ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи на
ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА WEB-СЕРВІСУ ДЛЯ ОНЛАЙН ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ МОВОЮ C#**»

Виконав: студент 4 курсу, групи ПД-43
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)
Заїчко І.О.
(прізвище та ініціали)
Керівник Поперешняк С.В.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Київ –2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти -«Бакалавр» Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
"____" _____2022 року

З А В Д А Н Н Я НА ДИПЛОМНУ РОБОТУ СТУДЕНТА

ЗАЇЧКУ ІЛЛІ ОЛЕГОВИЧУ

(прізвище, ім'я, по батькові)

1. Тема роботи:«Розробка web-сервісу для онлайн обслуговування автомобілів мовою C#»

Керівник роботи: _____ Поперешняк С.В., к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «18» лютого 2022 року №.

2. Строк подання студентом роботи «03» червня 2022 року
3. Вхідні дані до роботи

Методи створення програмного продукту;

Науково-технічна література з питаннями по створенню архітектури, тестуванню та розробці питань, пов'язаних з програмним забезпеченням щодо створення webдодатку по обслуговуванню автомобілів;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).

4.1Аналіз аналогів та специфіка використання продукту.

4.2 Вимоги до розробки та оцінка якості системи.

4.3 Опис архітектури системи. 4.4 Опис використаних технологій.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Актуальність програмного забезпечення в наш час
2. Існуюче програмне забезпечення та особливість програмного продукту
3. Представлення роботи системи
4. Тестування особливостей системи
5. Архітектура бази даних
6. Логічна діаграма компонентів архітектури програмного забезпечення

6. Дата видачі завдання «11»квітня 2022

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.04-15.04	Виконано
2	Дослідження існуючих аналогів	16.04-17.04	Виконано
3	Вивчення програмних засобів розробки	19.04-23.04	Виконано
4	Проектування архітектури програми	25.04-30.04	Виконано
5	Розробка функціоналу	28.02-04.05	Виконано
6	Вступ, висновки, реферат	05.05-19.05	Виконано
7	Розробка обов‘язкових демонстраційних матеріалів	20.05-26.05	Виконано
8	Попередній захист роботи	01.05	
9	Здача роботи	13.06	

РЕФЕРАТ

Текстова частина бакалаврської роботи 57 с., 20 рис., 15 джерел.

Об'єкт дослідження –процеси станцій технічного обслуговування та їх автоматизація.

Предмет дослідження – web-додаток на основі обліку існуючих послуг.

Мета роботи – створити програмний продукт щоб автоматизувати ведення обліку та полегшити процеси станцій технічного обслуговування..

Методи дослідження – використовувались методи спостереження, аналізу та узагальнення.

Використовуючи в роботі метод аналізу були взяті до уваги наступні додатки, такі як РемОнлайн, Автосервіс PRO.CTO.UA, СТО Тачки, TQM Systems, та ін. системи для обліку послуг, та зберігання інформації в Базах Даних.

Загальною проблемою цих продуктів є у напівавтоматичному обліку послуг які надає станція технічного обслуговування.

Особливістю методу є усі опції, якими володіє програма, слугують саме для ефективної роботи СТО будь-якого розміру та масштабу.

Описано архітектуру та основні принципи розробки. За основу було взято технологію WebApi з серверної сторони та JavaScript з клієнтської сторони для основного функціоналу.

Додаток використовує вже існуючі бази даних. В якості серверу баз даних було взято MongoDB та бібліотеку Mongoengine для взаємодії з нею.

Отже, розроблено та описано веб-додаток, завданням якого облік та автоматизація процесів СТО.

У якості вихідних даних є таблиці послуг.

Існуючий додаток може бути використано у сфері ремонту автомобілів на станціях технічного обслуговування різних розмірів.

Галузь використання – станції технічного обслуговування автомобілів.

ЗМІСТ

Перелік позначень

ПЗ- програмне забезпечення.

СТО-станція технічного обслуговування.

БД-база даних.

EF- Entity Framework

API- Application Programming Interfaces

MVC- models, vies, controllers.

ВСТУП

Створення різноманітних рішень для різних галузей уже довгий період займає передові місця в світі. Реалізація програмних продуктів з кожним днем виходить на новий рівень. Саме створення програмного забезпечення різного типу надає змогу полегшити та автоматизувати 99% існуючих завдань. Тому в наш час неможливо уявити провідну галузь без використання ПЗ. Чому так сталось?

Інформаційний потік у будь-якій провідній галузі збільшується з кожною хвилиною, саме тому для обробки інформації спеціалісти із розробки програмних продуктів створюють автоматизовані програми для обліку інформації. Вивчаючи галузь розробки програмного забезпечення я зрозумів, що кожне покоління росте серед великого потоку інформації та знань, а це означає, що людям потрібно вибирати, аналізувати все більше даних, орієнтуватися серед них та створювати різні рішення.

Саме тому темою моєї бакалаврської роботи є розробка веб застосунку для автоматизації процесу надання послуг у сфері обслуговування автомобілів. Додаток розроблено для полегшення роботи галузі обслуговування автомобілів. Опції, якими володіє програма, слугують саме для обліку роботи СТО.

Вивчивши питання проблематики сфери обслуговування автомобілів я створив рішення яке полегшить облік. Одним із таких рішень є те що, облік користувачів автоматизовано, кожна надана послуга проведена через систему та записана у програмі для СТО. Спочатку ви вказуєте послуги, які надає організація та відповідний тарифний план.

Результатом досліджень архітектура програмного продукту допоможе працівникам сфери обслуговування автомобілів оброблювати великий потік інформації та розширювати з кожним днем свою клієнтську базу.

Не менш важливим моментом є спосіб розробки програмного продукту. У цій роботі розкрито принципи розробки браузерних та серверних застосунків (клієнтсерверна архітектура). Основний напрямом є .Net платформа та мова програмування C# - продукти від компанії Майкрософт. Саме ці інструменти використовують для різноманітних видів рішень наприклад десктопні програми (лінукс, віндовс), андроїд, веб, ігри.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

Завдання до Бакалаврської роботи, оптимізація та автоматизація процесів в галузі автомайстерень, СТО, тощо. Таким чином, було досліджено галузь обслуговування автомобілів для конкретного поставлення цілей своєї роботи, а також розуміння проблематики данної галузі, її маштаби, її навантаження та процеси.

Вивчаючи галузь автообслуговування можу зазначити декілька факторів:

1. Мою увагу привернула модель процесу підвищення якості на СТО. Надання послуг представлено на прикладі схеми взаємозв'язаних процесів. Послуги починають надаватись з процесу маркетингового дослідження, вхідними даними якого є очікування користувачів та загальні потреби галузі, результатом діяльності вважається інформація про очікування споживачів, яка стає першим етапом процесу планування номенклатури послуг, які надаються, в якості управління саме цим процесом, рекомендовано використовувати методику оцінки технічного рівня автосервісу, ця технологія дозволяє співробітникам галузі детально оцінювати технічний рівень підприємства і більш детальніше скласти перелік послуг, які будуть надані, який в свою чергу рекламує підприємство для споживача за рахунок здатності задовольнити його конкретні вимоги.

Результат даного процесу будуть початком і управлінням двох інших процесів. Перелік придбання необхідного обладнання буде вхідним процесом закупівля обладнання, а перелік послуг та їх вартість, що надаються управлінням процесу, який аналізує вимоги конкретного споживача, разом з створеною методикою оцінки потреб споживачів та аналіз незадовільних відгуків споживачів.

Результатом процесу можна вважати набір послуг можна оцінювати як процентне співвідношення вже наданої номенклатури послуг до загальної номенклатури послуг, які пропонуються, як порівняння технічної оснащеності в потрібний час з технічного оснащення за такий же проміжок в минулому році і як результат відношення витрат закупленого обладнання за необхідний проміжок часу.

Даним процесом я зацікавився найбільше у своїй роботі буду використовувати цей етап. Звісно ж, для споживача головним є процес надання послуг, вхідними даними якого є набір замовлень які потрібно надати споживачу з несправним транспортним засобом. Предметами даного процесу буде досвідчений та мотивований завдяки «програмі мотивації співробітників» персонал, який безпосередньо є результатом додаткового процесу керування персоналом і інфраструктура відповідна до вимог споживачів, вона буде формується з виходів додаткових процесів керування середовищем обслуговування автомобілів, ремонтом і закупкою обладнання.

Результатом процесу є вимірювання відношення автотранспортних засобів, які обслужені до кількості зареєстрованих автомобілів на пункті запису обов'язково враховуючи часу надання послуги. Сам процес надання послуги націлений на технічне забезпечення потреб споживача і має кілька виходів: справний автотранспортний засіб, цим самим задоволений споживач, який в свою чергу залишає відгук у відділі роботи з клієнтами.

Далі проаналізуємо в бізнес процесі надання послуг, етап аналізу вимог для конкретного споживача задля реалізації цілі встановлення наявності претензій щоб контролювати якість обслуговування та позитивну оцінку від споживача за допомогою різного виду методику, після того зібрані та проаналізовані дані, які постійно оцінюються за показниками в залежності від загальної кількості задоволених, кількість зареєстрованих претензій у

співвідношенні до загальної кількості клієнтів які зареєстрували свій негативний відгук за певний час, відразу обробляються в якості вхідних даних для покращення процесу управління наданими послугами, кінцевий результат за рік аналізуються з боку керівництва підприємства.

Проаналізувавши підвищення якості галузі, я перейшов до управління якістю послуг автосервісу на основі стандартів організації.

Методика з оцінки рівня задоволеності споживачів якістю послуг, що надаються дозволяє не тільки обґрунтовано приймати рішення, спрямовані на задоволення вимог і запитів споживачів, але і робити висновки про конкурентоспроможність підприємства. Для здійснення методики оцінки задоволеності споживачів розроблені.

Для аналізу також було проаналізовано стандарт організації «Система збору та аналізу даних для оцінки рівня задоволених споживачів сервісу обслуговування автомобілів». Важливим фактором є система по обробці претензій та пропозицій споживачів, до даної системи повинна приділятись велика кількість часу для аналізу та впровадження пропозицій та скарг. Ця методика визначає обробляє всі звернення по скаргам та пропозиціям споживачів, детально аналізує оброблену інформацію з можливістю впровадження.

Для аналізу розбіжностей (відхилень) між потенційним клієнтом і галуззю надання послуг була створена інструкція «Вимірювання розбіжності між споживачем і постачальником послуги».

Також слід зазначити що підтримка якості обслуговування завжди повинна працювати. Так як автобудівництво не стоїть на місці та дуже швидко розвивається.

Це на свідчить про два фактори.

1. Стрімке збільшення споживачів;

2. Впровадження новітніх технологій які постійно потрібно вивчати, щоб в подальшому коректно їх обслуговувати.

Галузь автобудівництво та обслуговування автомобілів взаємозв'язанні, тому що, кожен автомобіль, який буде створений за новою технологією повинен ремонтувати автомобіль по новій схемі ремонту. Це не відноситься до всіх автомобілів та ремонту всіх деталей. Але впровадження новітніх технологій на станціях технічного обслуговування край важливий фактор для розширення бази споживачів та гарного прибутку, незалежно чи це велике підприємництво по обслуговуванню чи звичайна автомайстерня.

Проаналізувавши станції технічного обслуговування, можу зазначити що вони також поділяються на декілька категорій:

1.Спеціальними вважаються станції з обмеженним набором послуг, в них в наявності є спеціальне обладнання для надання послуг спеціального призначення;

2.Загальні СТО, вони потребують більшої площі преміщення та значно більше персоналу;

3.Станції обслуговування офіційних дилерів;

4.Ремонт та обслуговування автомобілів тільки певних брендів, заміна деталей тільки від офіційного дилера;

5.Також ми не могли не виділити «Гаражні СТО», цей варіант є найменш затратним. Але і має свої мінуси.

До мінусів можемо віднести те що для для роботи даного виду СТО не потрібно знімати приміщення, обслуговування ведеться у звичайному гаражі де працює максимум 2-3 людини. Немає резервів комплектуючих та малий набір послуг. Дане СТО зазвичай використовують у незначних поломках автомобіля де ремонт займає максимум 2 години.

Рис.1.1- Приклад загальної станції технічного обслуговування

.

СТО надає швидкий ремонт тільки при наявності власного складу з комплектуючими та запчастинами. Це обслуговування вашого автомобіля відразу протягом декількох годин (час залежить від проблеми яку вирішує спеціалст). Під час очікування як правило у вас є можливість пройти до кімнати очікування яка облаштована засобами для приємного проводження часу очікування (телевізор, кава, журнали).

ВИСНОВОК ДО РОЗДІЛУ 1

Висновком можу зазначити, що предметна галузь з обслуговування автомобілів базується на свого рівня якості обслуговування та вдосконалення можливостей. Вся проблематика галузі є в типах обслуговування. Контроль якості обслуговування повинен бути на регулярній основі, його аналіз та впровадження повинно бути не рідше чим один раз на рік. Я вважаю, що чим

краще здійснюється обробка скарг та пропозицій, тим швидше підприємство здобуває пріоритет за рахунок збільшення споживачів у геометричній прогресії.

Роблячи висновки з потреб галузі, можу зазначити що автоматизація даної сфери можлива за допомогою використання програмних технологій, які потрібні щоб реалізувати швидке обслуговування клієнтів, ведення документації та обліку в онлайн вигляді за допомогою звітів (Excel звіти). Реалізація функціоналу за допомогою онлайн запису на обслуговування клієнтів, полегшує розподіл робочого часу та місця а також приваблює клієнта.

Вивчивши предметну галузь, можу зазначити, що деякі функціональні здібності будуть описані і в моєму дипломному проекті. Використання мого дипломного програмного продукту, проаналізувавши типи СТО, найбільш зручне у загальних та спеціальних станціях технічного обслуговування.

РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМ НАЛОГІВ

2.1.РемОнлайн

Програма для автосервісу та СТО, створена для автоматизації бізнеспроцесів та спрощення контролю. Аналізуючи програму можна виділити три основні функціональні здібності:

1.Облік замовлень. Ведення обліку клієнтів та їх замовлень, вистежування історії робіт та використаних матеріалів.

2.Облік замовлень. Налаштування ланцюжків статусів для різних типів замовлень, для швидкого обслуговування та контролю термінів.

3.Планувальник. Запис на технічне обслуговування на дні та тижні наперед, дана можливість дає змогу рівномірно розподілити навантаження між співробітниками та ресурсами.

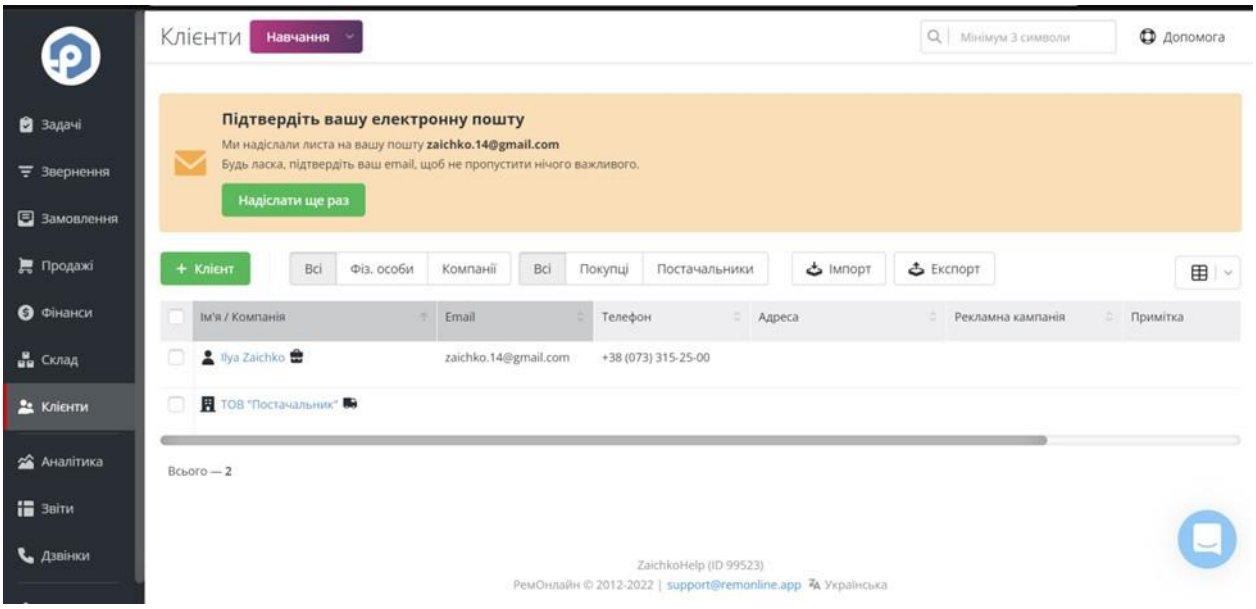


Рис. 2.1 – Інтерфейс програми РемОналайн

2.2.Автосервіс PRO.CTO.UA

Даний мобільний додаток реалізує зручний спосіб скористатися послугами автосервісу а також відображення актуальних новин і акцій СТО. Ознайомившись з даним додатком може виділити такі особливості:

1.Спостерігати історії свого автомобіля через онлайн книжку сервісу

2.Реалізація у додатку камер відеоспостереження, дає можливість спостерігати за ремонтом автомобіля в онлайн режимі.

3.Планування ремонту свого автомобіля згідно до заводських стандартів.

4.Зручно організована програма лояльності та система бонусів.

Рис.2.2 - Інтерфейс програми PRO CTO

2.3.CTO Тачки

1.Додаток спрощує спілкування клієнтів з менеджерами сервісу.Також є можливість отримувати у вигляді push-повідомлення сповіщення про знижки, важливі події на СТО.

2.Полегшує отримання вартості запчастин, їх технічні характеристики. Автоматизація у підборі необхідних деталей і вирішить будь-яку проблему.

3.Реалізовані рекомендації для водіїв, які допомагають в експлуатації.

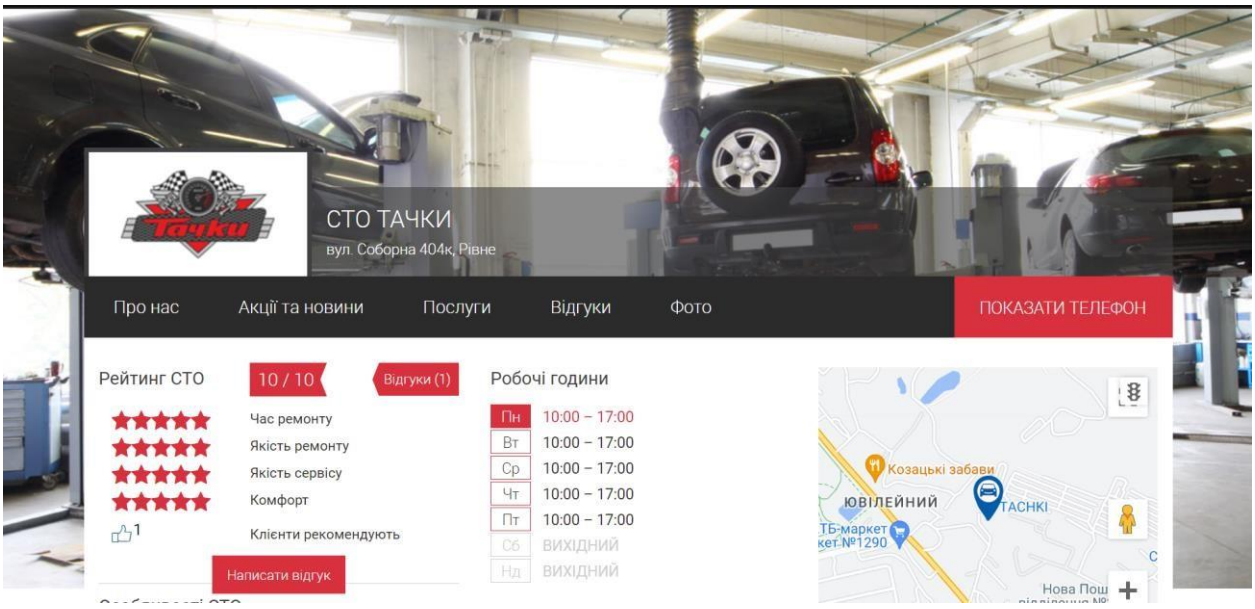


Рис.2.3- Інтерфейс СТО Тачки

2.4.YuKoSoft Автосервіс.

Додаток дозволяє керувати персоналом, відстежувати рух коштів СТО, вести облік робіт та запчастин у замовлення-нарядах, створювати єдину базу клієнтів та ін.. У представленому аналогу реалізований наступний функціонал:

1.Вартість робіт може нараховуватись з нормо-годин або фіксованої ціни.

2.Підтримка роботи як з локальною offline базою данних, так і з online БД через інтернет.

3.Робота з програмою розрахована на багато користувачів. Єдина база даних.

4.Зміна структури таблиць. Додавання, найменування та видалення колонок таблиць

5.Редактор дизайну форми.Створення свого виду форми, шляхом перетягування полів у будь-яке місце

6.Можливість зберігати зображення, сканованих копій документів та файлів безпосередньо в базі

7.Редактор друкованих форм. Створення своїх форм.

8.Експорт даних в Excel.

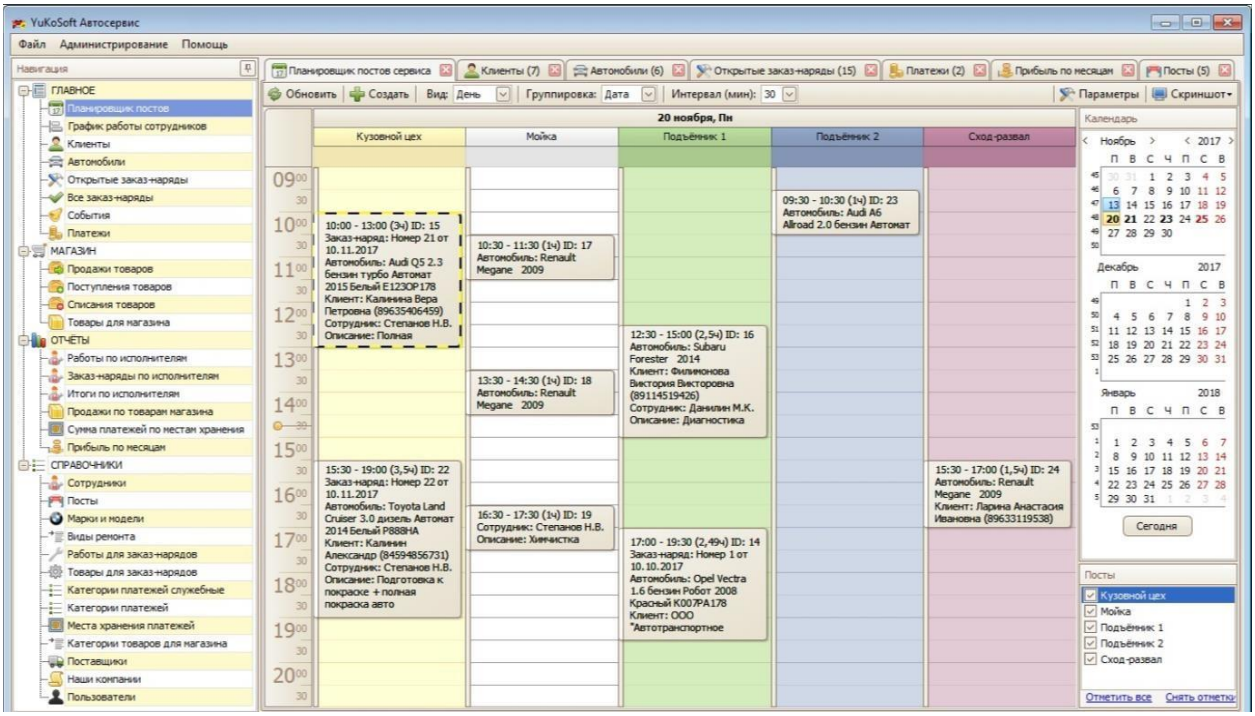


Рис.2.4 - Представлення функціоналу програмного продукту YuKoSoft Автосервіс

2.5.TQM Systems

Основною задачею застосунку є реалізація управлінського обліку на СТО. Данний програмний продукт представляє собою наступний функціонал:

1.Ведення обліку виконання обслуговування автомобіля документом "Замовлення-наряд" зі зберіганням історії і автоматичним формуванням друкованих форм замовлення-наряду, акт огляду автомобіля та ін. первинних документів

2.Вистежування стану взаєморозрахунків з деталізацією до "Замовленнянаряду"

3.Формування звітів про діяльність автосервісу та розрахунок прибутку.

4.Використання довідників для автосервісу : моделі автомобілів, нормогодини, види ремонту.

5.Процес імпорту проводиться наступним чином: з каталогу дістається калькуляція (список робіт) до файлу потрібного формату. Потім створюється документ "Замовлення-наряд", на закладці "Роботи", вибравши клавішу "Заповнення", проводиться завантаження файлу до замовлення-наряду, паралельно в довіднику номенклатури створюються роботи з ремонту, які завантажуються.

6.Формування друкованих форм документа "Замовлення-наряд": акт огляду, робочий лист, замовлення-наряд, чек до замовлення-наряду. Облік прийнятих на обслуговування автомобілів на території підприємства, в цехах, на стоянках.

7.Автоматичний розрахунок вартості виконаних робіт в замовлення-наряді згідно вказаними правилами ціноутворення. Підбір запасних частин і використаних матеріалів в замовлення-наряді, враховуючи залишки на складі.

8.Відображення переміщень запасних частин в процесі ремонту (видача зі складу для замовлення-наряду, повернення на склад, переміщення між замовленнянарядами).

9.Контроль складських залишків.

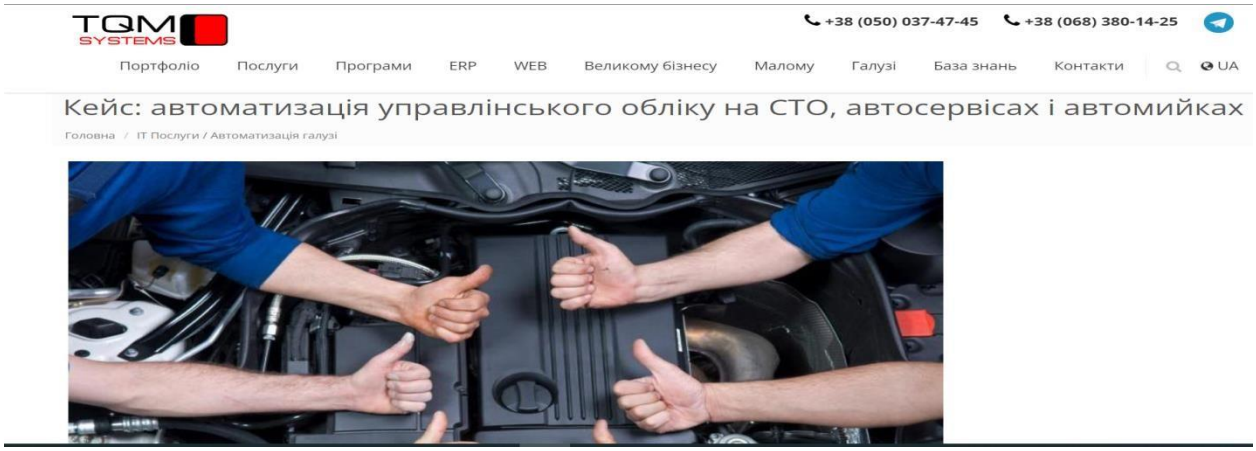


Рис2.4 - Представлення інтерфейсу програми TQM Systems

Дослідивши програмні аналоги: РемОнлайн, Автосервіс PRO.CTO.UA, СТО Тачки, YuKoSoft Автосервіс, TQM Systems, було створено таблицю основних функціональних здібностей програм та їх особливості.

Табличка 2.1- Основні функціональні здібності програм

Показник	**РемОнлайн**	**Автосервіс PRO.CTO.UA**	**TQM Systems**	**YuKoSoft Автосервіс.**
Платформи	Web	Android, iOs, Web	Web	Windows
Категорії витрат	-	-	-	Пробний період
Облік замовлень	+	-	+	+
Онлайн замовлення	+	+	+	+
Онлайн розрахунок	-	-	+	+
Планувальник	-	+	+	+
Онлайн база клієнтів	-	-	+	+
Ведення карти автомобіля	-	-	+	+
Використання «замовлення-наряд»	-	-	+	+
Контроль складських залишків.	-	-	+	-

Підтримка онлайн камер	-	+	+	-

ВИСНОВОК ДО РОЗДІЛУ 2

З наведених прикладів можна зробити висновок, що кожна вище представлена програма відрізняється одна від одної, не тільки інтерфейсними представленнями але і функціональними. Всі програмні продукти об’єднані однією спільною функціональною можливістю- це можливість записатись на сервіс обслуговування онлайн. У вище наданих аналогах представленні в основному два типи програмних продуктів.

Перший тип-це програми які реалізовані більш для надання реклами, повідомлень, новин, програм лояльності та онлайн запису. (СТО Тачки, PRO.CTO.UA)

Другий тип програм реалізований не тільки для онлайн запису, а для аналітики проробленої роботи, використовування бази даних клієнтів та її розширення, розподілення роботи та виставлення графіку, аналізування проблеми до прибуття до станції технічного обслуговування.(РемОнлайн, TQM Systems)

Також, проаналізувавши програми аналоги різних типів, я хотів би виділити найбільш сумісний аналог до програмного продукту який представлений у цій бакалаврській роботі.

Це ПО YuKoSoft Автосервіс. Дана программа реалізує в собі планувальник який розприділяє та полегшує роботу СТО. Також являє собою надзвичайно зручний інтерфейс по створенню нової задачі, оптимізована зручна робота з клієнтами як постійними та і новими за допомого синхронізації з БД.

Експорт даних в Exel для зручності ведення документації. На завершення можу зазначити що всі вище представлені рішення відрізняються одне від

іншого спроможністю вирішувати обмеженний тип задач, найбільш спроможною для використання в широкій галузі з автообслуговування я вважаю YuKoSoft Автосервіс та програмний продукт який буде описано в цій дипломній роботі “Car Service”.

РОЗДІЛ 3. АНАЛІЗ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ “Car Service”

В даному розділі представляються засоби та технології розробки програмного продукту “Car Service”, за темою бакалаврської роботи «Розробка вебсервісу для онлайн обслуговування автомобілів мовою C#».

Visual Studio

Розробником є компанія Microsoft.Програма містить комплексний функціонал для розробки програмного забезпечення. Я виділю три особливих функціональних можливостей:

- Швидке знаходження дефектів та помилок у програмному продукті який розробляється. Це зручно коли сталася помилка в класі який пов’язаний з іншими класами, і помилку дуже важко знайти вручну. А завдяки функції відлагодження програм система швидко знаходить помилку, тим самим економить час та нерви розробника.
- Редактор початкового коду системи, так зване редактор серця програми. Середовище автоматично редагує початковий код, який створив розробник.
- Автоматизація побудови програми. Автоматичне побудова завдання які використовує розробник постійно а також написання скриптів, реалізує

порядок виклику коду та етапи компіляції. При створені за основу бралася мета автоматизувати роботу компонувальника та компілятора.

Також інтегроване середовище містить в собі інтерпретатор, компілятор, систему контролю версіями програмного продуту а також інструменти які полегшують графічну розробку програми. Основною задачею будь-якого інтегрованого середовища є реалізація швидкого , зручного, та адаптованого написання програми розробником. Щоб контролювати та змінювати програму можна було в одному середовищі, не зважаючи чи нам потрібно внести зміни до графічного інтерфейсу чи до логіки певного класу.

Особливістю Visual Studio є те, що це середовище можна використовувати як для розробки невеликих консольних додатків та і великих програмних продуктів з графічними інтерфейсами за допомогою технологій Windows Forms, веб сайти які можна реалізувати за допомогою вбудованих фреймворків.

Розробка додатку виконувалась на основі платформи .Net Framework-

ООП

Об’єктно орієнтоване програмування- методологія розробки програмного забезпечення з використання програми як сукупність об’єктів які взаємодіють між собою, і кожен з них є екзимляром іншого класу в той час класи унаслідуються один від одного. Основні принципи розробки ПЗ з використанням методології ООП:

- Інкапсуляція-відповідає за приховування даних які знаходяться всередині та деталі по реалізації від інших компонентів які використовує ПЗ.
- Наслідування- програмісти даний принцип являють одним із основних в ООП. Тому що, використовуючи наслідування можна створити загальний клас, так як наслідування створює ієрархічну структуру об’єкту. Клас далі може наслідувати інші класи і використовувати методи тих класів які

унаслідував. Кожен клас впливає на поведінку базового класу. Крім базового класу також існують:

клас-батько, підклас, дочірній клас, клас-нащадок.

- Принцип наслідування реалізує інший принцип ООП- це поліморфізм. Реалізує визивання перевизначеного методу за допомогою змінної батьківського класа щоб отримати його поведінку яка співпадає з класом який наслідується.

- Принцип Абстарція, об'єктами в ООП є моделі навколишнього середовища, таких як Студент, Схема, Замітка у блокноті, тому виділяють ті властивості цього поняття, які необхідні у використанні для конкретного випадку щоб вирішувати конкретну програмну проблему..

Платформа .Net

Це новий погляд на розробку програмних продуктів, технологія сприяє створенню та виконанню веб-служб та програм Windows. .Net Frameworkрозроблена компанією Micrisoft.Оновлення платформи здійснюється кожен місяць задля виправлення помилок які виникають з безпекою даних та автоматизацією розробки програмних продуктів.

.Net платформа включає в себе такі компоненти та технології: NET Framework(WPF, Windows Forms, ASP.NET)

NET Core(ASP.NET Core, UWP)

NET Standart

Mono for Xamarin(Android, iOS, OS X)

За основу технології було взято середовище CLR. Це середовище управляє пам'яттю, виконанням потоків, виконанням коду, відповідає за перевірку безпеки коду, компіляцією та іншими системними службами.

Бібліотека класів є базовою об'єктно-орієнтованою колекцією повторно використовуваних типів, які використовуються для розробки додатків –

починаючи від розробки звичайних додатків, що запускаються за допомогою командної стрічки, та додатків з графічним інтерфейсом (GUI) і закінчуючи програмами, що реалізують технологічні можливості ASP.NET(веб-форми та веб-служби XML).

Класи колекцій .NET Framework надається набір інтерфейсів для створення власних класів колекцій. Користувацькі класи колекцій поєднують з класами .NET Framework.

Зазвичай у платформі .Net Framework для вирішення задач використовують рішення для програм:

- Консольні програми. Ці програми виристовують клас System.Console для виконання вводу та виводу символів. У класа System.Console є методи в яких реалізовано зчитування певних символів або ж цілі стрічки символів.
- Програми із графічним інтерфейсом Windows (Windows Forms). Платформа Windows Forms підтримує широкий спектр функцій для розробки додатків, включаючи елементи керування, графіку, звязування даних та занесення користувача. Можна вважати особливістю Windows Forms є експлуатація візуального конструктора з функцією використовування Visual Studio для спрощення створення програм Windows Forms.
- Програми Windows Presentation Foundation (WPF).Данна платформа не має залежності від рішення і використовує векторний механізм візуалізації, який спроможний використовувати сучасні переваги графічних обладнань. WPF надає комплексний набір функцій для розробки програм: привязка даних, використання анімації, стилів, шаблонів, документів, мультимедіа та топографічні функції. Особливістю можу визначити те, що WPF є частиною .NET, тому є можливість створювати програми, що включають інші елементи .NET API

- Програми ASP.NET. Платформа для створення веб-сайтів та вебпрограм за допомогою HTML, CSS, JavaScript. Розробка здійснюється за декількома технологіями: веб-форми, MVC, веб-сторінки, веб-API.
- служби Windows; Служби Microsoft Windows, минула назва служби NT, реалізують створення довготривалих виконуваних програм, процес збірки програми відбувається у власних сеансах Windows. Для служб не передбачено інтерфейс користувача. Вони включаються автоматично при запуску Windows, їх також можна зупиняти та перезапускати.
- Сервісно-орієнтовані програми, що використовують Windows Communication Foundation (WCF). Надає можливість розробникам будувати надійні рішення з підтримкою транзакцій та міжплатформної інтеграції, дає змогу взаємодіяти з існуючими інвестиціями.
- Програми, які підтримують бізнес-процеси Windows Workflow Foundation (WF). Дозволяє користувачам створювати користувацькі та системні робочі процеси в програмах які створені для Windows Vista, Windows XP, Windows server 2003 и Windows та серверних операційних систем 2008.

Досліджуючи платформу .Net Framework я можу зробити висновки, що за останні декілька років Microsoft зробила величезну роботу з вдосконалення своїх технологій до загальних трендів та підходів, давно популярних в інших.

Багаторівнева архітектура

При розробці програмного продукту “Car Service” використовувалась багаторівнева архітектура розробки. Даний архітектурний шаблон є найпоширенішим, обмежень за кількістю і типом рівнів ніяких немає, проте в більшості випадків архітектура складається з чотирьох рівнів: представлення даних, бізнес логіка, зберігання данних та база данних.

Рис.3.1- Схема багаторівневої архітектури

При розробці програмне забезпечення було розділене на модулі, які потім розроблялись окремо один від одного, взаємодія між частинами була мінімальною, забезпечивши міграцію, модифікацію та повторне використання.

При використанні багаторівневого шаблону ПЗ ділиться на сутності, які називають рінями. Кожен рівень – це група модулів, що надають взаємопов'язаний набір сервісів. Їх застосування має бути односпрямованим. Рівні розділяють ПО, причому кожна частина доступна через публічний інтерфейс.

При використанні даного шаблону в розробці мною був помічений недолік. Чим більше рівнів- тим менша продуктивність програми. А також додавання рівнів збільшує повну вартість шляхом ускладнення системи. На

кінець можу зазначити, використовувати цей підхід слід для невеликих, простих застосунків і веб-сайтів. Також цей шаблон добре підходить, якщо обмежений бюджет і час.

SQLLite

Збереження та доступ до даних був реалізований на основі системи управління базою даних SQLLite. Саме дана база є офіційно рекомендованою ситемою управління для створення локального сховища даних на Windows 10.
SQLite - це вбудована кросплатформова БД, яка надає досить повний набір команд SQL і доступна у вихідних кодах (мовою C).

SQLite не відноситься до використання парадигми клієнт-сервер, це означає що SQLite не окрема робота з процесом, з яким взаємодіє програма, а являється бібліотекою, з якою програма компонується, і двигун стає частиною програми. Отже, протокол обміну використовуються за викликами функцій (API) бібліотеки SQLite. Цей підхід зменшує накладні витрати, час очікування та надає простоти програмі.Вихідні коди SQLite перебувають у public domain, це означає що ми на шляху розробки не зустрінемо жодних обмежень використання. Простота в реалізації досягається за рахунок того, що перед початком транзакції запису увесь файл, що зберігає БД, блокується.

Для роботи з SQLLite ми використали наайбільш простим і популярний підходом є використання Entity Framework, шляхом додавання пакетів Entity Framework у проект. Для роботи ми використовували два пакети: Microsoft.EntityFrameworkCore.SQLite (для взаємодії з БД SQLite) та Microsoft.EntityFrameworkCore.Tools (для генерації бази даних).

Entity Framework Core

EF- це спрощена, розширювана крос-платформна одна із популярних версій технології доступу до даних Entity Framework з відкритим вихідним кодом.

EF Core може служити як об'єктно-реляційний перетворювач (O/RM), який:

- Дозволяє .NET розробникам співпрацювати з базою даних, використовуючи об'єкти .NET.
- Спрощує необхідність написання великої кількості коду доступу до даних, який зазвичай прописували розробники.

WebAPI

Для обробки запитів отримання, додавання та видалення даних було використано технологію Інтерфейс прикладного - це готова постановка коснтрукції мови програмування, які дають можливість розробнику будувати складну логічну функціональність витрачаючи меншу кількість зусиль. Принцип роботи, реалізований так, що складний код розробник не бачить,цим самим забезпечивши простоту використання.

Web API є способом реалізації програми ASP.NET, який назначений для роботи в стилі REST (Representation State Transfer або "передача стану уявлення"). REST- ця архітектура передбачає методи або типи запитів HTTP для обміну інформацією із сервером.(GET, POST, PUT, DELETE)

REST

Стиль особливо зручний при реалізації різноманітних Single Page Application, які часто використовують спеціальні javascript-фреймворки типу Angular, React або Vue.js.Можна затвердити, що Web API є веб-службою, до якої звертаються інші програми. Крім того, ці програми можуть мати будь-яку технологію та платформу(веб-програми, мобільні або десктопні клієнти).

Іншими словами, можу зазначити, стиль REST - це збірник правил як розробнику створити код серверного додатка, щоб усі існуючі системи додатку легко обмінювалися даними і при потребі програму можна було масштабувати.

Також, слід зазначити, REST свого роду є узгодженим набором обмежень, які враховуються при проектуванні розподіленої гіпермедіа-системи.

Як правило результатом є спрощення архітектури та підвищення продуктивності. Також існує декілька обмежень для побудови розподілених RESTдодатків, які обов'язково потрібно підтримувати, якщо хоча б одне правило в сервіс-додатку порушено система відразу перестає вважатись система REST.

Обов'язковими обмеженнями вважають:

1. **Модель клієнт-сервер.**

Залежність від потреби інтерфейсу клієнта та потреб сервера, які реалізують збереження даних, підвищення перетягування коду клієнтського інтерфейсу, спрощення серверної частини покращує масштаб програмного продукту.

2. **Відсутність стану**

Протокол потребує, щоб взаємодія між користувачем та сервером у період між запитами клієнта, інформація про стан клієнта на сервері не зберігається. Всі запити клієнта повинні бути реалізовані так, щоб сервер получав всю інформацію яка йому потрібна до виконання запиту.

3. **Кешування**

Відповіді від сервера, повинні мати явне або неявне позначення як кешовані або некешовані, для того щоб запобігти отримання клієнтами неактуальних або невірних даних у відповідь на послідуючі запити.

4. **Однообразність інтерфейсу**.

Фундаментальним вимогою дизайну REST сервіса є унифікований інтерфейс. Уніфіковані інтерфейси дають змогу кожному із сервісів розвиватися незалежно від іншого.

5. **Шарова взаємодія з сервером.**

Клієнт може взаємодіяти з сервером або ж напряму або проміжним вузлом у зв'язку з різноманітною структурою мереж.Застосування проміжних серверів впливає на маштабування завдяки розподіленого кешування та балансування навантаження.

ASP.NET MVC Framework

Для клієнської UI використано MVC-структура розділення даних програми і

управління логікою програми за допомогою трьох окремих компонентів:

Модель-реагує на команди контролера та надає данні та методи, цим самим змінюючи свій стан.Модель ніяким чином не залежить від представлення, вона не потребує інформації як данні відобразились користувачеві.Також не має зв’язку із контролером,так як не має взаємнодії з користувачем.Але надає управління та доступ до данних. Так як, вище було вказано, що модель не залежить від візуального представлення, вона може налічувати декілька різних представлень для однієї «моделі»

Представлення-реагуючи на зміни моделі, відповідає за відображення користувачу змінених даних моделі. Проте, не оброблює данні які були введені користувачем.

Контролер-повідомляє модель про дії користувача та вказує коли нада внести зміни. Контролер свого роду посередник між користувачем та системою. Для реалізації використовує і модель і представлення.

Використовуючи MVC Framework потрібно мати розуміння поведінки вебдодатків. MVC являє собою один із підходів створення веб-дадатків з «чистим» кодом, його набагато легше розширювати та підтримувати.

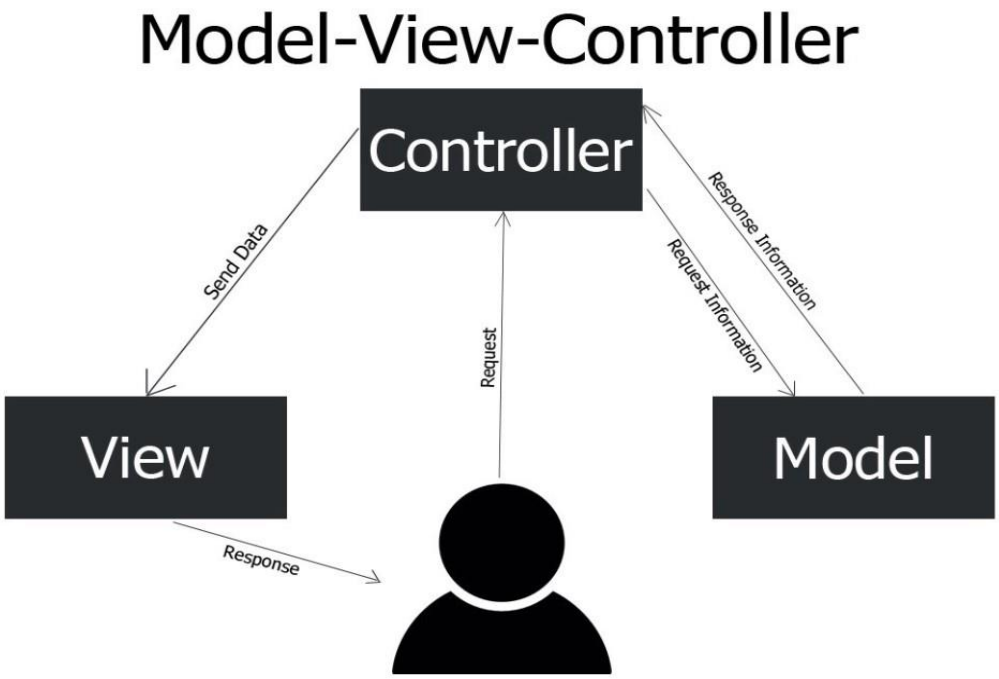


Рис. 3.2- Шаблон проектування MVC

Результатами дії контролера платформи ASP.NET MVC є такі типи:

1. Виевресулт — надає HTML та розмітку.
2. EmptyResult — результат не відображається.
3. Редиректресулт — реалізує периадрисацію на нову URL-адресу.
4. JsonResult —надає аннотацію об’єктів JavaScript результат, який використовується в програмах AJAX.
5. JavaScriptResult — відображає JS скріпт.
6. Контентресулт — представляє собою текстовий результат.
7. Филеконтентресулт — надає файл завантаження (з двійковим змістом).
8. FilePathResult — файл завантаження (вказується адреса).
9. Филестреамресулт — створює файл завантаження з файловим потоком данних.

JavaScript

У наш час створюючи веб-додаток важко не почути про JS.Тісно пов’язана з такими мовами веб-розробки як HTML та CSS. Так як за допомогою цієї об'єктноорієнтованої та динамічної мови програмування створюються сценарії вебсторінок, які дають можливість взаємодіяти з користувачем, виконувати асинхронний обмін данними,керувати браузером, та редагувати зовнішній вигляд сторінки.JS виділяють як динамічно типизовану скриптовану мову.

JavaScript є схожою до C мов програмування але має вагомі відмінності:

- Функції як об’єкти першого класу.
- Функції з можливістю динамічні зміни типів завдяки механізму прототипів.
- Атоматичне приведення до типу
- Обробка винятків
- Анонімні та стрілочні функції
- Автоматичне керування пом’ятю компьютера під час виконання програмного коду.

Також використовуючи JS можна втілювати такі архітектурні можливості як:

авто керування пам’яттю, обєкти першого класу, прототипне наслідування,слабка та динамічна типизація.

JavaScript є інтерпретованою мовою, без строгої типізації,саме це є основною проблемою JS.

JavaScript відрізняється від стандартних мов ООП, але має деякі властивості які використовують у мовах ООП.Функціональні здібності які надають мові більшої гунучкості:

- Каррінг- спосіб обчислення функції з багатьма оргументами, шляхом перетворення її в послідовність функцій одного аргумента.

• Анонімні функції- функція яка визначається без вказування пов’язаного з нею ідентифікатора.

• Замикання- підпрограма яка може виконуватись в середовищах з однією або декількома зв’язними змінними.

• Функції як списки.

JS містить в собі безліч вбудованих об’єктів, які поділені на підгрупи.(рефлекційні, ключові, фундаментальні тощо).Взявши всі ці функціональні здібності мови можна зазначити що JS, на данний момент, є найпопулярнішою мовою у світі це зазначено навіть в статистиці.

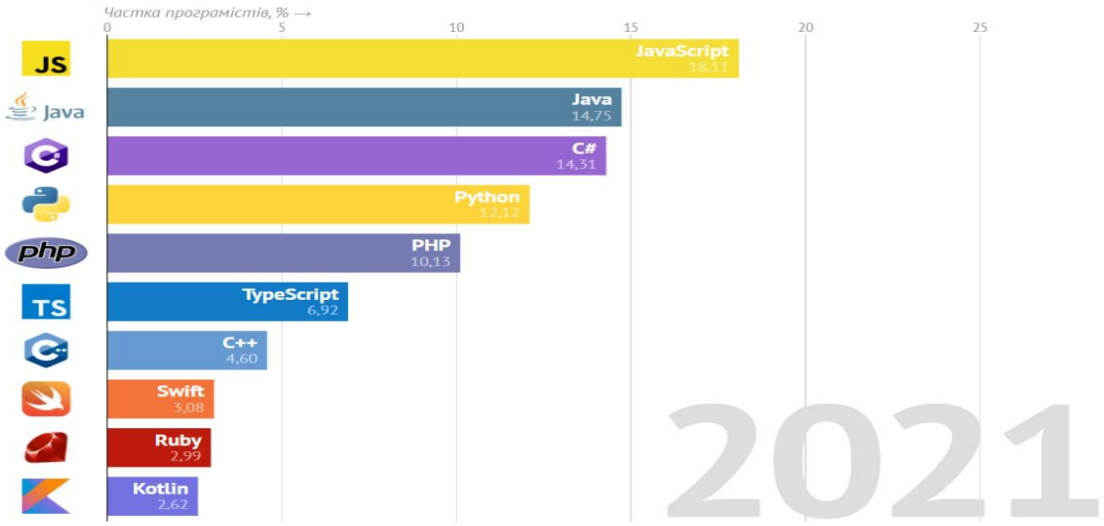


Рис.3.3- Рейтинг мов програмування за 2021 рік

Підсумовуючи можу зазначити, JavaScript – одна з найбільш сумісних мов для програмістів почтківців. На перший погляд можна зазначити,що мова достатньо проста: програма на JS являє собою текст, який можна писати неважливо в якому текстовому редакторі.Але слід зазначити, JS включає такі важливі для програмування речі, як структури, об'єктно-орієнтовану модель, алгоритми.

VIE.JS

JS реалізує безліч фреймфорків, один із фреймворків ми використовували у своїй дипломній роботі. Це фреймфорк Vie.js.

Потреба в даному фреймворкі була що уникнути написання повторюваного HTML.

Vue.js реалізує синтаксис шаблонів або прямо прописуючи рендерингові функції використовуючи JSX. Це потрібно, щоб зробити просто заміну шаблону на рендерингову функцію. Рендерингова функція надає можливості використання патернів базованих на компонентах.

Фрейморк використовує шляхи для забезпечення ефектів переходу, коли елемент додають, оновлюють або видаляють з об’єктної моделі документа. Існують такі шляхи:

- для CSS переходів та анімацій використовують автоматичне примінення класів
- для CSS анімацій використовується інтегрування сторонніх бібліотек
- використання JSдля прямих маніпуляцій з об’єктами моделі документів(DOM) під час переходів
- інтеграція сторонніх JS бібліотек анімацій.

Також, хочу зазначити що Vue використовує синтаксис шаблонів основаних на HTML, що дозволяє фіксовано зв'язувати рендеринг об’єктів моделі з екземплярами даних в Vue. Всі Vue шаблони підчиняються HTML.

Bootstrap

Для використання стилів ми користувались набором інструментів з відкритим кодом Bootstrap.

Його основне призначення надання інструментів для створення веб-сайтів та веб-додатків які містять шаблони CSS та HTML. Данний фреймворк реалізовує загалом клієнський інтерфейс. До появи фреймворка Bootstrap

розробникам приходилось користуватись бібліотеками що призводило до ускладнювання супроводу та суперечностей в програмі. Він сумісний з усіма останніми оновленнями браузерів таких як Google Chrome,Opera, Safari тощо. Bootstrap містить модульну структуру і складається в основному з наборів таблиць динамічної мови стилів LESS, які реалізують різноманітні компоненти цього набору інструментів.

При експлуатації Bootstrap можна використовувати такі інструменти:

1. Таблиці (table) — інструмент оформлення та забезпечення сортування таблиць
2. Форми (form) — класи щоб оформлювати форми та інколи події.
3. Типографіка (typography) —визначення класу для шрифту та його опис/
4. Мультимедіа (media) — інструмент керування відео та зображеннями
5. Навігація (nav, navbar) — набір класів для реалізації вкладок,меню, сторінок, панелей навігації.
6. Сповіщення (alert) — клас який надє керування оформленням діалогових вікон, спливаючих вікон а також підсказок
7. Іконочний шрифт (icon font) — склад набору іконок являє собою до 500 компонентів- це іконки у вигляді шрифту.
8. Сітки (grid) — заздалегіть готові до використання колонки

Шаблони (template) — шаблони сторінок які заздалегіть адаптовані та фіксовані для рішення задач

ВИСНОВОК ДО РОЗДІЛУ 3

Зазначивши та описавши усі технології, які використовувались при розробці програмного застосунку “Car Service”, хочу виділити середовище розробки Visual Studio яке значно оптимізувало час розробки програмного продукту за рахунок логічної побудови програмного коду, легкості пов’зуваня класів.

Токож хочув виділити Entity Framework який пришвидшив налаштування звязку програми з базою даних, при передачі там внесенні даних з програми модуль відпрацьовує без помилок. Розробивши продук за допомогою багаторівневої архітектури я отримав можливість розділення програми на модулі що знано пришвидшило розробку та полегшило знаходження проблем в коді програмного продукту.

Особливістю є те що програмний продукт реалізований за допомогою MVC Framework, реалізацію модулів представлення і контроллера було розроблено стандартно, а модуль був оптимізавоний за рахунок WebAPI що значно полегшело роботу та розуміння програми. Framework MVC був вибраний для швидкої обробки данних та її представлення у web-вигляді.

РОЗДІЛ 4. ПРОЦЕС РОЗРОБКИ ТА ПРЕДСТАВЛЕННЯ ВЕБ-ЗАСТОСУНКУ

Вся логіка веб-застосунку «Car Service» відбувалась за допомогою мови C#.Ця мова дає можливість для об'єктів взаємодіяти один з одним. Вся програма будувалась завдяки редактору коду Visual Studio.Програма була розділена на об’єкти та класи. Свого роду клас це тип даних на мові C#, який надає конструкцію що об’єднує методи, властивості та поля.

Кожен клас відповідає за певну функціональність програми, він описує об’єкт, а об’єкт як правило представляє екземпляр класу. При розробці був використаний Framework MVC. Це означає що програма містить окремо модулі Models, Views, Controllers. Модуль Views. Містить файли представлення, тобто код веб представлення.(HTMS,CSS,)А також JS, для реалізації звернення до класу контролера щоб отпримати данні, а потім відобразити їх.

Для більш детального розуміння роби програми була розроблена UML діаграма класів.

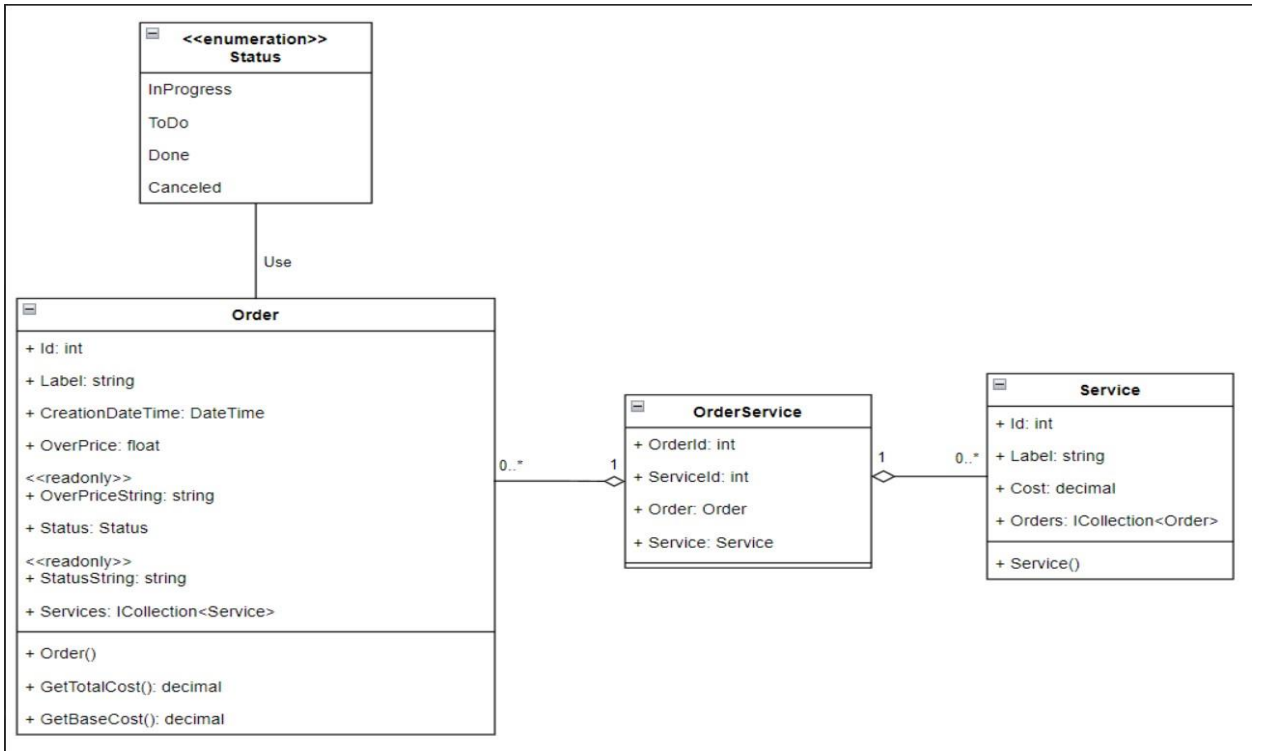
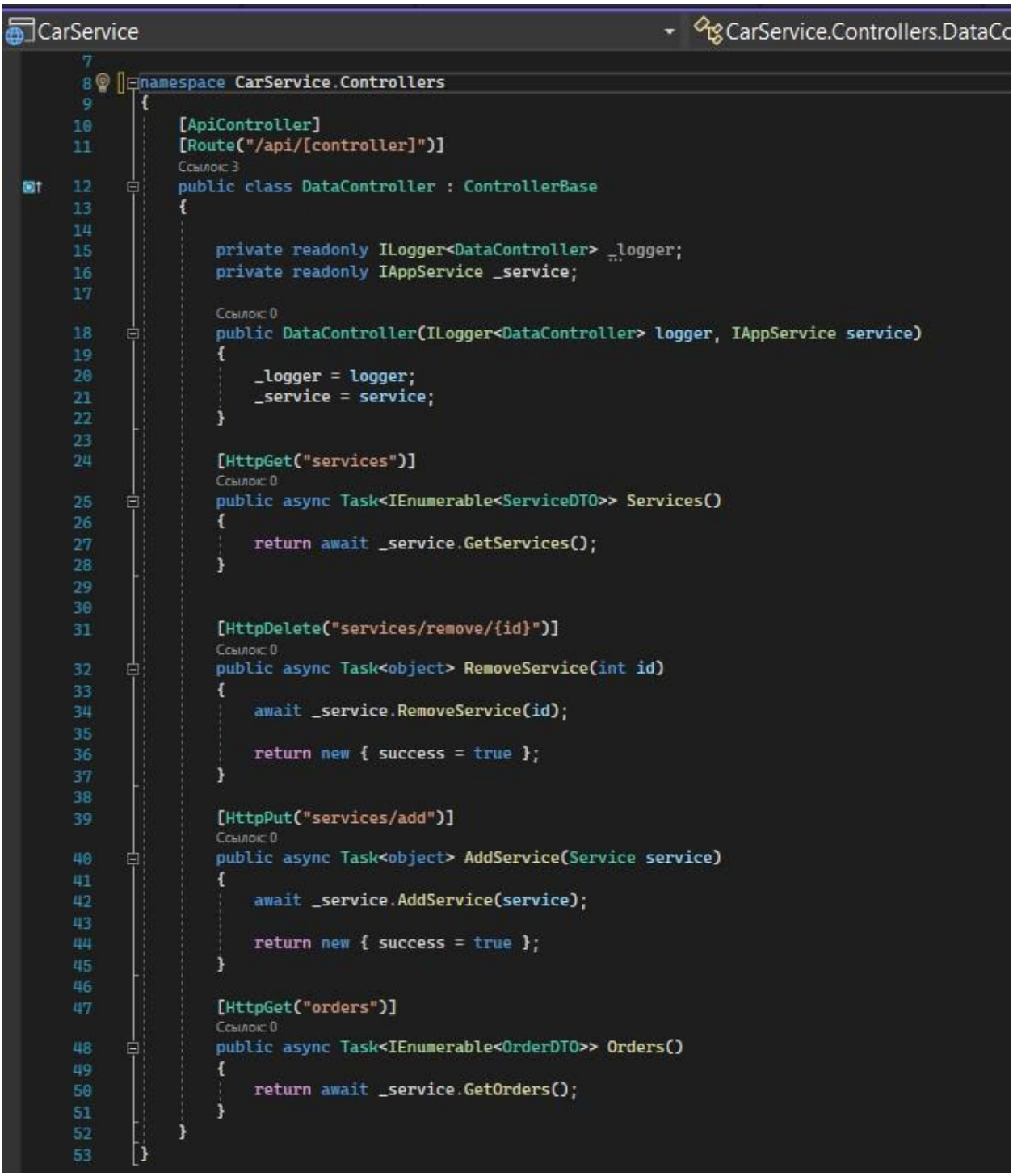


Рис.4.1-Діаграма класів

Модуль Controllers

В контроллерах реалізовано два класи DataController.cs та HomeController.cs.

DataController.cs даний клас реалізує WebApi, заміняючи Моделі MVC Framework. Завдяки цьому реалізовано отримання та оновлення данних.




```
namespace CarService.Controllers
{
    [ApiController]
    [Route("/api/[controller]")]
    Ссылок: 3
    public class DataController : ControllerBase
    {

        private readonly ILogger<DataController> _logger;
        private readonly IAppService _service;

        Ссылок: 0
        public DataController(ILogger<DataController> logger, IAppService service)
        {
            _logger = logger;
            _service = service;
        }

        [HttpGet("services")]
        Ссылок: 0
        public async Task<IEnumerable<ServiceDTO>> Services()
        {
            return await _service.GetServices();
        }


        [HttpDelete("services/remove/{id}")]
        Ссылок: 0
        public async Task<object> RemoveService(int id)
        {
            await _service.RemoveService(id);

            return new { success = true };
        }

        [HttpPut("services/add")]
        Ссылок: 0
        public async Task<object> AddService(Service service)
        {
            await _service.AddService(service);

            return new { success = true };
        }

        [HttpGet("orders")]
        Ссылок: 0
        public async Task<IEnumerable<OrderDTO>> Orders()
        {
            return await _service.GetOrders();
        }
    }
}
```

DataController.cs

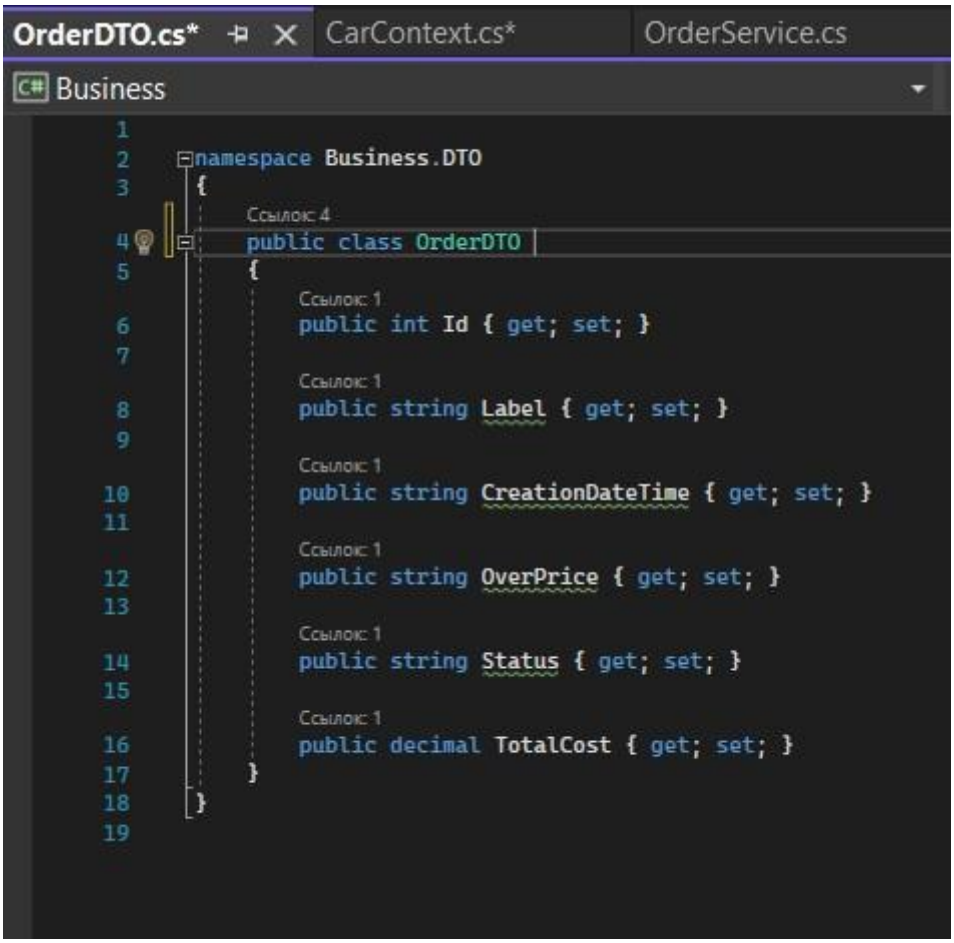
```
@{
     Layout = "~/Views/Shared/_Layout.cshtml";
}
<div>
    <div class="container">
      <h2>Види послуг</h2>
      <div style="width: 600px" class="d-flex justify-content-center">
          <table class="table table-bordered table-editable">
            <thead>
              <tr>
                <th>Назва послуги</th>
                <th>Ціна</th>
                <th></th>
              </tr>
            </thead>
            <tbody id="tableBody"></tbody>
          </table>
      </div>
    </div>
</div>


<script type="text/javascript">
    function getServices(){
        return fetch('/api/Data/services', {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json'
                }
            })
            .then(response => response.json())
            .then(function(response) {
                return response;
            })
            .catch(function(error) {
                console.log(error);
            });
    }

    function validateInput() {
        if (IsNewServiceValid()) {
            $('#button_add').prop('disabled', false);
        }
        else {
            $('#button_add').prop('disabled', true);
        }
    }
```

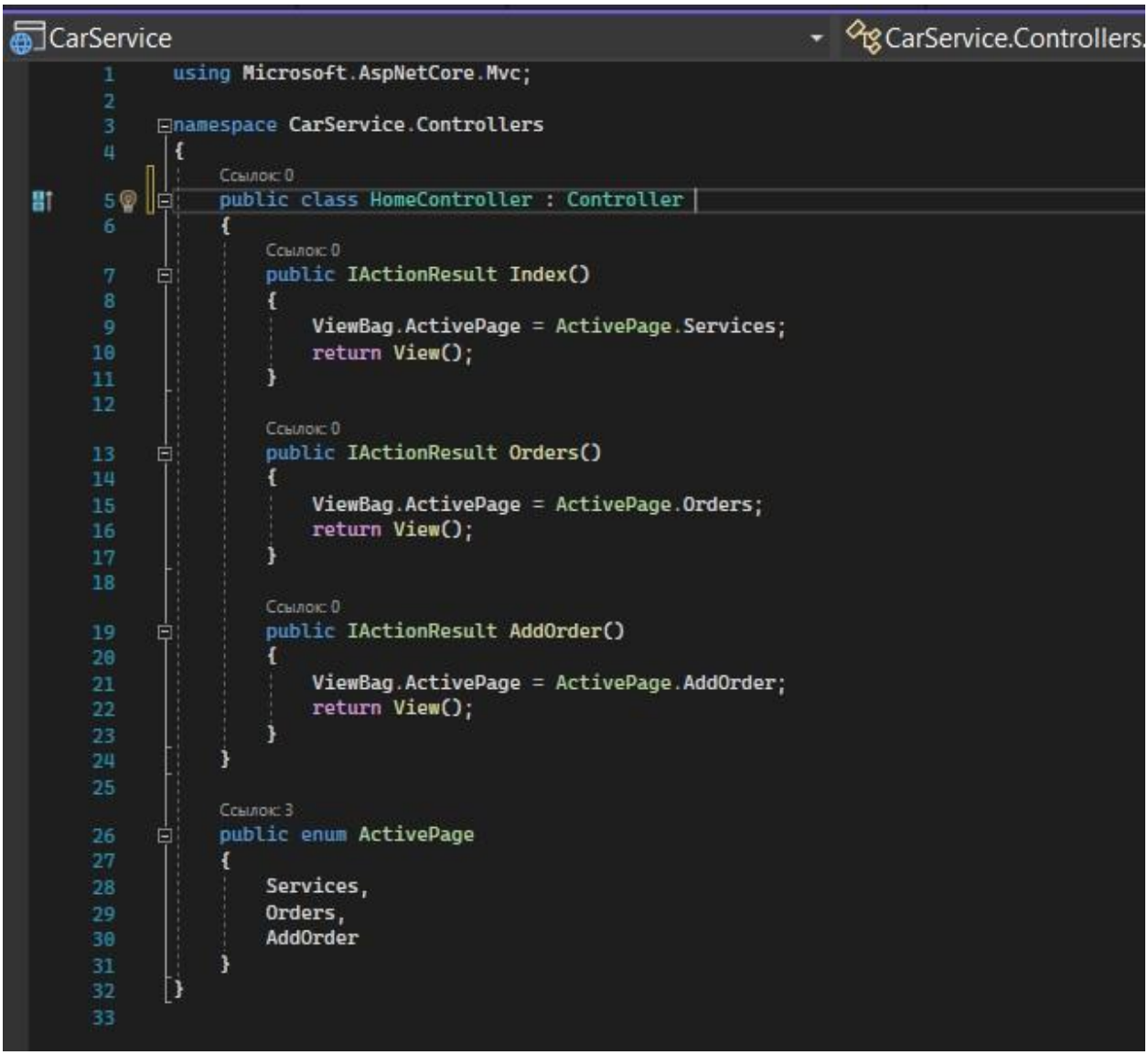
DataController.cs(2)

Для передачі та обробки отриманих даних на WebApi був реалізований клас OrderDTO.cs



OrderDTO.cs

Клас HomeController.cs. відповідає за відображення та керування даними в HTML.



HomeController.cs

Модуль Models містить три класи Order.cs, OrderService.cs, Service.cs. У всіх класа моделі Models реалізовані шаблони для веб-заповнення данних користувачів та синхронизується з WebApi, також відображає те що знаходиться в базі даних.

DataAccess carserviceapp.Models.St

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace carserviceapp.Models
{
    Ссылок: 3
    public enum Status
    {
        [Description("В процесі")]
        InProgress,

        [Description("Заплановано")]
        ToDo,

        [Description("Виконано")]
        Done
    }

    Ссылок: 9
    public class Order
    {
        [Key]
        Ссылок: 3
        public int Id { get; set; }

        Ссылок: 1
        public string Label { get; set; }

        Ссылок: 3
        public ICollection<Service> Services { get; set; }

        Ссылок: 4
        public DateTime CreationDateTime { get; set; }

        Ссылок: 2
        public float OverPrice { get; set; }

        Ссылок: 1
        public string OverPriceString => (OverPrice * 100) + "%";

        Ссылок: 3
        public Status Status { get; set; }

        Ссылок: 0
        public string StatusString => Status.GetDescription();

        Ссылок: 1
        public decimal GetTotalCost()
        {
            decimal sum = Services.DefaultIfEmpty().Sum(s => s.Cost);
            return (sum + (sum * (decimal)OverPrice));
        }
    }
}
```

Представлення класу Order.cs

Даний клас реалізує раніше описаний Entity Framework з базою даних.

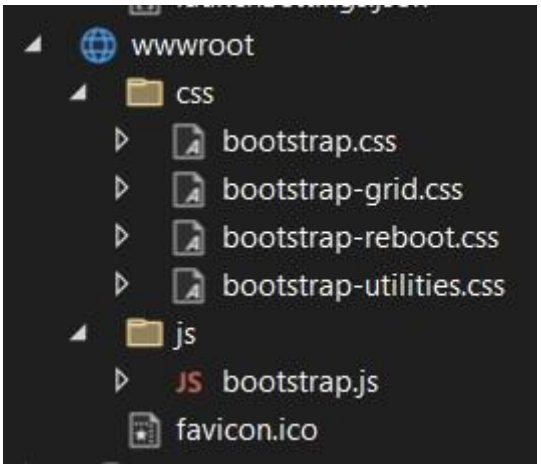
Фреймфорк являє собою свого роду адаптер, завдяки якому відбувається передача інформації з класів Model в базу даних. Представлення класу CarContext.cs.

DataAccess — DataAccess.CarContext

```
1   using carserviceapp.Models;
2   using Microsoft.EntityFrameworkCore;
3
4   namespace DataAccess
5   {
        Ссылок: 7
6       public class CarContext : DbContext
7       {
            Ссылок: 4
8           public DbSet<Order> Orders { get; set; }
9
            Ссылок: 3
10          public DbSet<Service> Services { get; set; }
11
12          //public DbSet<OrderService> OrderServices { get; set; }
13
            Ссылок: 0
14          protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
15          {
16              optionsBuilder.UseSqlite("Data Source=CarServiceDB.db;");
17          }
18
            Ссылок: 0
19          protected override void OnModelCreating(ModelBuilder modelBuilder)
20          {
21              modelBuilder.Entity<Order>().ToTable("Orders");
22              modelBuilder.Entity<Service>().ToTable("Services");
23
24              //modelBuilder.Entity<OrderService>()
25              //    .HasKey(item => new { item.OrderId, item.ServiceId });
26
27              modelBuilder
28                  .Entity<Order>()
29                  .HasMany(p => p.Services)
30                  .WithMany(p => p.Orders)
31                  .UsingEntity(j => j.ToTable("OrderServices"));
32          }
33      }
34  }
35
```

CarContext.cs

Вся логіка веб представлення зберігається в модулі wwwroot там описаний стиль Bootstrap.



Представленя web-інтерфейсу “Car Service”

Для повного представлення про функціональні можливості програмного продукту було розроблено UML діаграму варіантів використання.

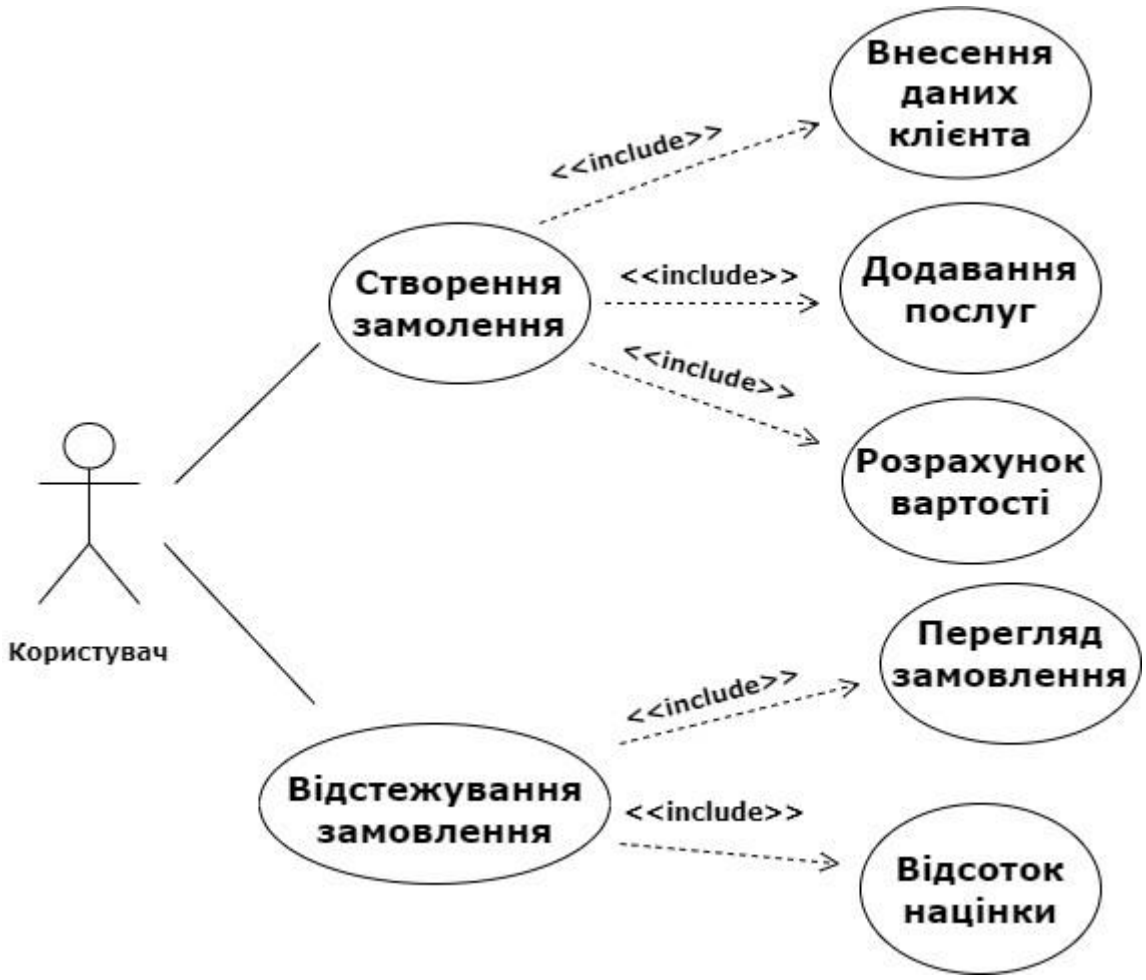


Рис.4.2 - Діаграма варіантів використання

Веб представлення початкової сторінки програми Car Service.

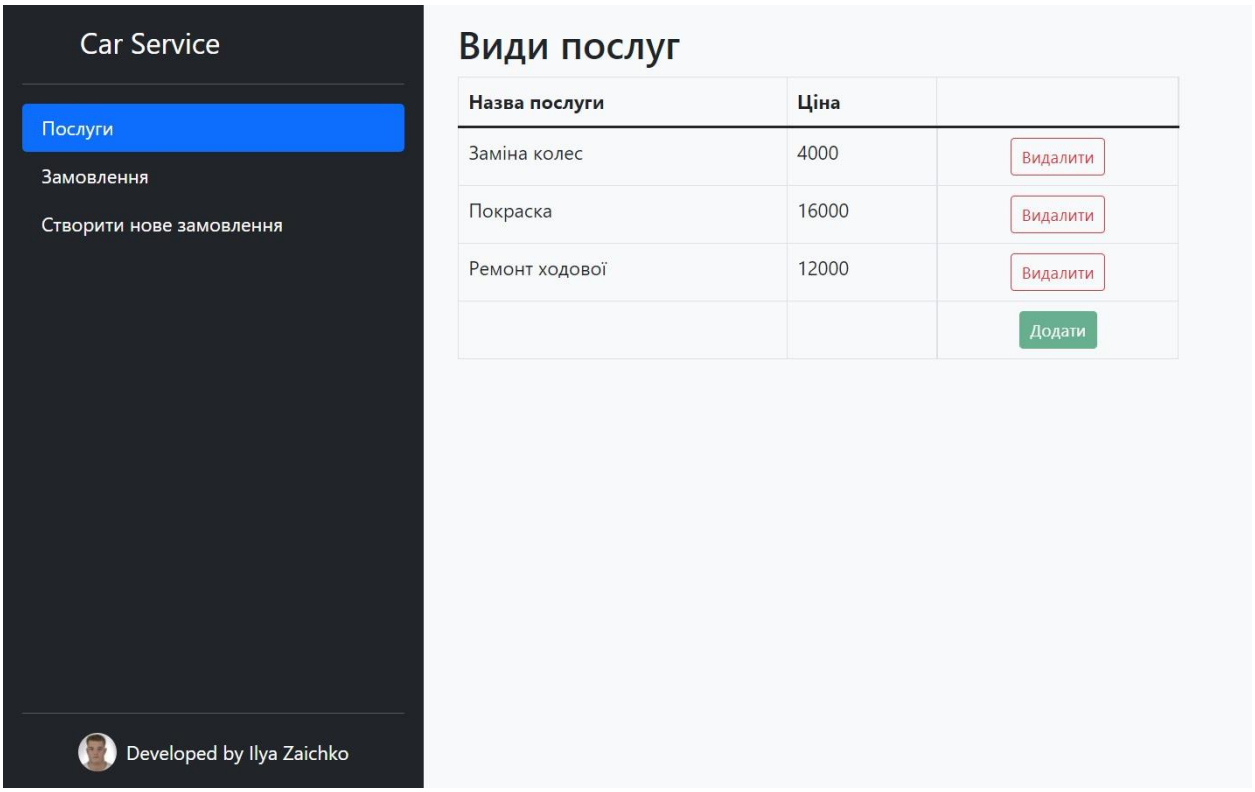


Рис.4.3- Полсуги

На скріншоті нам представлений інтерфейс вкладки «Послуги» який реалізує додавання послуги та ціни. За її допомогою ми відразу визначаємо основні потреби користувача та ціну яку споживач сплатить після обслуговування свого автомобіля. Також є функція видалення послуги якщо клієнт передумав виконання певної послуги або ж коли СТО не має технічної можливості надати обслуговування.

На наступному малюнку нам представлено вкладку «Замовлення». Вона зберігає в собі вже існуючі замовлення їх стан. На даній вкладці ми можемо знайти замовлення по назві, після можемо спостерігати інформацію про статус замовлення, їх існує три – це в процесі, заплановано та виконано. Завдяки цьому

ми визначаємо приблизний час очікування. Наступне що може спостерігати або змінювати користувач це націнка. Націнка визначається до та після виконаних робіт. Відсоток націнки рахується від часу виконання та витрачених матеріалів.

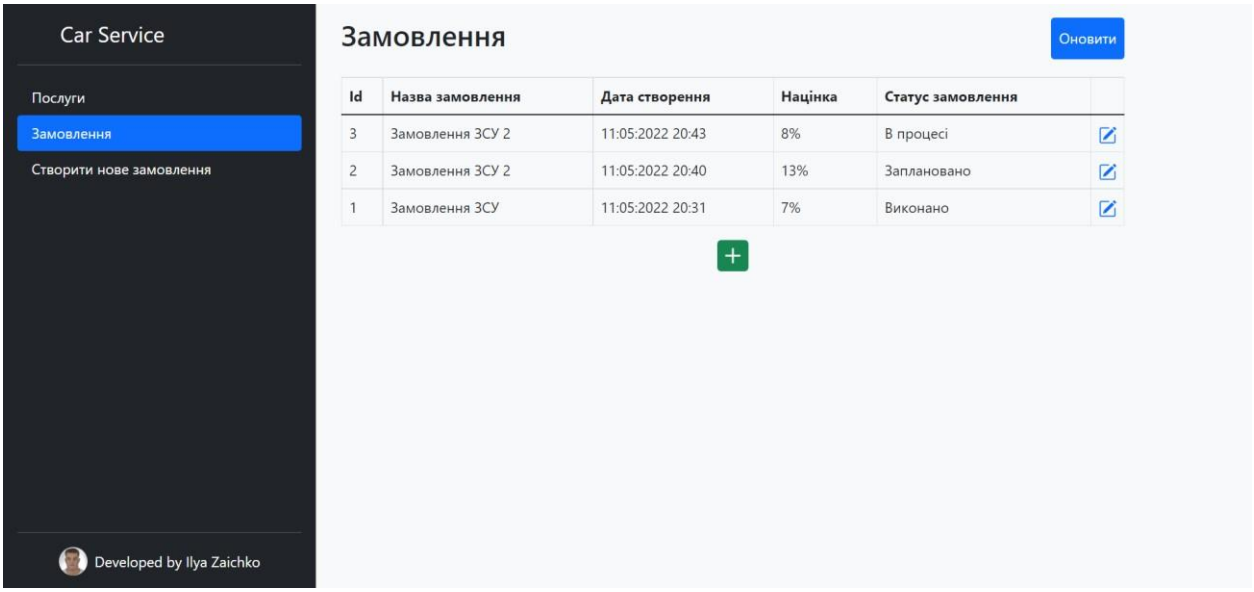


Рис.4.4- Вкладка «замовлення»

Програма фіксує час створення замовлення для коректного ведення обліку СТО. Є можливість оновити інформацію за допомогою клавіши «Оновити», а також створити нове замовлення. Підтримується редашування кожного замовлення на будь-якому етапі роботи.

Створення замовлення відбувається за допомогою внесення данних користувача, визначення вимог обслуговування в вкладці «Послуги». Кінцевим результатом створення є вдображення замовлення в вкладці «Замовлення».

ВИСНОВОК ДО РОЗДІЛУ 4

В даному розділі було представенно етапи та моделі розробки програмного забезпечення “CarServise”, весь функціонал створювався за принципом розробки багаторівневої архітектури. Були представлені основні

класи програми які завдяки даній архітектурі створювалися всі окремо хоч і всі взаємопов’язані між собою.

Також був представлений клієнтський інтерфейс програми який висвітлює дві основних частини програми. Першою є додавання послуг які потребує користувач та реєстрація самого користувача на етапі створювання замовлення. Іншою є представлення замовлення яке вже зареєстровано в системі, можна відслідковувати етапи роботи замовлення також вказати націнку згідно витрачених засобів та часу.

На кінець хочу зазначити що в даний розділ представляє собою реалізацію та інтерфейс програми яка реалізує всі основні технічні завдання моєї роботи.

ВИСНОВОК

Проведено дослідження галузі обслуговування автомобілів для конкретного поставлення цілей своєї роботи а також розуміння проблематики даної галузі, її маштаби, її навантаження та процеси. Створено порівняльну таблицю програм аналогів та проведено аналіз згідно з даними які були отримані на етапі оцінки та дослідження предметної галузі

Вирішено проблему обліку та замовлень клієнтів, а також автоматизовано процес створення нового замовлення.

В результаті реалізовано архітектура та розробка програмного продукту, який допоможе працівникам сфери обслуговування автомобілів оброблювати великий потік інформації та розширювати з кожним днем свою клієнтську базу.

Виконання системних задач які потребував програмний продукт було оптимізовано за рахунок MVC Framework. Тому вся розробка функціоналу базується на основах ООП програмуваня.

На сам кінець хотів би зазначити що,виконуючи данну дипломну роботу, я кожного дня отримував нові знання та навички з розробки та побудови процесу реалізації програмного продукту а також вдосконалював вже набуті знання. Результатом є досвід у розробці web-застосунків.

ПЕРЕЛІК КОРИСНИХ ПОСИЛАНЬ

1. Основи ООП. Джефрі Ріхтер, CLR via C#, 4-е видання 2014 by Фримен А. «ASP.NET MVC 4 с примерами на C# 5.0» Дата звернення: 17.04.2022
2. Основи технічного сервісу транспортних засобів : навч. посіб. / Є. Ю. Форнальчик, Р. Я. Качмар Дата звернення: 19.04.2022
3. Onion Architecture In ASP.NET Core MVC. [Електронний ресурс] – Режим доступу:https://www.c-sharpcorner.com/article/onion-architecture-in-asp-netcore-mvc/ Дата звернення: 17.04.2022
4. What is Entity Framework?. [Електронний ресурс] – Режим доступу: https://www.entityframeworktutorial.net/what-is-entityframework.aspx
5. Getting started with ASP.NET MVC 5 [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/mvc/overview/gettingstarted/introduction/getting-started Дата звернення: 17.04.2022
6. ООП в JavaScript [Електронний ресурс] – Режим доступу: https://frontendstuff.com/blog/object-oriented-programming/ Дата звернення: 09.04.2022
7. Лекція 22. Bootstrap [Електронний ресурс] – Режим доступу: https://etk.lntu.edu.ua/mod/page/view.php?id=4135 Дата звернення: 13.04.2022
8. Рівні ієрархії сучасної АСУТП [Електронний ресурс] – Режим доступу: https://studopedia.com.ua/1_378896_bagatorivneva-arhitektura.html
9. Overview to ASP.NET Core [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnetcore?view=aspnetcore-6.0 Дата звернення: 17.04.2022

10. База SQLite и основы SQL. [Електронний ресурс] – Режим доступу: https://maxsite.org/page/sqlite8 Дата звернення: 17.04.2022
11. Розробка додатків засобами мови програмування C#.[Електронний ресурс] –
Режим доступу: https://www.researchgate.net/publication/354860614_Rozrobka_dodatkiv_zasobami_movi_programuvanna_C Дата звернення: 17.05.2022
12. Общие сведения о платформе .NET [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview Дата звернення: 17.04.2022
13. Компиляция C# и XAML в JavaScript [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/articles/csharp/1.php Дата звернення: 17.04.2022
14. Фаулер Скотт До. UML в короткому викладі. Застосування стандартної мови об'єктного моделювання: Пер. з англ. – М.:Мир, 1999. – 89107 с. Дата звернення: 17.04.2022
15. ПРОФЕСІЙНА ПРАКТИКА ПРОГРАМНОЇ ІНЖЕНЕРІЇ. Лабораторний практикум / уклад. С. В. Поперешняк – К.: Вид-во «Друк», 2019. – 43-47 с.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

“Розробка web-сервісу для онлайн обслуговування автомобілі мовою C#”

Виконав студеня 4 курсу
Групи ПД-43
Заїчко Ілля Олегович
Керівник роботи
доцент кафедри ІПЗ
Поперешняк С.В

Київ – 2022

Актуальність програми

Вивчивши питання проблематику сфери обслуговування автомобілів я створив рішення яке полегшить облік користувачів. Одним із таких рішень є те що, автоматизовано створення клієнтів та відстежування їх замолень, кожна надана послуга проведена через систему та записана у програмі для СТО. Спочатку ви вказуєте послуги, які надає організація та відповідний тарифний план.

АНАЛОГИ

Показник	РемОнлайн	PRO.CTO.UA	TQM Systems	YuKoSoft
Платформи	Web	Android, iOs, Web	Web	Windows
Вартість використання програми	-	-	-	**Пробний період**
Облік замовлень	+	-	+	+
Онлайн замовлення	+	+	+	+
Онлайн розрахунок	-	-	+	+
Планувальник	-	+	+	+
Онлайн база клієнтів	-	-	+	+
Ведення карти автомобіля	-	-	+	+
Використання «замовлення-наряд»	-	-	+	+
Контроль складських залишків.	-	-	+	-
Підтримка онлайн камер	-	+	+	-

3

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** створити програмний продукт щоб атоматизувати ведення обліку та полегшити процеси станцій технічного обслуговування.
- **Об'єкт дослідження** процеси станцій технічного обслуговування та їх особливості.
- **Предмет дослідження** web-додаток на основі існуючих програм аналогів та потреб галузі обслуговування автомобілів.

4

ТЕХНІЧНІ ЗАВДАННЯ

1. Автоматизація обліку послуг які надає станція технічного обслуговування. Що зробити?
2. Архітектура програмного продукту реалізує обслуговування автомобілів, оброблення великого потоку інформації та розширювання клієнтської бази.
3. Впровадження багаторівневої архітектури, з подальшим розділенням розробки web-застосунку на модулі.

5

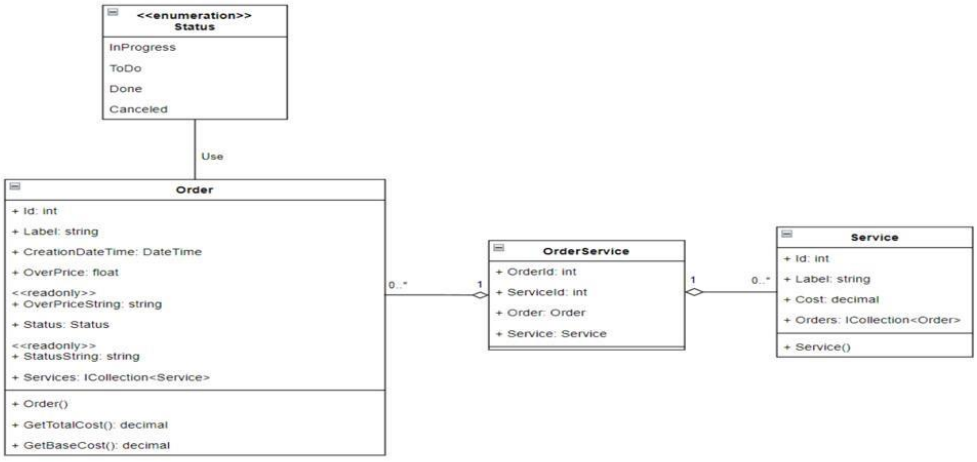
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

ДІАГРАМА варіантів ВИКОРИСТАННЯ

7

ДІАГРАМА КЛАСІВ



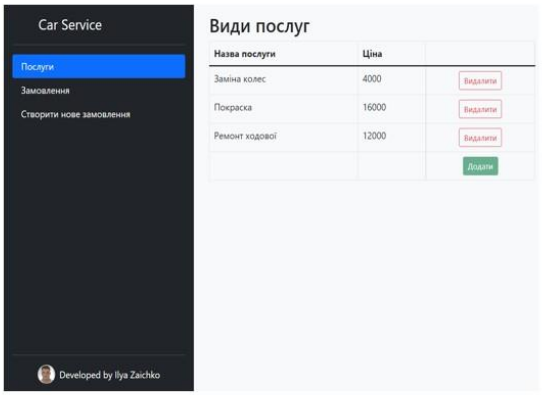
8

Представлення архітектури

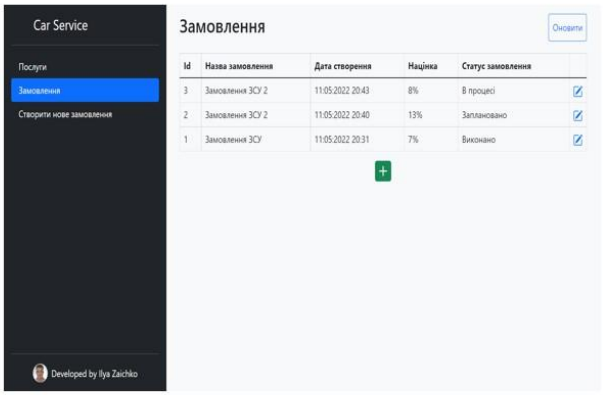
9

Інтерфейс програми

Додавання послуг



Статус замовлень



10

ВИСНОВКИ

1. Проведено дослідження галузі обслуговування автомобілів для конкретного поставлення цілей своєї роботи а також розуміння проблематики даної галузі, її маштаби, її навантаження та процеси.
2. Вирішено проблему обліку та замлень клієнтів, а також автоматизовано процес створення нового замовлення.
3. В результаті реалізована архітектура та розробка програмного продукту, який допоможе працівникам сфери обслуговування автомобілів оброблювати великий потік інформації та розширювати з кожним днем свою клієнтську базу.Що зроблено?

11

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Заїчко І.О. Життєвий цикл моделей Entity Framework / С.В. Поперешняк, І.О. Заїчко // Застосування програмного забезпечення в інфокомунікаціях: Матеріали науково-технічної конференції. Збірник тез. 20.04.2022, ДУТ, м. Київ — К.: ДУТ, 2022. — С. 43-44.

12

ДЯКУЮ ЗА УВАГУ!