЛ 2
РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ

2.1 Розробка архітектури програми

Після проведення аналізу області, формулювання чіткого завдання й списку функціоналу, проведення дослідження основних принципів об’єктно-орієнтованого програмування, та перед початком роботи над самою реалізацією додатку, була розроблена діаграма класів мовою моделювання Unified Modeling Language (UML) рис 3.1. UML – це сучасний підхід до моделювання та документування програмного забезпечення. Він базується на схематичному зображенні програмних компонентів. Використовуючи візуальне подання можна краще зрозуміти можливості, недоліки або помилки програмного забезпечення чи бізнес процесів, ще на стадії його моделювання. Саме тому, в командах з розробки програмного забезпечення все частіше з’являються UML діаграми різних типів.

Рисунок 2.1 UML діаграма розгортання архітектурної структури програмного модулю

Джерело: побудовано автором

					ДТЕУ 121 07-08 БР		
Зм.	Аркуш	№ докум	Підпис	Дата			
Зав. каф		Криворучко О.В.		03.03.23	Програмний модуль підприємства з продажу автомобілів з пробігом	Стадія	Аркуш	Аркушів
Керівник		Власенко Л.О.		03.03.23		Р2	19	46
Гарант		Рзаєва С.Л.		03.03.23		Факультет інформаційних технологій 4 курс, 7 група		
Розробив		Глижін В.С.		03.03.23	Розробка програмного модулю			

2.2 Елемент Grid

При розробці програмного модулю повсюди використовувався елемент керування Grid, Це досить потужний та вельми корисний макет у WPF. Він дозволяє розташовувати дочірні елементи в клітинках, визначених рядками та стовпцями. Якщо уявити програму без картинок, тексту та кнопок – від неї залишиться лише сітки, що й створюються за допомогою Grid. Насправді, коли ми додаємо новий документ XAML або створюємо новий проект WPF у Visual Studio, IDE автоматично додає сітку як перший контейнер всередині елемента вікна.

Ми можемо створити рядки та стовпці двома наступними способами. Перший спосіб: набравши код XAML.

За замовчуванням сітка має один рядок і один стовпець. Ми можемо додати більше рядків і стовпців, додавши елемент RowDefinition для кожного рядка всередині властивості Grid.RowDefinitions та елемент ColumnDefinition для кожного стовпця всередині властивості Grid.ColumnDefinitions. За замовчуванням GridLines невидимі. Для відображення GridLines слід встановити для властивості ShowGridLines Grid значення True. GridLines корисні під час налагодження для визначення того, який елемент знаходиться в якій комірці.

Нижче приведено простий приклад роботи цього елементу (рис 3.2 та рис. 3.3). У цьому прикладі створено 3 рядки та 3 стовпці. Потім додано 9 TextBlocks і збережено положення TextBlock всередині сітки, вказавши значення Grid.Row та Grid.Column. Якщо не вказати властивість Grid.Row і Grid.Column, тоді Елемент поміщається в Grid.Row = "0" і Grid.Column = "0", іншими словами в Перший рядок і Перший стовпець.

```
<Window x:Class="example_for_kursovaya.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:example_for_kursovaya"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid ShowGridLines="True">
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <TextBlock Text="Row0 Column0" Grid.Row="0" Grid.Column="0" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>
        <TextBlock Text="Row0 Column1" Grid.Row="0" Grid.Column="1" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>
        <TextBlock Text="Row0 Column2" Grid.Row="0" Grid.Column="2" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>
        <TextBlock Text="Row1 Column0" Grid.Row="1" Grid.Column="0" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>
        <TextBlock Text="Row1 Column1" Grid.Row="1" Grid.Column="1" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>
        <TextBlock Text="Row1 Column2" Grid.Row="1" Grid.Column="2" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>
        <TextBlock Text="Row2 Column0" Grid.Row="2" Grid.Column="0" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>
        <TextBlock Text="Row2 Column1" Grid.Row="2" Grid.Column="1" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>
        <TextBlock Text="Row2 Column2" Grid.Row="2" Grid.Column="2" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>
    </Grid>
</Window>
```

Рисунок 2.2 Фрагмент коду елементу Grid

Джерело: побудовано автором (IDE Microsoft Visual Studio 2022)

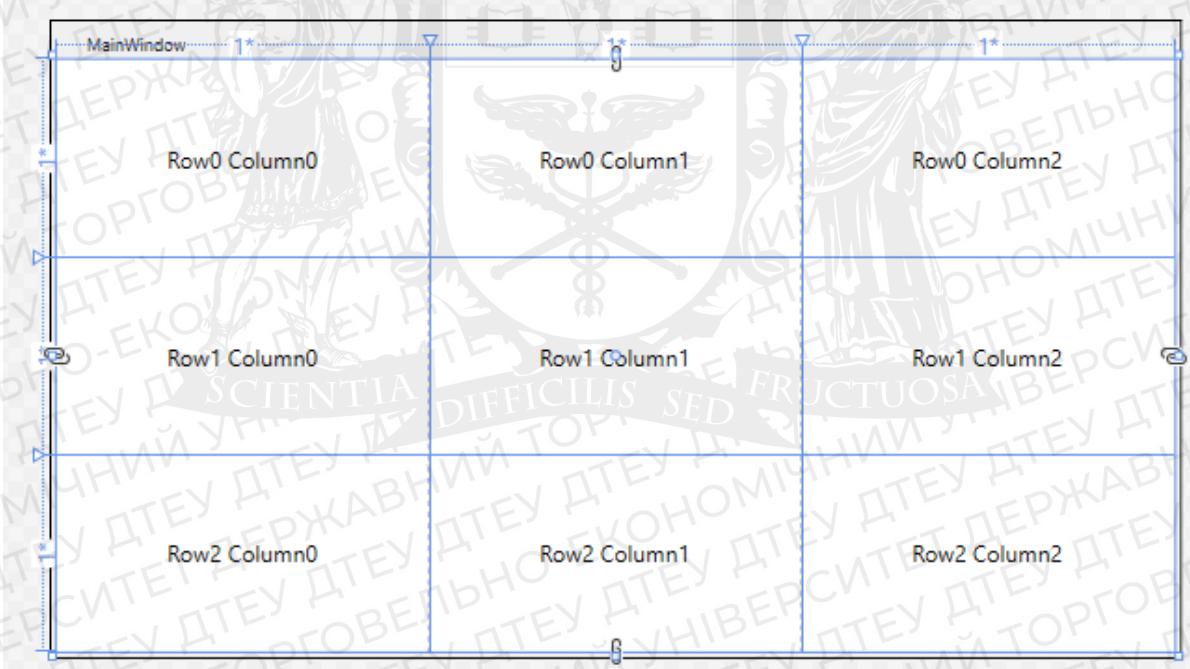

Рисунок 2.3 Фрагмент вікна з елементом Grid

Джерело: побудовано автором (IDE Microsoft Visual Studio 2022)

2.3 Елемент Data template

Data template це шаблон даних - це частина XAML, що описує, як повинні відображатися дані. Шаблон даних може містити елементи, кожен

з яких прив’язаний до властивості даних, а також додаткову розмітку, що описує макет, колір та інші властивості. DataTemplate, в основному, використовується для роботи з зовнішнім виглядом даних, що відображаються елементом управління, а не зовнішнього вигляду самого елемента керування.

Отже, DataTemplate можна застосувати до ContentControls або ItemsControl. Ім’я властивості, якій можна призначити DataTemplate, залежить від того, керуючий елемент є елементом керування вмістом або ItemsControl.

Елементи управління, які можуть містити один елемент (єдиний логічний дочірній тип об’єкта), називаються «елементами керування вмістом». Ці елементи управління походять від базового класу ContentControl. Елементи керування, які можуть містити колекцію елементів (багато логічних дочірніх елементів типу Object), називаються «елементами керування».

2.4 Елемент Button

Жодна програма не може обійтись без елементів управління й основним серед них є кнопка.

Багато елементів керування дозволяють розміщувати вміст між початковою та кінцевою тегами, а потім - вмістом елемента управління. Наприклад, елемент керування кнопкою дозволяє вказати текст, що відображається на ньому між початковим і кінцевим тегами Рис.3.4.

```
<Button>A button</Button>
```

Рисунок 2.4 – Елемент кнопки

Джерело: побудовано автором (IDE Microsoft Visual Studio 2022)

HTML не чутливий до регістру, але XAML це, оскільки ім'я елемента управління повинно відповідати типу в .NET framework. Те саме стосується імен атрибутів, що відповідає властивостям елемента керування. Ось кнопка, де ми визначаємо пару властивостей, додаючи атрибути до тегу Рис. 3.5.

```
<Button FontWeight="Bold" Content="A button" />
```

Рисунок 2.5 – Призначення властивостей елементу кнопки

Джерело: побудовано автором (IDE Microsoft Visual Studio 2022)

Ми встановлюємо властивість FontWeight, надаючи нам жирний текст, а потім встановлюємо властивість Content, яке є таким самим, як написання тексту між початковим і кінцевим тегом. Однак усі атрибути елемента керування також можуть бути визначені таким чином, де вони відображаються як дочірні теги основного елемента керування, використовуючи нотацію Control-Dot-Property Рис. 3.6.

```
<Button>
    <Button.FontWeight>Bold</Button.FontWeight>
    <Button.Content>A button</Button.Content>
</Button>
```

Рисунок 2.6 – Альтернативний спосіб запису атрибутів

Джерело: побудовано автором (IDE Microsoft Visual Studio 2022)

Результат точно такий же, як і вище, тому в даному випадку мова йде про синтаксис і ні про що інше. Однак багато елементів керування дозволяють вміст, відмінного від тексту, наприклад, інших елементів керування. Ось приклад, коли на одній кнопці ми маємо текст у різних кольорах за допомогою двох елементів керування TextBlock всередині кнопки Рис. 3.7.

```
<Button>
    <Button.FontWeight>Bold</Button.FontWeight>
    <Button.Content>
        <WrapPanel>
            <TextBlock Foreground="Blue">Multi</TextBlock>
            <TextBlock Foreground="Red">Color</TextBlock>
            <TextBlock>Button</TextBlock>
        </WrapPanel>
    </Button.Content>
</Button>
```

Рисунок 2.7 – Індивідуальні властивості елементів

Джерело: побудовано автором (IDE Microsoft Visual Studio 2022)

2.5 Події в XAML

Більшість сучасних платформ інтерфейсу керуються подіями, як і WPF. Усі елементи керування, включаючи вікно (яке також успадковує клас Control), містять низку подій, на які можна відслідковувати. Можна відслідкувати ці події, але це означає, що програма отримає сповіщення, коли подія відбудуться, і можна буде відреагувати на це.

Існує безліч типів подій, але деякі найчастіше використовуються саме для реагування на взаємодію користувача з додатком за допомогою миші або клавіатури. Для більшості елементів керування допоможуть такі події, як KeyDown, KeyUp, MouseDown, MouseEnter, MouseLeave, MouseUp, Click та кілька інших.

На рисунку 3.8 можна побачити ті елементи управління, про котрі йшлось в минулих розділах й подія Click. В даному випадку при натисканні мишею на кнопку DealMenuButton буде спрацьовувати метод, що знаходиться в класі MainWindow.xalm.cs, однак цей метод можна застосовувати не тільки в тандемі з кнопками, а й, наприклад, з зображенням. Роблячи з останнього елемент тригеру. Таким способом в ПЗ реалізовано вхід в користувацьке меню та відкриття бокового меню.

```
<Button Name="DealMenuButton" Click="DealMenuButton_Click" Margin="0, 10, 0, 0">
    <WrapPanel>

        <TextBlock Text="{Binding Name}" FontWeight="Bold"/>
        <TextBlock Text=", Пробіг: "/>
        <TextBlock Text="{Binding Odometet}"/>
        <TextBlock Text="миль, Ціна: "/>
        <TextBlock Text="{Binding Price}"/>
        <TextBlock Text="$ "/>

    </WrapPanel>
</Button>
```

Рисунок 2.8 Використання метода Click

Джерело: побудовано автором (IDE Microsoft Visual Studio 2022)

2.6 Visibility в Windows Presentation Foundation

Для початку слід сказати що це за параметр Visibility, та що за допомогою нього можна робити. Visibility це параметр видимості або ж активності. Це практично основа видимої частини й подій програмного модулю. Об'єднавши такі речі як Grid, події та Visiability, було досягнуто чіткість й ефективність роботи всіх POP-UP меню в програмі.

В WPF UIElement.Visibility має 3 стани: Visible, Hidden та Collapsed. Перший стан керування – робить елемент видимим, другий – не видимим, однак все ще може впливати на макет, й третій – не видимим й не активним. На рисунку 3.9 показано, приклад події після натискання на кнопку деавторизації користувача (всі кнопки й меню, що активні авторизованим користувачам отримують статус не активних).

```
private void UnLog_Click(object sender, RoutedEventArgs e)
{
    UnLogUser.Visibility = Visibility.Visible;
    LogUser.Visibility = Visibility.Collapsed;
    ToCreateButton.Visibility = Visibility.Collapsed;
    DeleteThisDeal.Visibility = Visibility.Collapsed;
    BuyIt.Visibility = Visibility.Collapsed;
}
```

Рисунок 2.9 – Деактивізація елементів

Джерело: побудовано автором (IDE Microsoft Visual Studio 2022)

2.7Model-View-ViewModel

Шаблон Model-View-ViewModel (MVVM) є популярним підходом до розробки програмного забезпечення, особливо для розробки інтерфейсу користувача. Цей шаблон має декілька переваг, включаючи:

Розділення відповідальності: MVVM дозволяє розділити відповідальність між різними компонентами додатку. Модель відповідає за логіку бізнес-логіки додатку, Представлення відображає інформацію користувачу, а ViewModel діє як посередник між двома компонентами. Це розділення забезпечує більшу модульність і зручність управління.

Тестування: MVVM сприяє більшій тестируваності додатку. Якщо відповідальність окремих компонентів розподілена належним чином, їх можна легко тестувати окремо. Це робить процес тестування більш простим і ефективним.

Розширення: MVVM забезпечує легкість розширення додатку. Якщо виникає потреба в додатковій функціональності, можна легко додати нову ViewModel та відповідний Представлення без зміни інших компонентів додатку.

Підтримка інтерактивності: MVVM дозволяє легко реалізувати інтерактивні елементи у додатку. ViewModel може додати команди та взаємодіювати з Представленням, що дозволяє користувачу взаємодіяти з додатком безпосередньо з Представленням.

Швидкість розробки: MVVM дозволяє знизити час розробки додатку. Розділення відповідальності між різними компонентами забезпечує зручний процес розробки та більш швидкий розвиток додатку.

2.8 Опис бази даних

База даних - це структурований набір даних, що організовано для збереження та доступу до них. Вона є основою більшості програмних

продуктів, що працюють зі збереженням, обробкою та аналізом великих обсягів даних. Бази даних дозволяють зберігати велику кількість інформації та швидко отримувати до неї доступ, забезпечуючи ефективне використання ресурсів та покращуючи роботу з даними.

Основною метою баз даних є забезпечення постійного доступу до інформації для більшості користувачів програмних продуктів. Це означає, що бази даних повинні бути забезпечені надійними методами зберігання даних та їх захисту від незаконного доступу. Крім того, бази даних дозволяють зберігати дані у структурованому вигляді, який дозволяє ефективно збирати, аналізувати та використовувати дані

Бази даних можуть бути використані в програмуванні для збереження різноманітної інформації, включаючи текст, зображення, звук та відео. Програмісти можуть використовувати бази даних для збереження даних, що використовуються в програмі, таких як імена користувачів, паролі, адреси електронної пошти та інші. Бази даних також можуть зберігати дані, що використовуються в інтернет-магазинах, соціальних мережах, фінансових системах та інших програмах.

Одним з основних типів баз даних є реляційні бази даних. Вони використовують таблиці для збереження даних та дозволяють виконувати запити до даних для отримання необхідної інформації. У реляційних базах

2.9 Логічна модель бази даних

Логічна модель бази даних - це концептуальне відображення даних та їх взаємозв'язків, яке використовується для створення реляційної бази даних. Вона дозволяє програмістам та архітекторам визначити, які дані будуть зберігатися та як вони будуть відноситися одне до одного, що допомагає відображати складні взаємозв'язки між даними.

Логічна модель бази даних містить таблиці, які відображають типи даних та їх взаємозв'язки. Вона дозволяє визначити ключові поля, які використовуються для зв'язку таблиць між собою, та поля, які використовуються для зберігання даних.

Логічна модель бази даних використовується для створення фізичної моделі бази даних. Фізична модель бази даних містить деталізовані відомості про структуру та збереження даних у конкретній базі даних. Це може включати визначення типів даних, індексів, розмірів таблиць, а також визначення фізичного місця збереження даних на диску.

Створення логічної моделі бази даних є важливим етапом у розробці бази даних, оскільки вона допомагає визначити структуру та зв'язки між даними. Логічна модель бази даних дозволяє програмістам та архітекторам побачити загальну структуру даних та відносини між ними, що допомагає визначити оптимальну структуру бази даних для конкретного проекту.

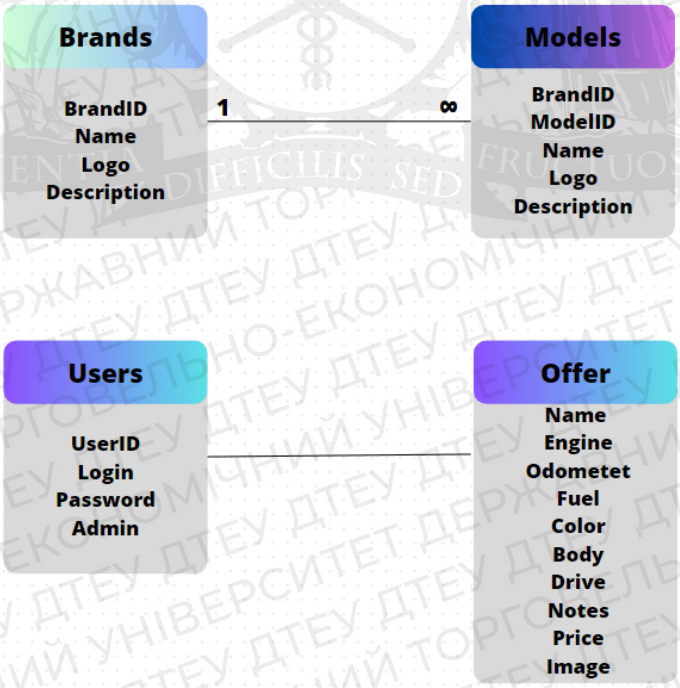

Рисунок 2.10. Логічна модель бази даних

Джерело: побудовано автором

2.10 Фізична модель бази даних

Фізична модель бази даних - це детальне відображення структури та зберігання даних у конкретній базі даних. Вона описує, як саме дані будуть зберігатися на диску та як вони будуть доступні для операцій з базою даних.

Фізична модель бази даних містить відомості про розмір таблиць, індексів, типів даних та їх зберігання, а також про фізичне розташування даних на диску. Вона також може включати інформацію про забезпечення безпеки даних, таку як авторизація доступу та резервне копіювання.

Фізична модель бази даних є важливим етапом у розробці бази даних, оскільки вона дозволяє програмістам та архітекторам визначити, як саме дані будуть зберігатися та як до них будуть звертатися. Вона допомагає оптимізувати продуктивність бази даних та підвищити її ефективність. Фізична модель бази даних також дозволяє створити резервні копії даних та відновити їх у разі потреби.

Створення фізичної моделі бази даних зазвичай відбувається після створення логічної моделі бази даних. Фізична модель бази даних відображає конкретні потреби та обмеження бази даних, що дозволяє забезпечити ефективну та надійну роботу бази даних.

Таблиця 2.11

ТАБЛИЦЯ USERS

Назва поля	Ключ	Індексоване поле	Тип даних	Null-значення
UserId	Primary key	Так	int	Не допустимі
UserName			nvarchar(256)	Не допустимі
Email			nvarchar(256)	Не допустимі
PasswordHash			nvarchar(MAX)	Не допустимі

Таблиця 2.12

ТАБЛИЦЯ BRANDS

Назва поля	Ключ	Індексоване поле	Тип даних	Null-значення
BrandId	Primary Key		int	Не допустимі
BrandName			nvarchar(256)	Не допустимі
Logo			image	Не допустимі
Description			nvarchar(256)	Не допустимі

Таблиця 2.13

ТАБЛИЦЯ MODELS

Назва поля	Ключ	Індексоване поле	Тип даних	Null-значення
BrandId	Foreign Key		int	Не допустимі
ModelId	Primary Key		int	Не допустимі
ModelName			nvarchar(256)	Не допустимі
Logo			image	Не допустимі
Description			nvarchar(256)	Не допустимі

2.11 Нормалізація бази даних

Нормалізація баз даних - це процес організації даних у базі даних, з метою забезпечення ефективної та надійної роботи з ними. Метою нормалізації є зменшення дублювання даних та забезпечення їх консистентності та інтегритету. Процес нормалізації включає кілька етапів.

Перший етап нормалізації баз даних - це перетворення нерозподіленої бази даних у першу нормальну форму (1NF). Для цього потрібно розбити таблиці бази даних на окремі поля та виключити багатовимірні значення. Крім того, кожен рядок таблиці повинен мати унікальний ідентифікатор.

Другий етап нормалізації - перетворення бази даних до другої нормальної форми (2NF). Цей етап вимагає, щоб кожне поле у таблиці залежало від всього первинного ключа, а не лише від частини його. Якщо у таблиці є поля, що залежать лише від частини первинного ключа, їх слід видалити та створити окрему таблицю.

Третій етап нормалізації - перетворення бази даних до третьої нормальної форми (3NF). У цій нормальній формі кожне поле таблиці залежить лише від первинного ключа, а не від інших полів у таблиці. Якщо у таблиці є поля, що залежать від інших полів, їх також слід видалити та створити окрему таблицю.

Четвертий етап нормалізації - перетворення бази даних до четвертої нормальної форми (4NF). У цій нормальній формі кожне поле таблиці залежить лише від первинного ключа та не залежить від інших полів у таблиці. Крім того, у цій нормальній формі уникнуто залежності між неключовими полями.

П'ятий етап нормалізації даних в програмуванні - це перетворення бази даних до п'ятої нормальної форми (5NF), також відомої як проектування баз даних з узагальненими залежностями.

У цій нормальній формі кожна залежність між атрибутами бази даних відображена як окрема таблиця. За допомогою узагальнених залежностей можна описати відношення між атрибутами в базі даних. Цей етап є досить складним та вимагає від програмістів пильного аналізу та проектування бази даних.

Перевагою п'ятої нормальної форми є те, що вона зменшує дублювання даних та забезпечує їх консистентність та інтегритет. Крім того, вона дозволяє більш ефективно управляти великими обсягами даних та забезпечує більш гнучкий та масштабований дизайн бази даних.

Однак, не завжди є необхідність переводити базу даних до п'ятої нормальної форми, особливо для менших проектів з невеликою кількістю даних. Кожен етап нормалізації баз даних повинен бути ретельно розглянутий та здійснений з урахуванням конкретних потреб проекту.

2.12 Висновок до Розділу 2

Розібравшись з усіма елементами, тегами, функціями та іншими помічниками що надаються при виборі мови програмування C#, середовища програмування Visual Studio 2022 й, особливо, Windows Presentation Foundation та шаблону Model-View-ViewModel, було розроблено програмне забезпечення з інтерактивною користувацькою структурою, широким функціоналом та доброю швидкістю роботи програми. Враховуючи те, що аналогів програмного забезпечення на момент роботи над випускною кваліфікаційною роботою немає – це єдиний в своєму роді продукт що є швидшим за сайти компаній що займаються діяльністю в дослідженій сфері, адже частина логіки й процесів роботи програми проводяться власне на персональному комп’ютері користувача. Програмний модуль підприємства з продажу автомобілів з пробігом став результатом пройденої роботи.

Також, в цьому розділі ми розглянули . основні переваги нормалізації баз даних, що включають зменшення дублювання даних, забезпечення консистентності та інтегритету, покращення ефективності управління даними та можливість створення більш гнучкого та масштабованого дизайну бази даних. Однак, необхідно виділити, що п'ята нормальна форма (5NF) може забезпечити ще більшу гнучкість та масштабованість бази

даних, вона не завжди є необхідною для всіх проектів. Для менших проектів з обмеженими обсягами даних може бути достатньо досягти третьої чи четвертої нормальної форми. Важливо врахувати конкретні потреби та обмеження проекту перед здійсненням кожного етапу нормалізації.

РОЗДІЛ 3
ТЕСТУВАННЯ ТА ОПИС ПРОГРАМНОГО МОДУЛЮ

3.1 Опис процесів тестування

Процес тестування не менш важливий за саму розробку додатку, адже без належних перевірок в ході експлуатації можуть виявитись як незначні помилки, так і взагалі незаплановане закриття додатку. Для запобігання таких проблем, програмний модуль підприємства з продажу автомобілів з пробігом було протестовано з застосуванням технік тест дизайну таких як: класи еквівалентності (Equivalence Partitioning), аналіз граничних значень (Boundary Value Analysis) та попарне тестування (Pairwise testing) на прикладі форми «Створення пропозиції» рис. 3.1 .

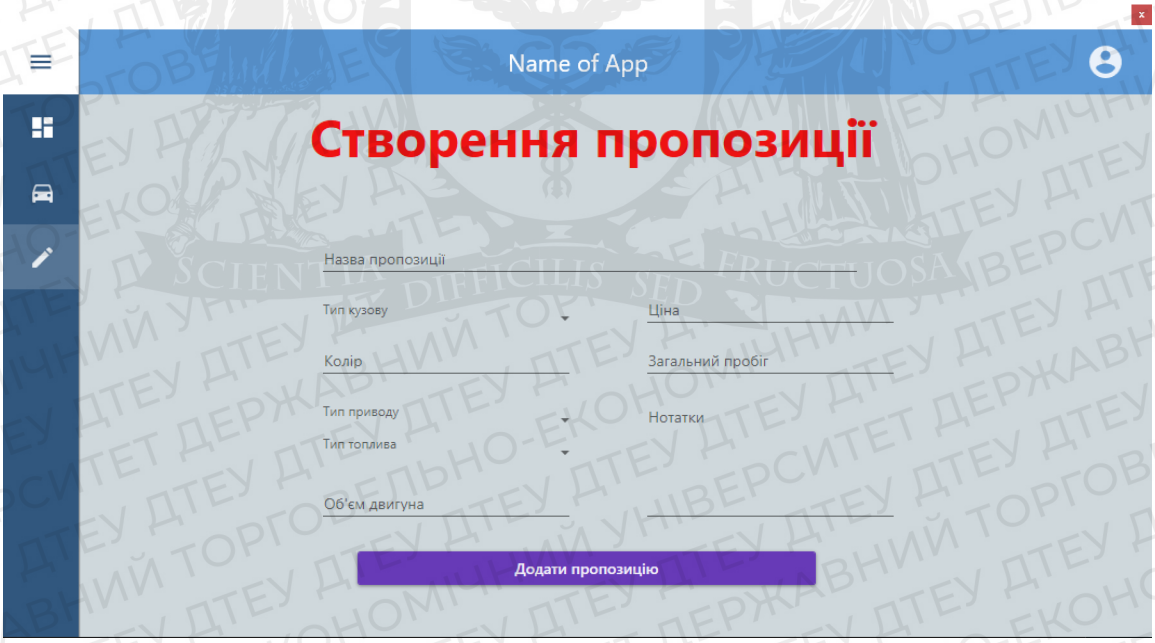

Рисунок 3.1. Форма тестування

Джерело: побудовано автором

					ДТЕУ 121 07-08 БР			
Зм.	Аркуш	№ докум	Підпис	Дата				
Зав. каф		Криворучко О.В.		14.04.23	Програмний модуль підприємства з продажу автомобілів з пробігом	Стадія	Аркуш	Аркушів
Керівник		Власенко Л.О.		14.04.23		Р3	34	46
Гарант		Рзаєва С.Л.		14.04.23		Факультет інформаційних технологій 4 курс, 7 група		
Розробив		Глижін В.С.		14.04.23	Тестування та опис програмного модулю			

Важливою частиною документації тестування є результуюча таблиця. Вона часто називається також звіт про тестування або звіт про результати тестування. Створюється для записів результатів тестування програмного продукту. Основні причини створення результуючої таблиці є наступними:

В основу результуючої таблиці закладено бути фундаментом документації про проведене тестування. Вона фіксує всі проведені тестові сценарії, входи, очікувані результати та фактичні результати. Це допомагає команді розробників, тестувальникам та менеджерам проекту зрозуміти, які тестові сценарії були пройдені, які проблеми виявлені та які кроки були вжиті для виправлення цих проблем.

Також важливим результатом є порівняння результатів: Результуюча таблиця дозволяє порівняти очікувані результати з фактичними. Це допомагає виявити розбіжності та невідповідності, що можуть бути наслідком неправильної реалізації функціональності або помилок в програмному продукті. Звіт також може включати додаткову інформацію про помилки, таку як кроки для відтворення проблеми або деталі про середовище тестування, що допомагає команді розробників зрозуміти проблему та виправити її.

Таблиця 3.2

РЕЗУЛЬТУЮЧА ТАБЛИЦЯ

Поле	**Positive/Negative**	**Значення**	**Коментар**
Назва пропозиції	Positive	Ввести назву авто	True
	Negative	Залишити пусте поле	False
Тип кузову	Positive	Обрати один з пункт з переліку	True
	Negative	Залишити пусте поле	False
Ціна	Positive	Ввести число	True
	Negative	Ввести символи літерами	False
	Negative	Залишити пусте поле	False

Продовження табл. 3.2

Колір	Positive	Ввести слово	True
	Negative	Ввести число	False
	Negative	Залишити пусте поле	False
Загальний пробіг	Positive	Ввести число	True
	Negative	Ввести від'ємне число	False
	Negative	Залишити пусте поле	False
Тип приводу	Positive	Обрати один з пункт з переліку	True
	Negative	Залишити пусте поле	False
Тип палива	Positive	Обрати один з пункт з переліку	True
	Negative	Залишити пусте поле	False
Об'єм двигун	Positive	Ввести число	True
	Negative	Ввести від'ємне число	False
	Negative	Залишити пусте поле	False
Нотатки	Positive	Ввести довільні символи	True
	Negative	Залишити пусте поле	False

Тест-кейси табл. 3.3 є важливим інструментом при тестуванні програмного забезпечення, адже тест-кейси допомагають виявити помилки та неправильну роботу модуля, адже навіть після розробки й успішних запусків програми, існує можливість того, що під час розробки сталася помилка або не було враховано певні сценарії використання. Тест-кейси допоможуть виявити такі проблеми та запобігти їх поширенню.

По-друге, якщо модуль був змінений або оновлений після програвання, тест-кейси дозволять виконати регресійне тестування, щоб переконатися, що внесені зміни не призвели до появи нових помилок або не порушили наявну функціональність. Це має назву регресійного тестування.

Таблиця 3.3.

ТАБЛИЦЯ ТЕСТОВИХ КЕЙСІВ

	Кроки	Очікуваний результат
Positive	1. Назва пропозиції: "BMW X5" 2. Тип кузова: SUV 3. Колір: "Чорний" 4. Тип приводу: Повний 5. Тип палива: Бензин 6. Об'єм двигуна: 3000 7. Ціна: 50000 8. Загальний пробіг: 10000 9. Нотатки: "Без додаткових нотаток"	Очікуваний результат: Форма успішно відправлена.
Negative	1. Назва пропозиції: "" 2. Тип кузова: Вибрати пункт "Не вибрано" 3. Колір: "Червоний" 4. Тип приводу: Вибрати пункт "Не вибрано" 5. Тип палива: Вибрати пункт "Не вибрано" 6. Об'єм двигуна: -2000 7. Ціна: 0 8. Загальний пробіг: -100 9. Нотатки: "Тестові нотатки"	Очікуваний результат: Відображення помилок валідації для полів: "Назва пропозиції", "Тип кузова", "Тип приводу", "Тип палива", "Об'єм двигуна", "Ціна" і "Загальний пробіг".
Positive	1. Назва пропозиції: "Toyota Corolla" 2. Тип кузова: Седан 3. Колір: "Сірий" 4. Тип приводу: Передній 5. Тип палива: Дизель 6. Об'єм двигуна: 2000 7. Ціна: 25000 8. Загальний пробіг: 50000 9. Нотатки: "Історія обслуговування повна"	Очікуваний результат: Форма успішно відправлена.

3.2. Інструкція користувача по роботі з системою

На початку роботи з програмним забезпеченням, користувача зустрічає вікно авторизації рис 3.2, що використовується для входу в систему та має такі поля:

Поле "Логін": Користувач вводить свій ідентифікатор або ім'я користувача.

Поле "Пароль": Користувач вводить свій пароль в це поле. При введенні це поле маскується для забезпечення безпеки.

Кнопка "Авторизуватись": Користувач натискає цю кнопку для підтвердження введених даних та авторизації в систему.

Кнопка "Реєстрація": Ця кнопка перенаправляє користувача на сторінку реєстрації, де він може створити новий обліковий запис.

Кнопка "Пропустити авторизацію": Ця кнопка дозволяє користувачу пропустити процес авторизації та продовжити використання деяких функцій з обмеженням доступу до системи.

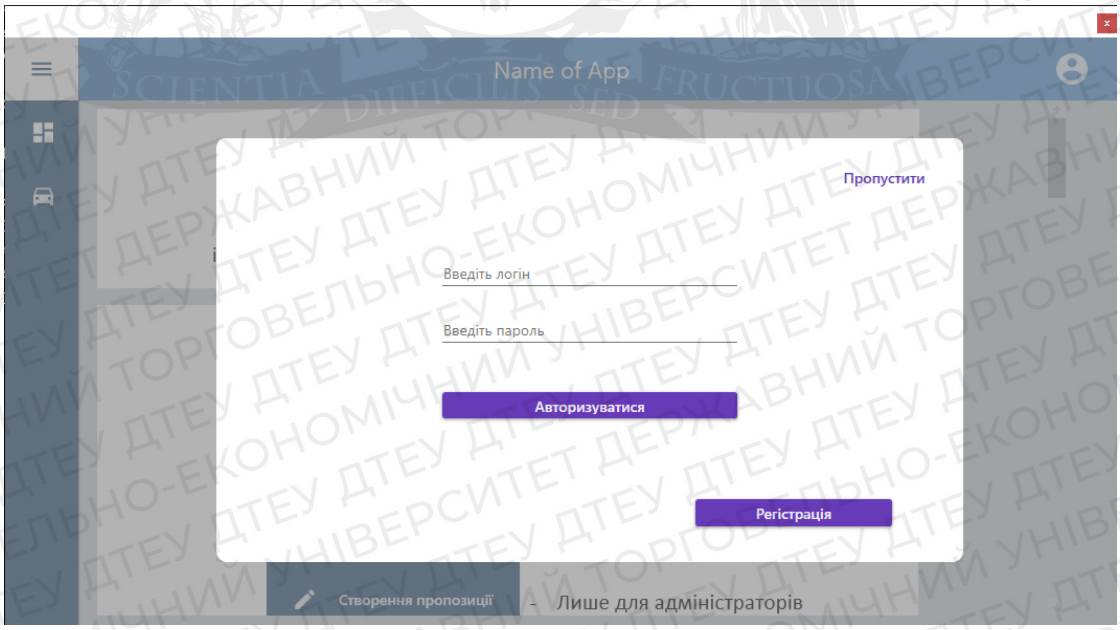

Рисунок 3.2. Вікно авторизації

Джерело: побудовано автором

Початковий екран рис 3.3 несе в собі опис початкового користувацького досвіду користувача та найактуальніші пропозиції.

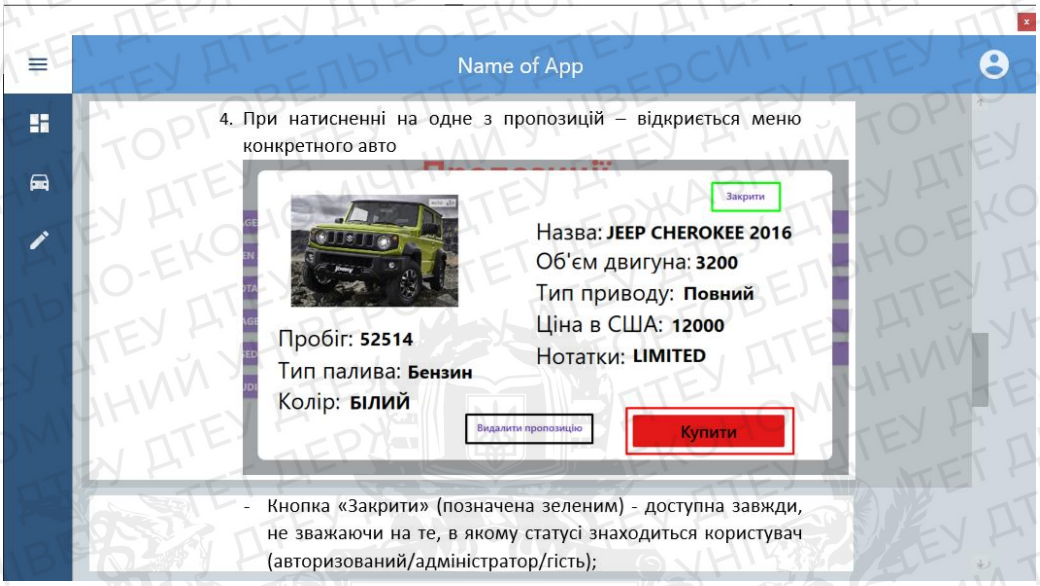

Рисунок 3.3. Початковий екран
Джерело: побудовано автором

Наступний елемент, про який слід розповісти це основне навігаційне, меню додатку рис 3.4. Відкривається воно ліворуч за кнопкою «Меню». Відкриття супруводжується плавною анімацією й має наступні пункти до вибору:

"На головну": цей пункт меню перенаправляє користувача на головну сторінку додатку. Він надає швидкий доступ до основного вмісту або інформації, яка є центральною для додатку.

"Перегляд пропозицій": цей пункт меню відображає список доступних пропозицій.

"Створення пропозиції" (доступне лише для авторизованих користувачів зі статусом адміністратора):надає можливість створення нової пропозиції. Після натискання на цей пункт користувач буде перенаправлений на сторінку створення нової пропозиції, де він може ввести всю необхідну інформацію та записати пропозицію до бази даних.

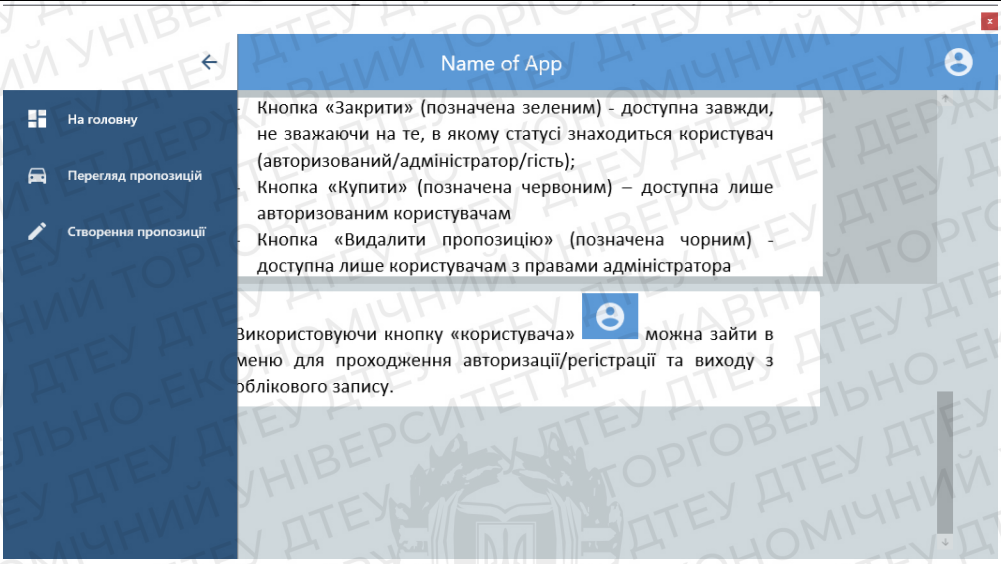

Рисунок 3.4. Бокове меню
Джерело: побудовано автором

При натисканні на кнопку меню «Перегляд пропозицій» відкривається відповідне вікно «Пропозиції» рис.3.5. Користувачу одразу пропонується до ознайомлення всі наявні пропозиції автомобілей. При натисканні на одну з пропозицій відкриється картка обраної пропозиції зі всіма її данними.

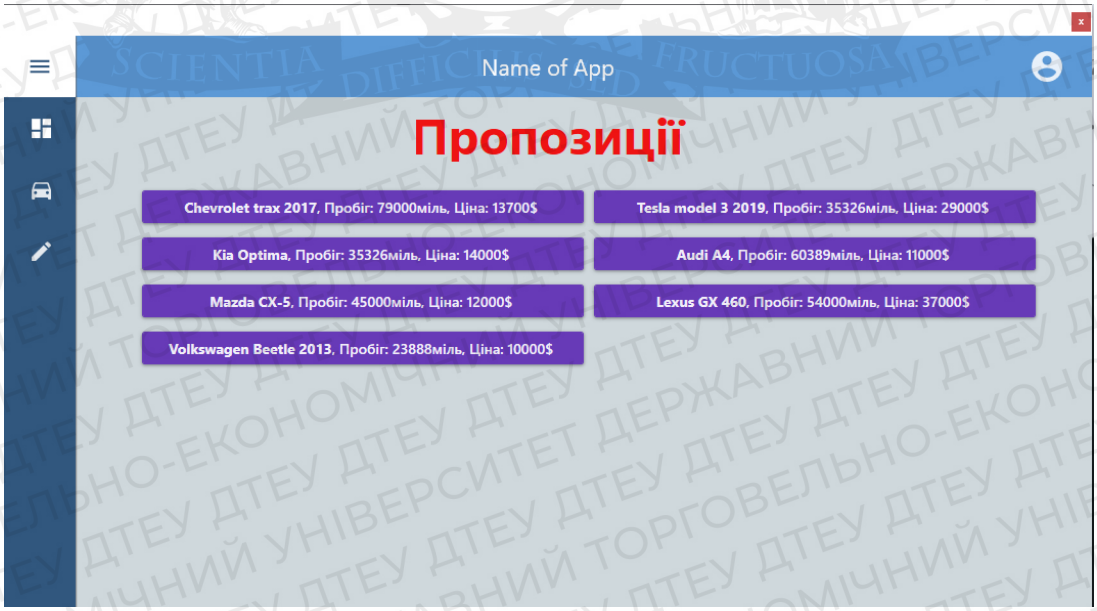

Рисунок 3.5. Вікно асортименту
Джерело: побудовано автором

При натисканні на кнопку меню «Створення пропозиції» відкривається відповідне вікно «Створення пропозиції» рис.3.6.

Вікно створення пропозиції має наступні поля:

Назва пропозиції: це текстове поле, в яке користувач вводить назву бренду та моделі автомобілі, також можна додати такі показники як назву комплектації. Воно дозволяє користувачеві чітко ідентифікувати та описати пропозицію.

Тип кузова: це поле вибору, де користувач може вибрати тип кузова автомобіля зі списку доступних варіантів, наприклад, седан, хетчбек, кросовер і т. д.

Колір: в цьому полі користувач може вказати колір автомобіля, наприклад, червоний, синій, чорний тощо.

Тип приводу: це поле вибору, де користувач може обрати тип приводу автомобіля, наприклад, передній, задній або повний привід.

Тип палива: поле вибору, де користувач може вказати тип палива з переліку, який використовується автомобілем, наприклад, бензин, дизель, електрика.

Об'єм двигуна: в цьому полі користувач вводить об'єм двигуна автомобіля числовим значенням, наприклад, 1.6 або 2.0.

Ціна: в цьому полі користувач вводить ціну пропозиції числовим значенням, наприклад, 15000 або 25000.

Загальний пробіг: користувач вводить загальний пробіг автомобіля числовим значенням, наприклад, 50000 або 100000.

Нотатки: в цьому текстовому полі користувач може ввести додаткову інформацію або нотатки щодо пропозиції, наприклад, деталі про стан автомобіля, його історію обслуговування тощо.

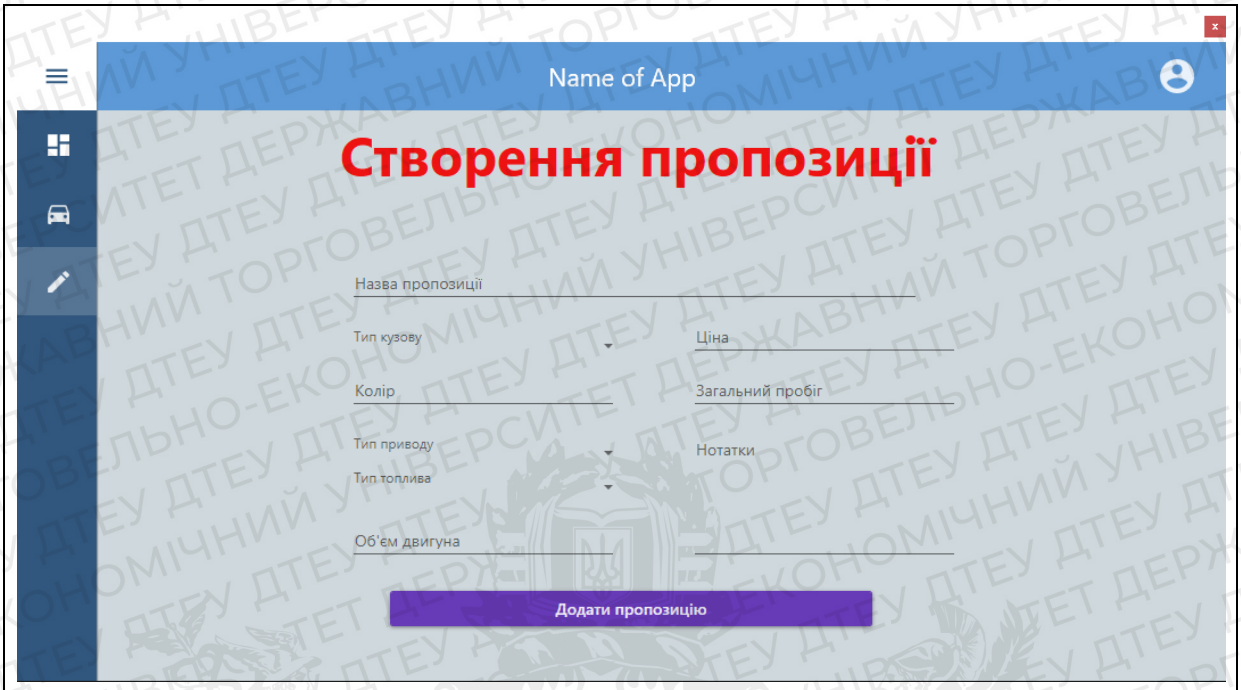

Рисунок 3.6. Меню створення номенклатури

Джерело: побудовано автором

3.3 Висновок до Розділу 3

В третьому розділі випускної кваліфікаційної роботи було описано процес вкрай важливої часини роботи над програмним модулем, а саме тестування. Використання технік тестування, таких як класи еквівалентності, аналіз граничних значень та попарне тестування, допомогло виявити помилки та забезпечити належну якість додатку.

Результуюча таблиця або звіт про тестування є необхідною частиною документації тестування. Вона фіксує всі проведені тестові сценарії, входи, очікувані результати та фактичні результати. Це допомагає команді розробників, тестувальникам та менеджерам проекту зрозуміти проведені тести, виявлені проблеми та зроблені кроки для їх виправлення.

Тест-кейси є важливим інструментом для виявлення помилок та неправильної роботи програмного забезпечення. Вони допомагають

перевірити функціональність модуля та виконати регресійне тестування після змін або оновлень.

Друге, про що було розроблено, це інструкція користувача по роботі з системою. Вона надає детальний опис різних елементів і функціональності програмного додатку, допомагає користувачам зрозуміти, як використовувати додаток і використовувати певні елементи, такі як авторизація, перегляд пропозицій і створення нових пропозицій.

Описуючи інструкцію користувача, було надано загальну інформацію про систему, включаючи її основні функції та цілі використання. Також, в цих блоках інструкції включено й роз'яснення основних функцій та можливостей системи. Кожна функція описана докладно, вказуючи кроки, необхідні для її виконання, та пояснюючи, які результати можна очікувати.

ВИСНОВКИ

В результаті випускної кваліфікаційної роботи було розроблено інтуїтивний, швидкий, та легкий у користуванні програмний додаток мовою C# для клієнтів та робітників компаній що займаються торгівлею авто з пробігом. Зважаючи на те, що технології доступні нам сьогодні полегшують наше життя. Слід користуватися цими, наданими нам попередниками, можливостями. Інновації за котрими ми слідкуємо у майбутньому стануть незамінними технологіями, як і ті, котрими ми користуємось сьогодні.

Сучасні інструменти праці допомагають створювати інші, не менш важливі інструменти котрими ми користуємось повсякдень. Такі знаряддя праці існують в кожній сфері нашого життя починаючи від медицини й розваг, закінчуючи математичними розрахунками й торгівлею.

Областю дослідження цієї випускної кваліфікаційної роботи була сфера реалізації вживаних авто, а точніше автомобільний ринок бувших у використанні автомобілей на прикладі тих, що припливають на України з інших країн. Слід зазначити, що ця сфера не стоїть на місці й можна сміливо сказати, що за останні п’ять років вона виросла у декілька раз. Про це свідчать статистичні данні Державної митної служби України та Федерації роботодавців автомобільної галузі України. Позитивно відгукуються на ІТ-сфері факт того, що ринок стрімко розвивається й росте, ці зміни сьогодні помітні практично в кожній компанії. Всім їм потрібно мати свій, індивідуальний інформаційно-статистичний ресурс для закликання нових клієнтів, статусності, підвищення ефективності роботи й пришвидшення виконання дій за для економії часу.

					ДТЕУ 121 07-08 БР		
Зм.	Аркуш	№ докум	Підпис	Дата			
Зав. каф		Криворучко О.В.		28.04.23	Програмний модуль підприємства з продажу автомобілів з пробігом	Стадія	Аркуш	Аркушів
Керівник		Власенко Л.О.		28.04.23		ВП	44	46
Гарант		Рзаєва С.Л.		28.04.23		Факультет інформаційних технологій 4 курс, 7 група		
Розробив		Глижін В.С.		28.04.23	Висновки			

Сучасне програмне забезпечення тісно пов’язане зі всіма сферами бізнесу, а розробка цього програмного забезпечення тісно пов’язана з методологією розробки й конструювання програм об’єктно-орієнтованого програмування. Можна з впевненістю сказати, що саме ця методологія була та є рушійною силою в індустрії програмування протягом вже дуже довгого часу і залишатиметься такою в осяжному майбутньому. Пришвидшуючи роботу, покращуючи структуру програми як її окремі модулі так і її в цілому, покращуючи «читаємість» коду та полегшення його підтримки в майбутньому всі ці дії допомагають девелоперам з усього світу працювати ефективніше й марнувати менше часу на одну й ту саму задачу.

Як додаткове підтвердження почесного першого місця серед методологій – можна привезти факт, що сьогодні практично всі основні мови програмування створенні для написання коду за методологією об’єктно-орієнтованого програмування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Метт Вайсфельд, «Об’єктно-орієнтоване мислення». ISBN 978-0321861276
2. WEB-портал dou.ua «Рейтинг мов програмування 2022» [Електронний ресурс] URL: https://dou.ua/lenta/articles/language-rating-2022/ (дата звернення 05.01.2023)
3. Аукціон COPART [Електронний ресурс] URL: https://www.copart.com/ (дата звернення 07.05.2023)
4. Аукціон Manheim [Електронний ресурс] URL: https://www.manheim.com/ (дата звернення 07.05.2023)
5. Аукціон IAAI [Електронний ресурс] URL: https://www.iaai.com/ (дата звернення 07.05.2023)
6. Державна митна служба України [Електронний ресурс] URL: https://bi.customs.gov.ua/trade/#/ (дата звернення 08.05.2023)
7. Федерація роботодавців автомобільної галузі [Електронний ресурс] URL: https://fra.org.ua/ru/st/statistika/infoghrafika/import-legkovikh-avtomobiliv-2015-2019 (дата звернення 01.05.2023)
8. Навісний WEB-портал Бізнес Цензор [Електронний ресурс] URL: https://biz.censor.net/resonance/3143249/yak_avtomobl_vezut_v_ukranu_z_ssha (дата звернення 21.04.2023)
9. «Записки про структурне програмування» Єдгер Вайбе Дейкстра. [Електронний ресурс] URL: https://www.cs.utexas.edu/users/EWD/ewd13xx/EWD1308.PDF (дата звернення 11.04.2023)
10. «WPF Recipes in C# 2008 A Problem-Solution Approach» Sam Noble, Sam Bourton, and Allen Jones ISBN-13 (pbk): 978-1-4302-1084-9, 746c. (дата звернення 11.04.2023)

					ДТЕУ 121 07-08 БР			
Зм.	Аркуш	№ докум	Підпис	Дата				
Зав. каф		Криворучко О.В.		23.12.22	Програмний модуль підприємства з продажу автомобілів з пробігом	Стадія	Аркуш	Аркушів
Керівник		Власенко Л.О.		23.12.22		СВД	46	46
Гарант		Рзаєва С.Л.		23.12.22		Факультет інформаційних технологій 4 курс, 7 група		
Розробив		Глижін В.С.		23.12.22	Список використаних джерел			

ДОДАТКИ

ДОДАТОК А

Код класу Deal.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace CarsFromWest_master
   class Deal    {
      public int id { get; set; }
      private string name;
      private int engine;
      private int odometet;
      private string fuel;
      private string color;
      private string body;
      private string drive;
      private string notes;
      private int price;
      private string _image;
      public string Image
      {get
        {switch (body)
          {case "Купе":{
                return $"{Environment.CurrentDirectory}\\image\\купе.jpg";}
            case "Кабріолет":
              {return $"{Environment.CurrentDirectory}\\image\\кабріолет.jpg";}
            case "Кросовер":
              {return $"{Environment.CurrentDirectory}\\image\\кросовер.jpg";}
            case "Ліфтбек":
              { return $"{Environment.CurrentDirectory}\\image\\ліфтбек.jpg";}
            case "Мінівен":
              {return $"{Environment.CurrentDirectory}\\image\\мінівен.jpg";}
            case "Пікап":
              {return $"{Environment.CurrentDirectory}\\image\\пікап.jpg";}
            case "Позашляховик":
              { return $"{Environment.CurrentDirectory}\\image\\позашляховик.jpg";}
            case "Седан":
              {return $"{Environment.CurrentDirectory}\\image\\седан.jpg";}
            case "Універсал":
              {return $"{Environment.CurrentDirectory}\\image\\універсал.jpg";}
            case "Фургон":
              {return $"{Environment.CurrentDirectory}\\image\\фургон.jpg";}
            case "Хетчбек":
              {return $"{Environment.CurrentDirectory}\\image\\хетчбек.jpg";}
            default:
              return _image;              }
```

```
}
    }
    public string Name { get { return name; } set { name = value; } }
    public int Engine { get { return engine; } set { engine = value; } }
    public int Odometet { get { return odometet; } set { odometet = value; } }
    public string Fuel { get { return fuel; } set { fuel = value; } }
    public string Color { get { return color; } set { color = value; } }
    public string Drive { get { return drive; } set { drive = value; } }
    public string Notes { get { return notes; } set { notes = value; } }
    public int Price { get { return price; } set { price = value; } }
    public string Body { get { return body; } set { body = value; } }
    public Deal() { }
    public Deal(/*int id,*/ string name, int engine, int odometet, string fuel, string color,
string body, string drive, string notes, int price)
    {
      this.name = name;
      this.engine = engine;
      this.odometet = odometet;
      this.fuel = fuel;
      this.color = color;
      this.body = body;
      this.drive = drive;
      this.notes = notes;
      this.price = price;
    }
  }
}
```

ДОДАТОК Б

Код класу User.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CarsFromWest_master
{
    class User
    {
        public int id { get; set; }
        private string login, pass, status;
        public string Login { get { return login; } set { login = value; } }
        public string Pass { get { return pass; } set { pass = value; } }
        public string Status { get { return status; } set { status = value; } }

        public User() { }
        public User(string login, string pass, string status)
        {
            this.login = login;
            this.pass = pass;
            this.status = status;
        }
    }
}
```

Код класу AppContextDeal.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity;

namespace CarsFromWest_master
{
   class AppContextDeal : DbContext
   {
      public DbSet<Deal> Deals { get; set; }

      public AppContextDeal() : base("DefaultConnection") { }

   }
}
```