Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Пояснювальна записка

до бакалаврської дипломної роботи

на тему: «Веб-сервіс надання побутових послуг»

Виконав: студент 2 курсу, групи 1КІ-20мс
123 – «Комп'ютерна інженерія»
Кошельник В. О.
Керівник роботи к.т.н., доц. Колесник І.С.
Рецензент_ д.т.н.,проф._Лужецький В. А.

Допущено до захисту
д.т.н., проф. Азаров О.Д.

"15" 06 2022 р.

Вінниця 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень — бакалавр

Спеціальність — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
проф., д.т.н. О.Д.Азаров
« 9 » лютого 2022 р.

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

студент Кошельнику Вадиму Олександровичу

1 Тема бакалаврської дипломної роботи «Веб-сервіс надання побутових послуг», керівник роботи Колесник Ірина Сергіївна, затверджені наказом вищого навчального закладу від 24.03.2022 року № 66

2 Строк подання студентом роботи: 13.06.2022

3 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз застосування сучасних веб-сервісів для надання побутових послуг, вибір засобів для розробки веб-сервісу, опис веб-сервісу, висновки, перелік джерел посилання.

4 Дата видачі завдання: 10.02.2022

5 Календарний план наведений в таблиці 1.

Таблиця 1 — Календарний план

№ з/п	Назва етапів виконання бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	11.02.22	вик.
2	Аналіз сучасних веб-сервісів для надання побутових послуг	12.02-28.02.22	вик.
3	Вибір засобів для розробки веб-сервісу для надання побутових послуг	01.03-12.03.22	вик.
4	Проектування веб-сервісу	13.04-17.04.22	вик.
5	Підготовка матеріалів та опис розробки	20.03-22.03.22	вик.
6	Оформлення пояснювальної записки та ілюстративного матеріалу	23.04-29.04.22	вик.
7	Аналіз виконання роботи, висновки, додатки	30.04-17.05.22	вик.
8	Перевірка якості виконання бакалаврського проекту та усунення недоліків	18.05.22	вик.

Студент _____ Кошельник В.О.

Керівник роботи _____ Колесник І.С.

АНОТАЦІЯ

Дана бакалаврська дипломна робота присвячений розробці веб-сервісу, що об'єднує замовників, яким потрібно створити будь-яку роботу, і компетентних виконавців, які шукають підробіток або додатковий заробіток..

Користувач має змогу зареєструватись як замовник послуг або спеціаліст і в залежності від цього буде надаватись певний функціонал, що можна використовувати. Веб-додаток розроблено за допомогою двох мов програмування: JavaScript та C#. З використанням технологій .NET Core, React і використанням сторонніх бібліотек.

Ключові слова: веб-сервісу, замовник, виконавець .NET Core, React, JavaScript, C#.

ABSTRACT

This bachelor's thesis is dedicated to the development of a web service that unites customers who need to create any job, and competent performers who are looking for part-time or extra income.

The user has the opportunity to register as a customer or specialist and, depending on this, will be provided with certain functionality that can be used. The web application is developed using two programming languages: JavaScript and C #. Using .NET Core, React and third-party librarie

Keywords: web service, customer, performer .NET Core, React, JavaScript, C #.

ЗМІСТ

ВСТУП 8

1 ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ІСНУЮЧИХ ВЕБ-СЕРВІСІВ В ДАНОМУ НАПРЯМКУ 11

1.1 Постановка задачі 11

1.2 Огляд існуючих аналогів 11

1.2.1 Work.ua 14

1.2.2 Upwork 14

1.2.3 Kabanchik.ua 15

1.2.4 Rabotniki.UA 16

1.2.5 Freelancehunt.com 17

1.2.6 OLX 18

1.2.7 Robota.ua 19

1.2.8 Послуги.ua 20

1.2.9 Flagma 20

2 ОГЛЯД ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ 21

2.1 Мова програмування C# 21

2.2 .NET-платформа 23

2.3 База даних SQL 24

2.4 Мова програмування JavaScript 24

2.5 Додаткові інструменти для frontend 27

2.7 Програмні інструменти (середовище розробки) 31

3 РОЗРОБКА ВЕБ–СЕРВІСУ 34

3.1 Розбір та аналіз бази даних 34

3.2 Розробка бекенд частини 36

3.3 Робота з запитом в проекті 42

3.4 Процес авторизації / реєстрації 44

3.5 Функціонал веб-додатку 45

ВИСНОВКИ 58

					08-23.БДР.035.00.000 ПЗ		
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Кошельник В.О.*			Веб-сервіс надання побутових послуг	*Літ.*	*Арк.* 6 / *Аркушів* 80
Перевір.		*Колесник І.С.*					
Реценз.		*Лужецький В. А.*			Пояснювальна записка	*ВНТУ, гр. 1КІ-20мс*	
Н. Контр.		*Швець С. І.*					
Затверд.		*Колесник І.С.*					

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 59
ДОДАТОК А Технічне завдання 61
ДОДАТОК Б Лістинг класу Header.js 64
ДОДАТОК В Лістинг класу OrderRepository 70
ДОДАТОК Г Лістинг інтерфейсу IOrderRepository 78
ДОДАТОК Д Структура бази даних 79
ДОДАТОК Е Протокол перевірки кваліфікаційної роботи на наявість текстових запозичень 80

ВСТУП

Особливе місце в структурі управління займає підприємництво у сфері побутового обслуговування населення. У всьому світі, в тому числі і в Україні, сфера побутових послуг — організована діяльність суб'єктів господарювання, пов'язана з наданням побутових послуг — постійно розширюється. Основним продуктом у сфері побутового обслуговування є побутове обслуговування населення [2].

Побутова послуга — вид діяльності суб'єктів господарювання діяльність, пов'язана із задоволенням конкретного домогосподарства потреби окремого клієнта [3].

Відповідно до правил про побутове обслуговування населення, затверджених постановою Кабінету Міністрів України від 4 червня 1999 р. № 974, усі громадяни мають рівні права на задоволення своїх потреб у цьому питанні. Мати перевагу над будь-яким із них, і не допускається пряме чи непряме обмеження прав.

З появою на Українському ринку аналога американського сервісу «Taskrabbit», усе більше людей виявляють інтерес до подібної моделі бізнесу. Фактично даний сервіс є новинкою, з майданчиком завжди актуальних міні тендерів, за які «борються», зареєстровані раніше користувачі для подальшої реалізації.

Відразу відкинемо тендерні майданчики, що створені для державних закупівель та великого бізнесу. Усі ці майданчики відносяться до окремої категорії та ніяк не зацікавлять людину, яка хоче забити цвях або покласти у себе вдома плитку чи ще щось зв’язано з домашньою роботою. Та й у цілому буде дивно виглядати тендер з доставки піци на сайті «Прозорро», поруч із замовленням на ремонт міської лікарні.

Аналізуючи ринок подібних проектів, можна виявити десятки (якщо не сотні) спроб зробити подібні сайти, проте переважна більшість із них потерпіла краху. При цьому їх творці чесно віддали багато сил та часу на

розробку та безрезультатне розкручування своїх проектів. Причина цьому дуже проста: складність і затратність виведення цих проектів на масовий ринок. Якщо локально вони можуть успішно працювати, то на глобальному ринку втрачають свою унікальність і домінацію. Такий успішний запуск стартапів під силу тільки великим інтернет холдингам, у яких уже є багатомільйонна аудиторія.

Мета роботи є розробка веб-сервісу, який буде складатися з бази даних, бекенд частини та фронтенд частини. Для досягнення поставленої цілі потрібно виконати такі **завдання**:

— огляд аналогічничних веб-сервісів для пошуку виконавців для побутових задач за відповідною тематикою, виявлення їх переваг і недоліків;

— ознайомлення та вивчення і налаштування спеціалізованого програмного забезпечення;

—описати структуру бази даних;

— описати логіку серверної частини за допомогою мови програмування C# та .NET;

— описати логіку для клієнтської частини за допомогою мови програмування JavaScript та бібліотеки React;

— провести функціональне тестування серверної частини;

— провести функціональне тестування клієнтської частини;

— провести комплексне тестування веб-додатку.

Об’єктом дослідження є процес створення веб-сервісу для надання побутових послуг.

Предмет дослідження є сучасні технології проектування веб-сервісів для надання побутових послуг користувачам.

Методи дослідження є використання архітектурного підходу REST, та принципів об’єктно орієнтованого програмування, з використанням патерна програмування медіатор.

Практичне значення одержаних результатів — створено веб-сервіс для надання побутових послуг, з можливістю клієнту створити замовлення, а спеціалісту прийняти взяти замовлення в роботу.

Апробація результатів зроблено доповідь на науково-технічній конференції:

Кошельник В.О. Веб-сервіс для надання побутових послуг//Тези доповідей НТКП. Факультет інформаційних технологій на комп'ютерної інженерії (2022). Україна, Вінниця, 2022 р.: матеріали конференцій.

1 ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ІСНУЮЧИХ ВЕБ-СЕРВІСІВ В ДАНОМУ НАПРЯМКУ

У першому розділі наведена постановка задачі, а також проведено огляд існуючих аналогів в даній тематиці.

За допомогою веб-сервісу ми зможем працевлаштувати велику кількість людей та допомогти тим, хто потребує допомоги з побутовими проблемами.

1.1 Постановка задачі

Розробити веб-сервіс для замовлення та надання побутових послуг. Буде розроблений наступний функціонал: сторінка авторизації та реєстрації (аутентифікації), головна сторінка з можливими категоріями для замовлення, сторінка для створення замовлення, сторінка із відкритими замовленнями, сторінка перегляду окремого замовлення, сторінка з загальними замовленнями, можливість двофакторної авторизації, чат для спілкування адміністратора з відвідувачем сайту, реалізовано сповіщення для користувача, створений функціонал для редагування даних клієнта на спеціаліста. Також реалізована локалізація на таких мовах, як українська та японська.

1.2Огляд існуючих аналогів

Безробіття — це соціально-економічне явище, у якому частина робочої сили (економічно активного населення) не зайнята у виробництві товарів та послуг. Безробітні, поряд із зайнятими, формують робочу силу країни.

Сьогодні на ринку існує багато різних веб-сервісів, які надають подібні послуги. Протягом 2017-2019 рр. український ринок таких послуг, які підключені до побутових послуг, зростав у середньому на рік приблизно на 19%. У другому кварталі 2020 року через карантинні обмеження зростання ринку зупинилося та змінилося, погіршилася загальна економічна та соціально-політична ситуація в країні. У 2021 році попит знову починає зростати, що означає збільшення пропозиції.

Рисунок 1.1 — Динаміка обсягів реалізованих послуг підприємствами сфери послуг різним групам споживачів у 2016-2020 рр. в Україні, млрд. грн.

Віддалену роботу називають четвертою промисловою революцією. Нагадаємо, що все починалося з фермерства (перша революція), далі ручну працю замінили на машинну і люди пішли працювати на заводи (друга революція), а в 21 столітті панує розумова праця - робота в офісі (третя революція), проте офіс, як місце для продуктивної роботи стає вже непопулярним. Після цього залишається незрозумілим питання – як і де знайти проекти, якщо ви новачок?

Для цього є онлайн-біржі праці. Це такі майданчики, де клієнти публікують вакансії, а зареєстровані користувачі (як клієнти) відгукуються на такі оголошення. Робота відбувається онлайн, віддалено.

Існує безліч сервісів для пошуку роботи по типу фріланс. Вони поділяються на відповідні категорії, є загальними або вузьконаправленими. Тобто клієнт може зареєструватись на онлайн-біржі задля отримання оферу або зробити запит на надання роботи, тобто він буде виступати роботодавцем.

Моніторинг ринку наявних сервісів для замовлення побутових послуг в Україні показує, що найбільшим попитом в українських споживачів користуються такі послуги:

— майстри в таких сферах як: сантехніка, електрика, та складання;

— прибирання житла і офісів;

— кур’єрська доставка;

— ремонт побутової техніки і електроніки.

— оздоблення приміщень;

— ремонт в квартирах або будинках.

Найбільшу частину в структурі ринку України з надання побутових послуг займають послуги з доставки, які стали ще популярнішими в період карантину, особливо в великих містах.

Аналіз ринку в даній сфері свідчить, що для замовлення побутових послуг в Україні прогнозується подальший розвиток в галузі за допомогою таких факторів:

— зміни у формі організації роботи надавачів послуг, де все більше спеціалістів на ринку послуг, які замовляють побутові послуги в Україні вибирають самозайнятість, що дозволяє їм добре контролювати робочий час та рівень доходу;

— відмова корпоративних споживачів від штатних фахівців з надання побутових послуг на користь разових замовлень або використання аутсорсингу в міру необхідності;

— зниження купівельної спроможності населення країни, що призводить до скорочення споживання послуг, які не є предметами першої необхідності;

— розширити сферу ринку послуг із замовлення побутових послуг та можливості надання цих послуг за рахунок технічного прогресу, підвищити рівень технічної грамотності населення;

— реакція державних органів та суспільства на пандемію COVID-19.

З часом ринок трансформується і модифікується до сучасних тенденцій, ті послуги які раніше люди отримували не в дома такі як: перукарні, громадське харчування, офлайн магазини, тепер все більше надаються за

місцем проживання клієнтів або дистанційно з наступною доставкою. Розберем ряд платформ, сайтів, та сервісів.

1.2.1 Work.ua

Прикладом цього можна вважати сервіс для пошуку роботи «Work.ua». Це в Україні сервіс для пошуку роботи та співробітників, що надає понад 18 500 вакансій та кандидатів. Work.ua — сайт пошуку роботи №1 в Україні. Провідні компанії України розміщують свої вакансії та знаходять співробітників за допомогою Work.ua.

Рисунок 1.2 — Логотип сервісу «Work.ua»

1.2.2 Upwork

Хоч це й не є повноцінним аналогом, проте містить досить хороший інструментарій. Більш розширеним сервісом можна вважати «Upwork». Це найбільша біржа праці для фрілансерів. Американська компанія заснована в 2003 році (тоді називалася oDesk) та процвітає до сьогоднішнього дня.

Рисунок 1.3 — Логотип сервісу «Upwork»

На «Upwork» є три сторони:

— клієнт — йому потрібні послуги фахівця, формує завдання та оплачує його;

— фрілансер — фахівець, який виконує завдання клієнта та отримує за це гроші;

— агентство — організує спільну роботу кількох фрілансерів та клієнту зручно звертатися в агентство, якщо він має великий і різнопрофільний проект.

Наприклад, потрібно створити проморолик для компанії. Клієнт звертається в агентство, яке розподіляє завдання за різними фахівцями: сценарист займається сценарієм, 3Д-дизайнери створюють моделі, композитор пише музику. Така організація роботи зручна клієнту тим, що не потрібно створювати багато різних завдань і пояснювати їх кожному з фрілансерів. Агентство отримує відсоток від оплати замовлення, фрілансер — велике замовлення та репутацію агентства до свого профілю. Якщо клієнт має невелике завдання, він може звернутися безпосередньо до виконавця.

Якщо клієнту не сподобалась робота, яку виконав фрілансер, він може відкрити суперечку. Незалежний арбітраж розгляне ситуацію та винесе рішення на користь клієнта чи фрілансера. Це схоже на суперечку на «Алліекспрес».

Якщо клієнт після отримання виконаної роботи не виходить на зв'язок тобто не вдається зв’язатись із ним усіма методами, що він вказав у профілю чи під час формування замовлення, фрілансер також може звернутися до підтримки. Якщо правда на його боці, сайт перерахує кошти на рахунок фрілансера біржі.

1.2.3 Kabanchik.ua

Kabanchik.ua — веб-сервіс для пошуку кваліфікованих приватних фахівців для вирішення побутових задач. Майданчик об'єднує замовників, спеціалістів та різні сфери послуг, які необхідно виконати за певний проміжок часу за відповідну плату.

Рисунок 1.4 — Логотип онлайн сервісу «Kabanchik»

Сервіс не є вузько направлений і спеціалізується у багатьох сферах. Наприклад, сервіс надає такі послуги:

— адресна доставка (закупівля, доставка, отримання товарів і документів);

— домашній майстер (всі роботи, пов'язані зі збірки та монтажу меблів, підключенням і ремонтом побутової техніки, сантехніки);

— логістика (перевезення вантажів, пасажирів, оренда складу, вивезення сміття та снігу);

— розробка меблів (все, що пов'язано з виготовленням, частковим ремонтом та реставрацією);

— технічне обслуговування (прибирання приватної та комерційної нерухомості, виробничих об'єктів, прилеглих територій, видалення комах-паразитів, миття фасадів, вікон) і т.ін.

1.2.4 Rabotniki.UA

Rabotniki.ua — вузько направлений веб-сервіс який охоплює виключно тематику будівництва, ремонту та прибирання. Також модель монетизації та пошуку виконавця дещо відрізняється від Кабанчика. Кабанчик отримав вплив в більш менших послугах по обсягу такі як:

— ремонт розетки;

— монтаж люстри.

Rabotniki.UA публікують більше об’ємні і великі приватні будівельні послуги на великі обсяги робіт. У цілому всі замовлення відкриті для перегляду і все можна оцінити і переконатися особисто.

Рисунок 1.5 — Логотип сервісу «Rabotniki»

1.2.5 Freelancehunt.com

Freelancehunt.com — це фріланс сервіс для пошуку та взаємодії фрілансерів та замовників.

Слово «фріланс» походить від англійського freelance (вільний спис). Звичайно, сьогодні немає вільних найманців, фрілансери — це ті, хто вважає організацію трудових процесів і відносин у вільній формі і працює тоді, коли їм зручно виконувати роботу, яка їм подобається .

Термін фрілансер спочатку використовувався Вальтером Скоттом для опису «середньовічного найманця» в романі «Айвенго». Для виконання позаштатних робіт, тобто стати фрілансером, можна запросити будь-якого хорошого спеціаліста з прикладними знаннями та навичками. Серед фрілансерів є люди різних професій — від фізиків, інженерів, техніків, журналістів до копірайтерів, дизайнерів, програмістів , не обов’язково бути IT-спеціалістом — ви можете виконувати прості види роботи: розміщувати оголошення, залишати коментарі на форумах або заповнювати картки товарів в інтернет-магазині.

Як для замовників, так і для виконавців на сервісі пропонуються такі послуги:

— безпечна угода — гарантує оплату виконавцям у разі якісного виконання замовлення та повернення коштів замовнику, якщо робота не буде виконана або виконана неякісно де комісія послуги становить лише 9-10%, залежно від того, як вона буде розподілена, сторони можуть узгодити між собою, хто сплачуватиме 9% — замовник чи виконавець, також є можливість розділити комісію навпіл — по 5% у разі спірних питань сторони можуть звернутися до арбітражу;

— персональний проект — замовники можуть запросити до роботи виконавця, що підходить під вимоги, обравши його в каталозі фрілансерів також цей сервіс можна використовувати для доплат, якщо робота виконавця дуже якісно або, наприклад, початковий обсяг чи бюджет був неправильно розрахований;

— безпечна угода для бізнесу — безпечна угода між фрілансерами та юридичними особами з наданням документів, що закривають, під час роботи через відповідну угоду ви можете платити (як замовник) або отримувати оплату (як виконавець) за послуги за допомогою розрахункового рахунку у разі спірних питань учасникам надається арбітраж.

Рисунок 1.6 — Логотип сервісу «Freelancehunt»

1.2.6 OLX

OLX (англ. Online Exchange — онлайн обмін) — це є онлайн-платформа оголошень, яка об'єднує людей для купівлі, продажу або обміну товарами та послугами. У цьому сервісі виділяють також категорію для пошуку роботи.

Користувач може зареєструватись та обрати відповідну роль. У зележності від того як зареєструється користувач, він може знайти роботу або запропонувати свої послуги, як роботодавець.

Рисунок 1.7 — Логотип сервісу «OLX»

Онлайн-платформа надає можливість купівлі / продажу товарів та послуг, як приватним особам, так і представникам бізнесу. Користувачі можуть розміщувати свої оголошення лише попередньо зареєструвавшись за допомогою мобільного телефону, електронної пошти або увійти через облікові записи соціальних мереж. У формі подачі оголошення необхідно додати опис та фото позиції (фото не обов’язковим), а також свої контактні дані. У веб-сервісі є можливість відгукнутися на опубліковане оголошення, зв'язавшись з його автором через онлайн-чат на сайті або за номером телефону.

1.2.7 Robota.ua

Ще одним аналогом можна вважати сервіс для пошуку роботи «Robota.ua». Це є найпопулярніший сайт з пошуку роботи серед українців. Даний сервіс придназначений тільки для пошуку роботи, що дозволяє швидко найти роботу на відповідну ваканцію.

Рисунок 1.8 — Логотип сервісу «Robota.ua»

1.2.8 Послуги.ua

Також середаналогів в Україні розглядався веб-сервіс «Послуги.ua».

Даний сервіс дозволяє знайти виконавця для будь-яких задач, які вам потрібно виконати. Він також надає дві ролі, як майстра чи замовника.

Рисунок 1.9 — Логотип сервісу «Послуги.ua»

1.2.9 Flagma

Останнім сервісом, з аналогів є «Flagma». Даний сервіс орієнтований на користувачів з України та Польщі . Він теж надає великий спектр послуг які можуть надатися людям, зокрема переселенцям з України в Польщі.

Рисунок 1.10 — Логотип сервісу «FLAGMA»

2 ОГЛЯД ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ

Даний розділ присвячений опису використаних технологій, які використані під час розробки веб-сервісу. Існують різні інструменти, які описані нижче, завдяки їм є можливість реалізувати мою версію поставленого завдання. Проте розглянувши запропоновані доступні засоби реалізації, аналізуючи їх можливості, переваги, недоліки було вирішено скористатись таки засобами як: C#, .NET, JavaScript,SQL, React, MaterialUI.

2.1 Мова програмування C#

C# — сучасна об’єктно-орієнтована мова програмування. C# має великий спектр можливостей, що дозволяє розробникам вирішувати різної складності задачі та дозволяє писати безпечних та надійних програми, що виконуються в .NET. C# відноситься до широко відомого сімейства мов C. Ця мова програмування буде легка для розуміння, хто мав справу з такими мовами програмування як: C, C++, Java чи навіть JavaScript.

C# — об’єктно-орієнтована мова програмування, яка орієнтована на компоненти в програмуванні. C# надає широкі мовні конструкції підтримки такої концепції роботи. Завдяки такому підходу C# підходить для створення і застосування програмних компонентів. З моменту створення мова C# завжди розвивалася, та удосконалювалася новими можливостями, які полегшували роботу програміста при використані такого інструмента як C# [3].

Рисунок 2.1 — Логотип мови програмування C#

Переваги мови програмування C#:

— ця мова використовує об’єктно-орієнтований підхід до програмування всього, це означає, що вам потрібно буде описати абстрактні структури з точки зору предметних областей, а потім реалізувати взаємодії між ними, крім того цей метод популярний, оскільки дозволяє не запам’ятовувати всю інформацію і працює за принципом чорного ящика;

— у мові багато синтаксичного цукру, що робить важке життя програміста набагато простішим, щоб писати +100500 рядків коду, ви можете використовувати готовий дизайн, і компілятор виконає всю «брудну» роботу, хоча деякі з цих проектів не є оптимальними з точки зору продуктивності, усі вони перекриваються через читабельність коду та високу швидкість розробки;

— NET Framework [4], код, написаний програмістами на C#, перекладається на проміжну мову (IL), яка в свою чергу перетворюється безпосередньо в машинний код на комп’ютері під час виконання програми (JIT), у цьому випадку два програмісти можуть писати програмний код для одного проекту різними мовами, і жодному з них не потрібно вчитися заново, оскільки остаточна компіляція проміжного коду виконується на конкретній людині-машині, продуктивність можна підвищити за допомогою спеціальних команд людсько-машинного процесора;

— простий у використанні, необхідні розчини nuget завантажені та готові до використання, більшість із них безкоштовні, сюди також входить багато навчального та довідкового матеріалу;

— суворий введення дозволяє встановлювати надійний і кросплатформний у .NET Core [27].

Недоліки мови програмування C#:

—C# дуже легко дизассемблується, це означає, що є велика ймовірність того, що код буде отриманий і вивчений і проаналізований конкурентами;

—.NET будується на концепції JIT-компіляції, це означає, що програма буде скомпільована в машинний код при необхідності прямо під час роботи програми.

2.2 .NET-платформа

.NET — це платформа розроблена корпорацією Microsoft, яка дозволяє створювати програмні продукти. Вважається, що .NET Framework було розроблено, як альтернативу платформі Java від компанії Sun. Головна різниця цих платформ полягає в тому, що .NET Framework розрахований для роботи з операційними системами сімейства Microsoft Windows. Саме від першої версії пройшло багато часу , де версії фреймворка оновлювалися від 1.0 до 4.8 (18 квітня 2019), і на сьогодні .NET Framework залишається популярним, незважаючи на нового конкурента а саме появу нового покоління платформи (.NET Core), як і раніше, досить популярна: існує безліч програмних продуктів, бібліотек і фреймворків, які написані та розвиваються під .NET Framework [6].

У 2016 році до .NET Framework було випущено модульна платформа .NET Core, яка має сумісність з різними операційними системами. Іншими словами, дана платформа стала кросплатформованою. Кросплатформенність .NET Core відкрила великі можливості для нових сценаріїв і можливостей для подальшого використання. Це стало історичною подією, що відіграло велику роль у просуванні .NET серед розробників та бізнесу.

Багато хто думає, що мова програмування C# і платформа .NET це рівнозначні поняття, але це не так. Вони, стрімко розвиваються з огляду один на одного, але не мають якоїсь залежності між собою. В додачу до офіційно підтримуваних реалізацій .NET, існують аналогічні варіанти, такі як Mono, .NET Compact Framework, .NET Micro Framework та інші. Всі ці платворми частково використовують мову програмування C#. Якщо розглянути іншу сторону то .NET сумісний як C#, а й інші мовами програмування: F#, VB.NET і навіть C++ і Python.

2.3 База даних SQL

SQL — декларативна мова програмування для взаємодії користувача з базою даних, для того щоб сформувати запит, оновлення і керування реляційними БД, створення схеми бази даних та модифікації, засоби контролю та доступу до бази даних [7].

Іншими словами SQL — це мова програмування структурованих запитів (SQL, Structured Query Language), яка використовується як ефективний спосіб взаємодії з збереженими даними, пошуку, оновлення, видалення, вибірка за додатковими параметрами з бази даних.

Microsoft SQL Server — система управління базами даних, яка була розроблена корпорацією Microsoft. Як сервер даних виконує головну функцію а саме зберігання даних та доступ до наявних даних у відповідь на запити інших застосунків, які можуть оброблятися як на тому ж самому сервері, так і у мережі [24].

SQL дуже легкий у вивчені та розумінні, ця мова широко застосовується в галузі вільного програмного забезпечення, яка активно застосовується:

— розробниками баз даних (забезпечують функціональність додатків);

— адміністраторами (керують бд);

— тестувальниками (в ручному та автоматичному режимі).

Мова універсальна і має чітку визначену структуру за рахунок прописаних стандартів. Робота з базую даних відбувається швидко навіть у ситуаціях, де обробляються великі обсяги даних (Big Data). Крім того, ефективне управління можливе на інтуїтивному рівні навіть без особливих глибоких знань коду.

2.4 Мова програмування JavaScript

Багато хто з розробників визнають, що JavaScript має недоліки і складні частини. Однак JavaScript все рівно є, найбільш використовувана мова програмування. У результаті проведеного на «Stack Overflow» у 2021

опитування 69,7% з 47 184 опитаних професійних програмістів віддали перевагу JavaScript

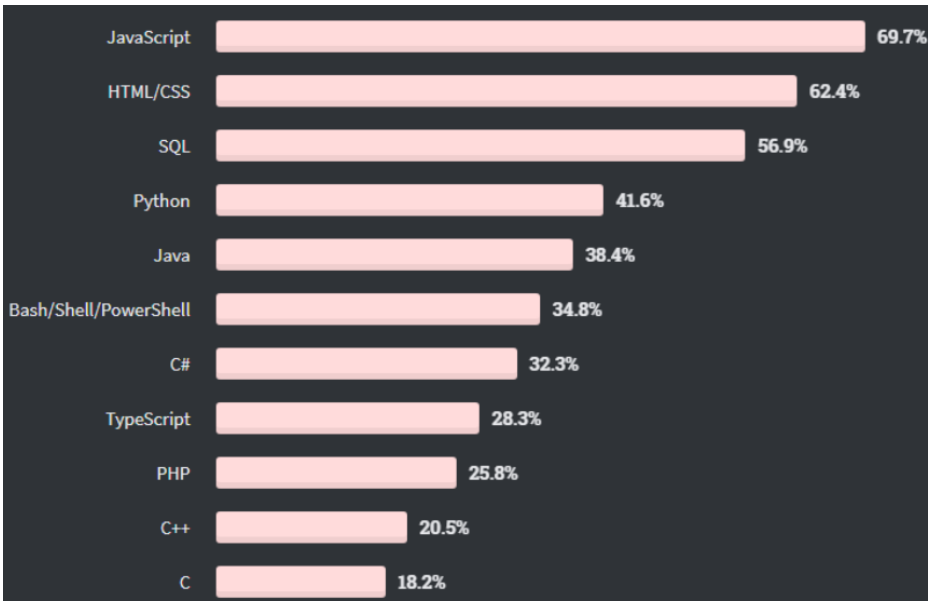

Рисунок 2.2 — Результат опитування

Зрозуміло, що JavaScript має ряд недоліків і є далеко не ідеальною мовою програмування. І все ж таки, не можна не звернути увагу на його багату екосистему з великою кількістю бібліотек, фреймворків, та інших корисних інструментів, які побудовані на JavaScript, а також величезний відсоток розробників які використовують JS.

JavaScript — мова програмування, що дозволяє використовувати ряд складних рішень у web-документах. Це допомагає зробити сторінки сайту більш інтерактивними. Це об’єктно-орієнтована клієнтська мова, яка підтримується додатками, що працюють з дизайном сайту. JavaScript став ще більш популярним у розробників, коли з'явилася AJAX-технологія, що призвело до нового етапу в розробці сайтів.

Ось лише кілька окремих прикладів застосування технології:

— відображення контенту, який час від часу;

— створення анімації і графічних об'єктів у форматі 2D/3D;

— опція роботи з відеозаписом в медіа програвачі.

Рисунок 2.3 — Логотип мови програмування JavaScript

Переваги JavaScript:

— швидкодія — є дуже швидким, тому що він часто запускається відразу в браузері клієнта.

— простота — синтаксис був схожий з Java і його порівняно легко вивчити ніж інші популярними мовами.

— сумісність — на відміну від PHP чи інших мов сценаріїв, JavaScript можна використати на будь-якій веб-сторінці. JavaScript можна використати в багатьох різних видах програм, завдяки підтримці інших мов:

— навантаження сервера — робота на стороні клієнта, тому загалом зменшує попит на сервер, а простим і нескладним програмам сервер може взагалі і не знадобитися.

— розширені інтерфейси — можливість використовувати для створення такі функцій, за допомогою яких ми можем реалізувати перетягування компонентів, таких як повзунки, що покращує користувальницький інтерфейс та роботу з таким сайтом.

— універсальність — є велика різноманітність використовувати різними методами JavaScript на різних етапах роботи.

— оновлення — з моменту випуску нового стандарту ECMAScript 5 (специфікація сценаріїв, на яку покладається JavaScript), ECMA International оновлює JavaScript щорічно.

Недоліки JavaScript:

— захист — на стороні клієнта бо код JavaScript працює на стороні клієнта, то помилки та недогляди іноді можуть бути використані для зловмисних цілей, тому через це деякі люди вирішують повністю вимкнути JavaScript в налаштуваннях браузера;

— підтримка браузера — сценарії на стороні сервера які завжди повертають однаковий результат, різні браузери іноді інтерпретують код JavaScript по-різному.

2.5 Додаткові інструменти для frontend

Для розробки фронтенд частини було обрано бібліотеку React.

React — JavaScript бібліотека з відкритим вихідним кодом для створення інтерфейсів користувача. React дозволяє створювати компоненти для мобільного та комп'ютерного програмного забезпечення. Ще його рекомендують використовувати для розробки SPA та корпоративних додатків. В рейтингах GitHub React — займає друге місце за популярністю. Його використовують Facebook, Instagram, WhatsApp.

Плюси використання:

— легка взаємодія з JavaScript і HTML;

— проста в розробці динамічні веб-сервісів;

— проста дебагу;

— підтримка спільноти.

Ця бібліотека є лише рівнем представлення.

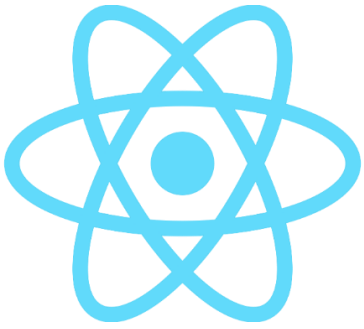

Рисунок 2.4 — Лого бібліотеки React

React часто порівняють з іншими JavaScript фреймворками, але суперечки «React vs Angular» не мають сенсу, тому що їх не можна порівнювати, це є різні речі. Angular — фреймворк (що включає і рівень представлення). React — бібліотека. React дає мову шаблонів та деякі callback-функції для відображення HTML. Весь результат роботи React — HTML. Зв'язки HTML/JavaScript, які називають компонентами, займаються тим, що зберігають свій внутрішній стан у пам'яті (наприклад: яка закладка обрана), але в результаті випльовується лише HTML.

Головні плюси:

— ви завжди можете визначити, як буде відображатися компонент, подивившись на вихідний код, це може бути важливою перевагою, хоча це мало чим відрізняється від шаблонів Angular, це дуже важливо при розробці складних програм, особливо в командах.

— зв’язування JavaScript і HTML у JSX робить компоненти легко зрозумілими, дивна комбінація HTML/JavaScript може ввести в оману де часто вчать не вставляти JavaScript в DOM (наприклад, обробники OnClick).

— ви можете відтворити React на сервері, якщо розробник створює загальнодоступний сайт або програму і відображає все на клієнті, він вибрав неправильний шлях.

Недоліки**:**

— не отримуватиме: системні події (окрім подій DOM), використовуючи AJAX, будь-які дані рівня, обіцянки, фреймворк, незалежно від обставин;

— погана і незрозуміла документація (наприклад, статистика);

— React досить великий, враховуючи те, як мало ви отримуєте від нього, включаючи погану міжбраузерну підтримку.

Для візуалізації було обрано нестандартну гіпертекстову розмітку та каскадні таблиці стилів, а також спеціальну бібліотеку «MaterialUI». MUI

надає повний набір інструментів користувальницького інтерфейсу, які допомагають швидше надавати нові функції.

Рисунок 2.5 — Лого бібліотеки «MaterialUI»

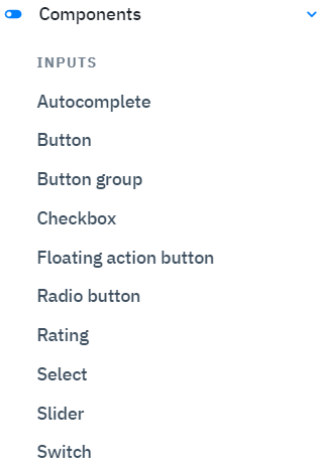

Рисунок 2.6 — Декілька варіантів компонент із бібліотеки

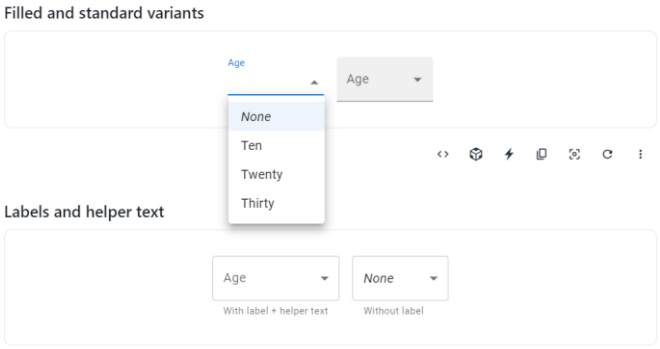

Рисунок 2.7 — Приклад компоненти «select»

HTML — це стандартизована мова розмітки для перегляду веб-сторінок. Веб-браузер отримує HTML-документ із сервера через HTTP/HTTPS або відкриває його з локального диска та інтерпретує код як інтерфейс, який буде відображатися на екрані монітора. Це код, який використовується для відображення та структурування веб-сторінки та її вмісту. Наприклад, вміст може бути структурований у вигляді кількох абзаців, маркованих списків або за допомогою зображень і таблиць даних.

Рисунок 2.8 — Логотип гіпертекстової розмітки (HTML)

Що таке справжній HTML? HTML не є мовою програмування, це є мова розмітки, яка вказує вашому браузеру, як відображати веб-сторінки, які ви відвідуєте. Це може бути складним або простим, залежно від того, як цього хоче веб-дизайнер. HTML складається з ряду елементів, які можна використовувати для вбудовування або обертання різних частин вмісту, щоб зробити його видимим або маніпулювати певним чином. Закриваючі теги можуть зробити слово або зображення посиланням на щось, виділити слово курсивом, збільшити або зменшити шрифт тощо.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Рисунок 2.9 — Структура HTML документу

CSS. Щоб знову зрозуміти його значення, вам потрібно посилання на Вікіпедію. Тому для опису їх зовнішнього вигляду використовується спеціальна мова стилю сторінки.

Рисунок 2.10 — Логотип мови стилів сторінок CSS

Це так званий код, який розробники використовують для стилізації веб-сторінок. Основи CSS допоможуть вам зрозуміти, як зробити текст чорним або червоним, як зробити вміст десь на екрані та як прикрасити веб-сторінки фоновими зображеннями та кольорами.

Як і HTML, CSS насправді не є мовою програмування. Це не мова розмітки, а мова таблиць стилів. Наприклад, щоб вибрати всі елементи абзацу на сторінці HTML і змінити текст всередині них з чорного на червоний, потрібно написати цей CSS:

Рисунок 2.11 — Приклад коду стилів

2.7 Програмні інструменти (середовище розробки)

Для розробки веб-сайтів доступно багато програм. Наприклад: «Visual Studio Code»,«Visual Studio».

«Visual Studio Code» — це інструмент, представлений Microsoft у квітні 2015 року для створення, редагування та налагодження сучасних веб-

додатків і програм хмарної системи. Основною перевагою є те, що це безкоштовний продукт.

Рисунок 2.12 — Логотип програмного продукту

Функції Visual Studio Code:

— VS Code дозволяє розробляти консольні та графічні додатки, а також веб-сайти та веб-додатки у рідному та керованому коді на всіх платформах;

— редактор має інструменти для роботи з Git, а також інструменти для рефакторингу, навігації по коду, автозаповнення для стандартної структури та контекстні підказки;

— цей продукт підтримує розробку на платформах ASP.NET і Node.js.

— однією з великих переваг редактора є те, що він підтримує велику кількість мов, таких як C++, C#, Python, PHP, JavaScript.

Переваги «Visual Studio Code»:

— налаштування;

— розширювані бібліотеки і готові рішення;

— простота та гнучкість.

«Visual Studio 2019» — чудова IDE, до того ж є повнофункціональна з безкоштовною версією Community. Microsoft Visual Studio — продукт компанії Microsoft, що передбачає інтегровану середу розробки програмного забезпечення та ряд допоміжних інших інструментів [14].

Рисунок 2.13 — Логотип програмного продукту

Особливості «Visual Studio»:

— «VS» дозволяє працювати з різними додатками;

— робота з відкладкою коду;

— робота з git;

— підказки для програміста;

— продукт дозволяє розробку на платформі ASP.NET.

3 РОЗРОБКА ВЕБ–СЕРВІСУ

3.1 Розбір та аналіз бази даних

База даних дуже важливий елемент в буть якому додатку, саме там зберігаються дані на рис. 3.1 можна побачити повну структуру нашої бази даних.

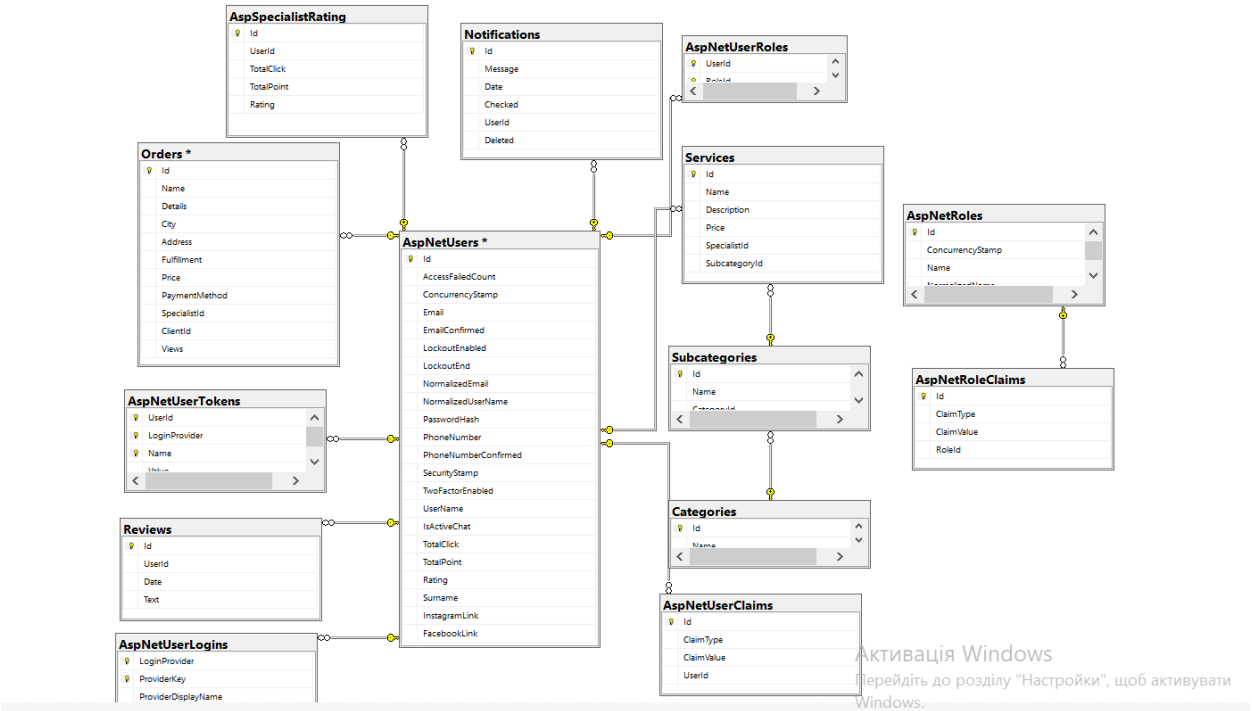

Рисунок 3.1 — Структура бази даних

У нас є ряд таблиць які згенерувалися за допомогою .NET Framework коли ми реалізовували авторизацію та аутентифікацію, а саме такі таблиці як: AspNetUsers, AspNetUserRoles, AspNetRoles та і інші таблиці з приставкою Asp.

Головною таблицею в нашій БД виступає AspNetUsers на рис. 3.2 більш детальна інформація про цю таблицю. Мабуть наступною по важливості є таблиця Orders більш детальна інформація на рис 3.3.

І мабуть останнє що б я хтів виділити в базі даних з таблиць це як в нас побудована робота сервісів. Це головна таблиця Services яка посилається на Subcategories, а сама таблиця має ще одну дочірню таблицю і це є Categories. Більш детальна інформація наведена в рис. 3.4.

AspNetUsers *

Column Name	Data Type	Allow Nulls
Id	nvarchar(450)	☐
AccessFailedCount	int	☐
ConcurrencyStamp	nvarchar(MAX)	☑
Email	nvarchar(256)	☑
EmailConfirmed	bit	☐
LockoutEnabled	bit	☐
LockoutEnd	datetimeoffset(7)	☑
NormalizedEmail	nvarchar(256)	☑
NormalizedUserName	nvarchar(256)	☑
PasswordHash	nvarchar(MAX)	☑
PhoneNumber	nvarchar(MAX)	☑
PhoneNumberConfirmed	bit	☐
SecurityStamp	nvarchar(MAX)	☑
TwoFactorEnabled	bit	☐
UserName	nvarchar(256)	☑
IsActiveChat	bit	☐
TotalClick	int	☑
TotalPoint	float	☑
Rating	float	☑
Surname	nvarchar(256)	☑
InstagramLink	nvarchar(MAX)	☑
FacebookLink	nvarchar(MAX)	☑
		☐

Рисунок 3.2 — таблиця AspNetUsers

Orders *

Column Name	Data Type	Allow Nulls
Id	int	☐
Name	nvarchar(50)	☐
Details	nvarchar(50)	☐
City	nvarchar(50)	☐
Address	nvarchar(50)	☐
Fulfillment	datetime	☐
Price	decimal(10, 2)	☐
PaymentMethod	nvarchar(50)	☐
SpecialistId	nvarchar(450)	☑
ClientId	nvarchar(450)	☐
Views	int	☑
		☐

Рисунок 3.3 — таблиця Orders

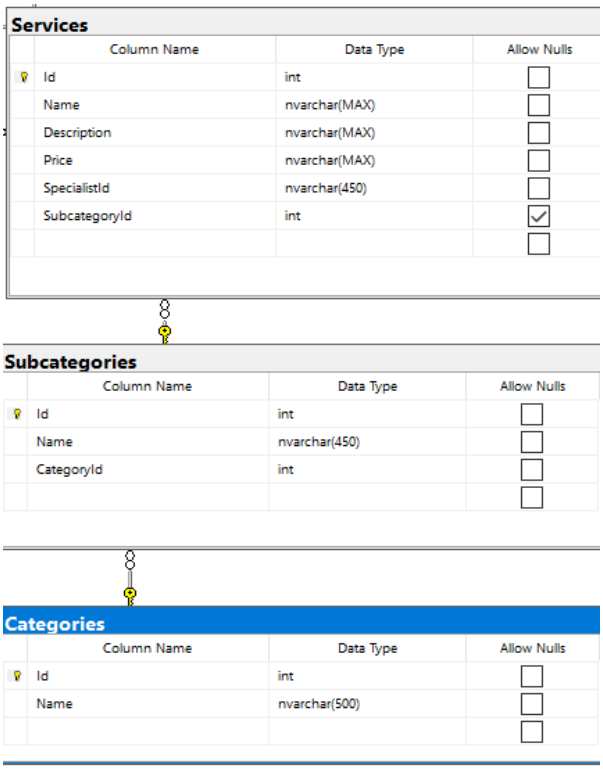

Рисунок 3.4 — таблиці Services, Subcategories, Categories

Інші таблиці теж важливі але вони охоплюють менший обсяг, і створені під конкретні задачі.

3.2 Розробка бекенд частини

Для розробки бекенд частини було прийнято рішення користуватися багаторівнева архітектура з використанням патерна програмування медіатор. На рис. 3.5 наведено структуру бекенд частини. Сам наш проект має назву Orders.

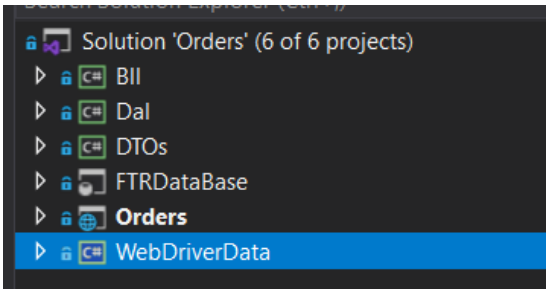

Рисунок 3.5 — Структура бекенд проекту

Як видно наш проект складається з таких з таких проектів як Bll, Dal, DTOs, FTRDataBase, Orders, WebDriverData. Де кожен проект відповідає за конкретну задачу і взаємодію.

Проект Bll відповідає за бізнес логіку в нашому додатку, на рис. 3.6 наведена структура проекту Bll.

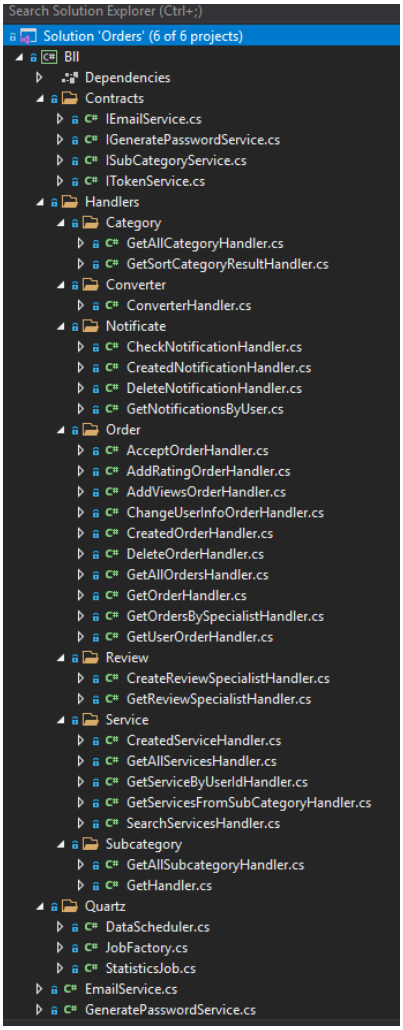

Рисунок 3.6 — Структура Bll проекту

Проект Bll містить такі папки як: Contracts, Handlers, Quartz. Кожна папка в себе включає ряд класів, які обробляють спеціальний функціонал для подальшої роботи з даними. Даний проект включає в себе ще ряд класів які не знаходяться в папках, а саме EmailService, GeneratePasswordService, JwtService.

Клас JwtService призначений для генерування jwt-токіну, це дозволя важати наш сервіс захищеним.

Проект Dal відповідає за взаємодію з БД, а саме записування даних в таблиці, витягування даних з різними умовами та інші маніпуляції з даними, на рис.3.7 наведена повна проекту Dal структура.

Велика кількість репозиторіїв які звертаються до бази даних, та отримують дані для подальшої роботи. Проект має такі папки як: Contracts, Models, OrderResponseRepositories.

Не враховуючи класи та інтерфейси які знаходяться в вище привединих папках, в проекті Dal є такі репозиторії: CategoryRepository, NotificationRepository, OrderRepository, ReviewRepository, ServiceRepository. В кожному з репозиторіїв є методи для проведення маніпуляції з даними.

Наступним проектом є DTOs саме тут створюються всі моделі які ми використовуємо по всьому проекту на рис. 3.8 зображена повна структура DTOs.

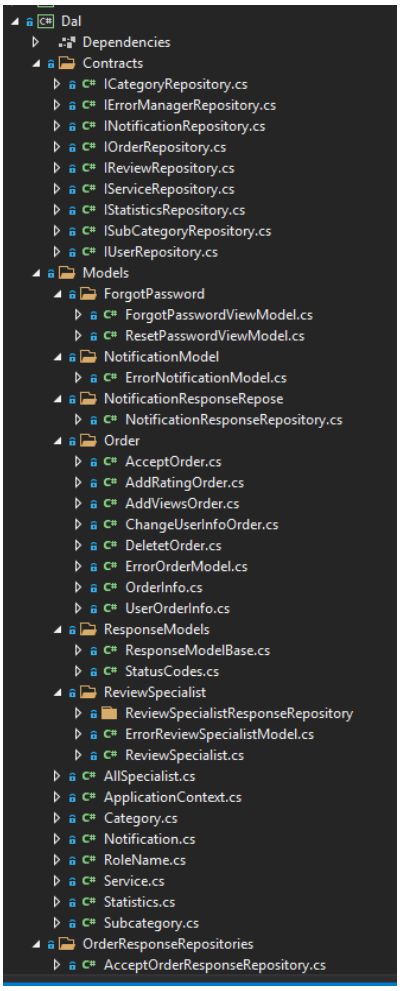

Рисунок 3.7 — Структура Dal проекту

В проекті DTOs прослідковується чітка структура де головними папками є Queries, ViewModels, відповідно кожна папка має свою ієрархію вкладення. В кінцевому випадку ми дійдем до моделі. Кожна модель має свої відповідні поля, з відповідними типами даних для зберігання.

Якщо пройти вкладення папки Queries, ми з в кожному класі зможем побачити конструктор для ініціалізації даних, та перезування даних на нищій рівень.

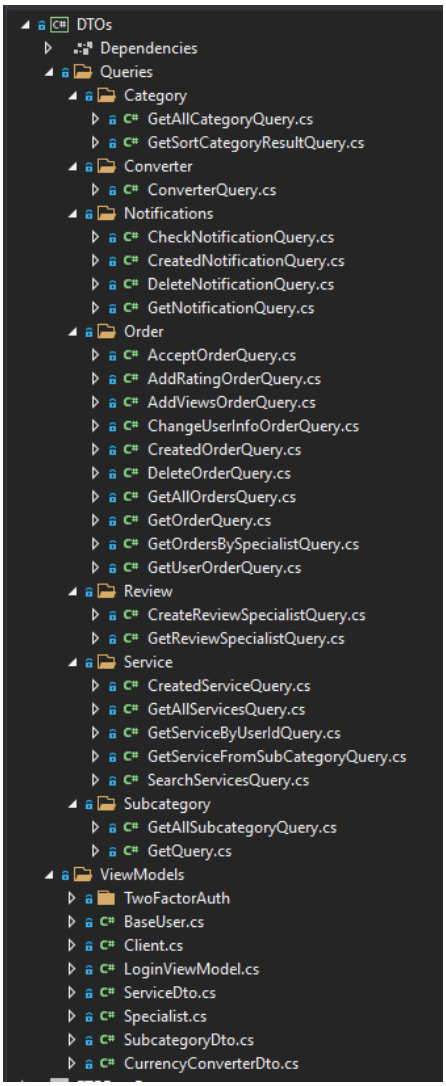

Рисунок 3.8 — Структура DTOs проекту

Проект FTRDataBase містить в собі всі наявні сторедж процедури та скріпти на створенні таблиць в базі даних, структура наведена в рис. 3.9.

Якщо розглядати даний проект більш детально то ми зможемо побачити таку чітку ієрархію, а саме це папку dbo де зберігаються скріпти для створення таблиць в базі даних.

В проекті існують такі таблиці: AspNetRoleClaims, AspNetRoles, AspNetUserClaims, AspNetUserLogins, AspNetUserRoles, AspNetUsers, Orders, Services.

Інші файли які наявні в проекті це сторедж-процедури, які виконують чітку задачу яка описана sql кодом. Прикладом таких сторедж-процедр є GetAllOrders, DeleteOrder. Де чітко із назви зрозуміло що ця процедура повинна виконати.

Головним проектом є Orders, саме тут в нас точка входу в наш бекенд частину, з головних папок виділив би Controllers структура даної папки наведена на рис.3.10 а структура всього 3.11

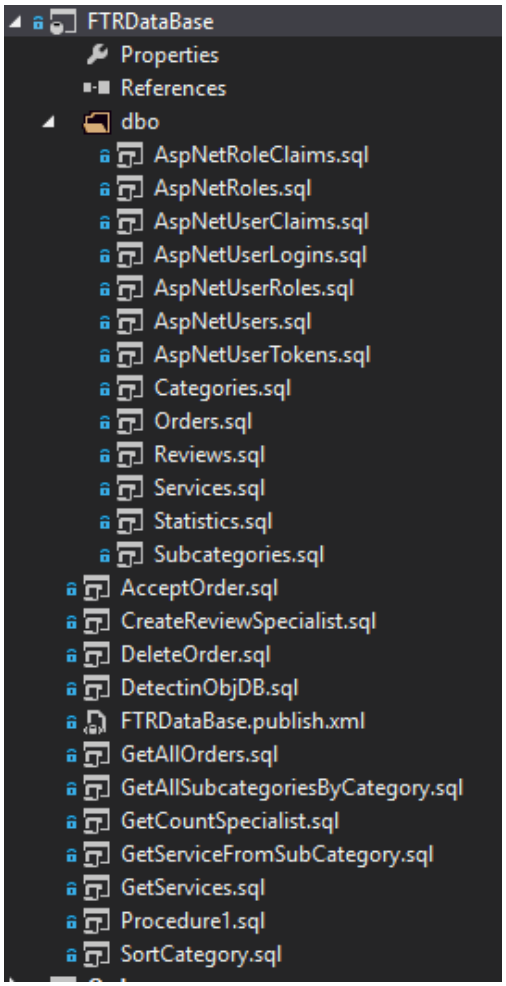

Рисунок 3.9 — Структура FTRDataBase проекту

Відповідно кожний контролер відповідає за свою чітку область роботи, де чітко з назви зрозуміло в якому напрямку ми будем рухатися. В кожному контролері існує безліч різних запитів які формують функціональність веб-сервісу.

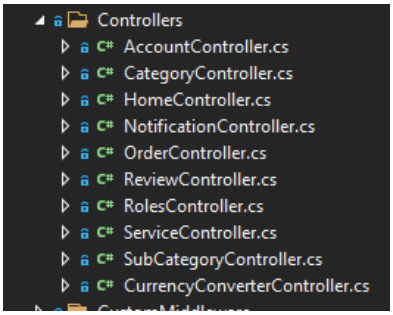

Рисунок 3.10 — Структура папки Controllers проекта Orders

Як видно з структури проекту Orders в даному проекті ще є такі папки як: ChatRepo, CustomMiddleware, NotifyRepo, Support. Кожна папка має свою конфігурацію файлів а саме клас, інтерфес, хаб.

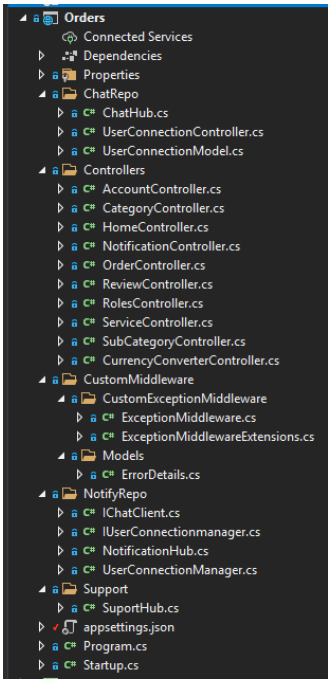

Рисунок 3.11 — Структура проекта Orders

Та останній проект це WebDriverData який використовується для збирання даних з сайту та запису даних в базу даних, тут реалізована така

технологія як Selenium, структура проекту WebDriverData наведена на рис3.12.

В проекті створено два класи WebDriver, TranslitMethods. В класі WebDriver створено такі методи GetCategory, GetUsers(), GetSubCategories(), GetServices(). Де кожен із методів знаходив дані які потрібно з парсити та передати в базу даних.

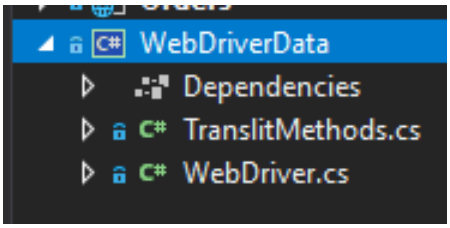

Рисунок 3.12 — Структура проекта WebDriverData

3.3 Робота з запитом в проекті

Так як ми знайомі з структурою проекта наприкладі розберемся як працює запит на бекенд частині. За приклад візьмем запит який буде нам поверти наявні замовлення в базі даних для користувача, топто ті замовлення які користувач створив, або якщо ми на сайті маєм роль як спеціаліста то ми

```
[HttpGet("GetOrdersByUser")]
0 references
public async Task<ResponseModelBase<IEnumerable<OrderInfo>>> GetOrdersByUser(string userId)
{
    var result = await _mediator.Send(new GetOrdersBySpecialistQuery(userId));
    return new ResponseModelBase<IEnumerable<OrderInfo>>(result);
}
```

отримаєм всі замовлення які виконав, або всзяв в роботу. Цей запит в нашому проекті називається GetOrdersByUser і знаходиться в контролері OrderController.cs на рис. 3.13 реалізація запиту GetOrdersByUser.

Рисунок 3.13 — Запит GetOrdersByUser

Одразу ми бачим що наш метод приймає userId де string виступає типом даних. Далі ми використовуємо клас GetOrdersBySpecialistQuery де його вміст наведений в рис.3.14. Тут в нас є конструктор який приймає і ініціалізує userId.

```
4 references
public class GetOrdersBySpecialistQuery : IRequest<IEnumerable<OrderInfo>>
{
    public readonly string UserId;

    1 reference
    public GetOrdersBySpecialistQuery(string userId)
    {
        UserId = userId;
    }
}
```

Рисунок 3.14 — клас GetOrdersBySpecialistQuery

Далі ми потрапляєм в клас з такою назвою як GetOrdersBySpecialistHandler, на рис. 3.15 наведений цей клас.

У цьому хендлері відбувається деяка бізнес логіка, а саме ми виконуєм метод GetUserRole який приймає userId та повертає нам роль користувача а саме спеціаліст або клієнт, далі в нас іде розгалуження відповідно до ролі, в нашому випадку припустим що ми отримали роль клієнта і опишем саме цю гілку.

В самому хендлері буде виконуватися метод GetOrdersByClient який приймає userId, та переходить в клас OrderRepository до реалізації методу GetOrdersByClient. У рис. 3.16 повністю реалізований метод GetOrdersByClient

```
1 reference
public class GetOrdersBySpecialistHandler : IRequestHandler<GetOrdersBySpecialistQuery, IEnumerable<OrderInfo>>
{
    private readonly IOrderRepository _repository;
    private readonly IUserRepository _userRepository;

    0 references
    public GetOrdersBySpecialistHandler(IOrderRepository repository, IUserRepository userRepository)
    {
        _repository = repository;
        _userRepository = userRepository;

    }

    0 references
    public async Task<IEnumerable<OrderInfo>> Handle(GetOrdersBySpecialistQuery request, CancellationToken cancellationToken)
    {
        var spec = await _userRepository.GetUserRole(request.UserId);

        var res =  spec.Role switch
        {
            Dal.Models.UserRole.Client => await _repository.GetOrdersByClient(request.UserId),
            Dal.Models.UserRole.Specialist => await _repository.GetOrdersBySpecialist(request.UserId),
            _ => null
        };

        return res;
    }
}
```

Рисунок 3.15 — клас GetOrdersBySpecialistHandler

```
2 references
public Task<IEnumerable<OrderInfo>> GetOrdersByClient(string userId)
{
    var sqlQuery = @"SELECT o.*, users.*
                     FROM Orders o
                     JOIN AspNetUsers users on users.Id = o.ClientId
                     WHERE ClientId = @ClientId
                     ";
    return Connection.QueryAsync<OrderInfo, UserOrderInfo, OrderInfo>(sqlQuery, (o, users) =>
    {
        if (o.User == null)
            o.User = users;

        return o;
    }, new { ClientId = userId });

}
```

Рисунок 3.16 — реалізація методу GetOrdersByClient

Після виконання методу GetOrdersByClient ми з даними які отримали з бази повертаємся у зворотному напрямку до котролера, і саме з котролера ми будем приймати дані на фронті. Роботу фронтенд частини наведено далі.

3.4 Процес авторизації / реєстрації

На рис. 3.17 можна побачити, що при переході на сторінку входу, з’являється вікно авторизації зі стандартними полями. Після виконання потрібного алгоритму дій можна увійти на сайт.

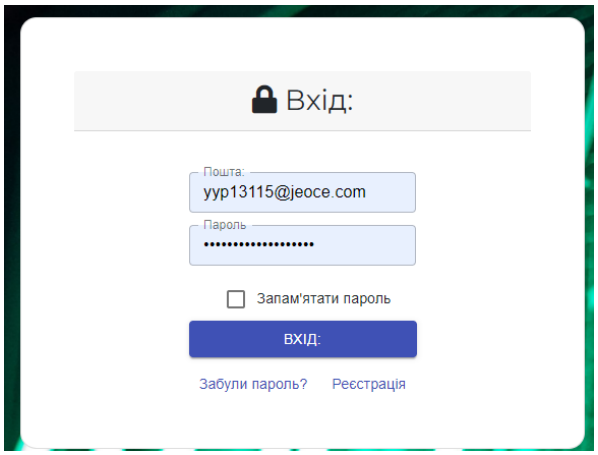

Рисунок 3.17 — Форма авторизації

На рис. 3.18 можна побачити, що при переході на сторінку реєстрації, з’являється вікно з формою для реєстрації, з можливість обрати роль в додатку. Після введення валідних даних в всі відповідні поля, та натискання кнопки, ми проходим алгоритм для реєстрації користувача в нашій системі, а

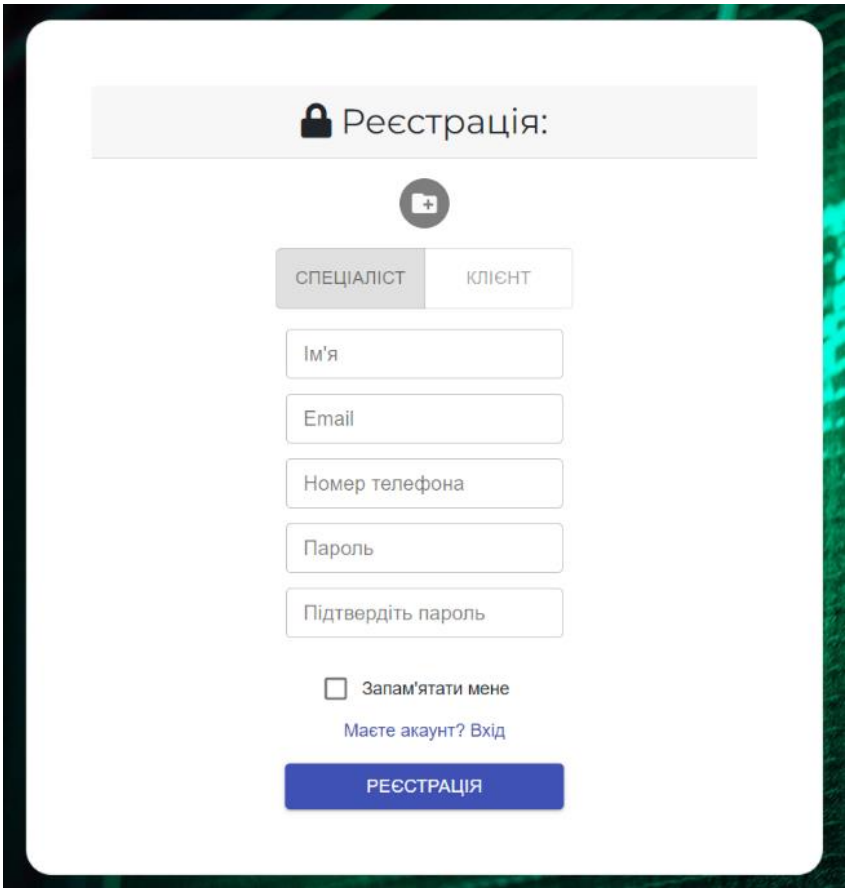

Рисунок 3.18 – Форма реєстрації

саме потрібно перейти на електрону пошту та підтвердити свою особистість. Далі відповідно від обраної ролі користувача буде надаватись певний функціонал з яким можна працювати в подальшому в системі.

3.5 Функціонал веб-додатку

Після виконання входу, користувач перейде на головну сторінку, рис.3.19. Де можна побачити категорії з підкатегоріями, які є на сайті для інформування користувача. Цим показано, що види послуг надаються в залежності від даних категорій / підкатегорій.

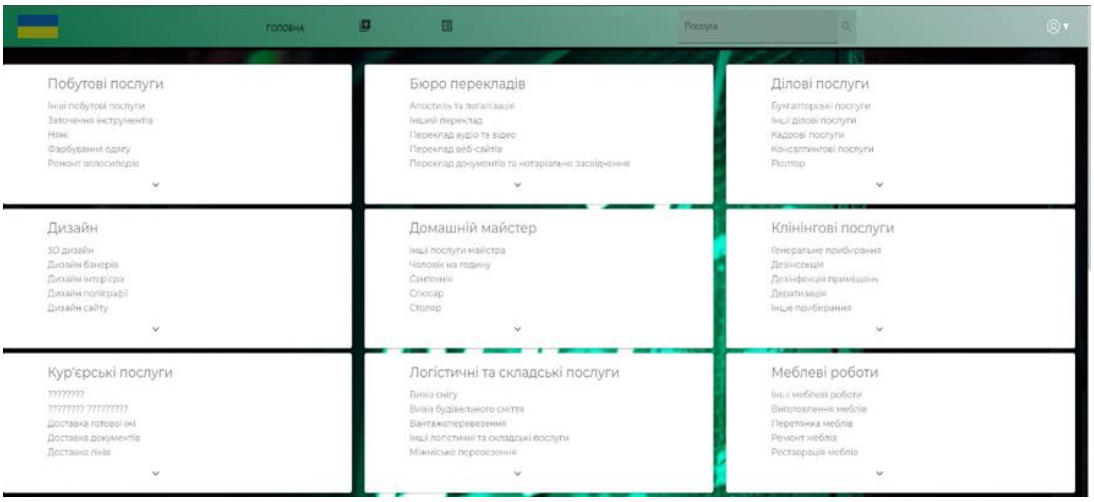

Рисунок 3.19 — Головна сторінка

У хедері можна побачити, що є навігація для зручного користування рис. 3.20.

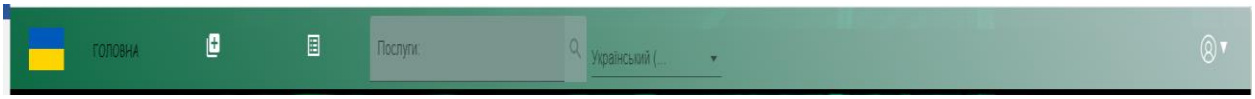

Рисунок 3.20 — Хедер додатку

При натисканні на кнопку, що має вигляд прапора України рис.3.21, користувач може побачити модальне вікно, де є інформація з рахунками для підтримки української армії та лінк, що веде на сайт фонду «Повернись живим».

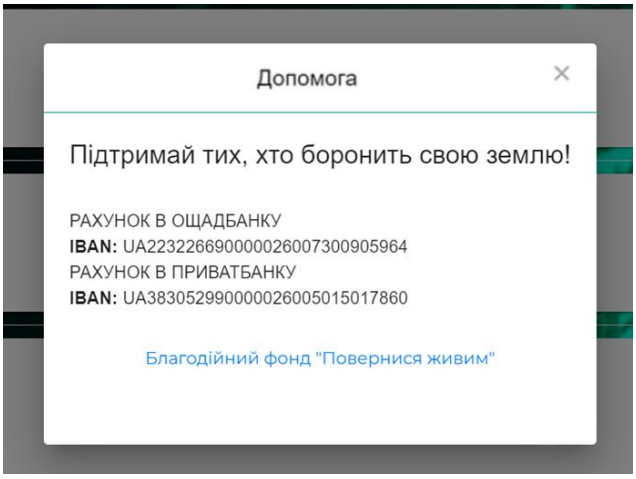

Рисунок 3.21 — Вікно для підтримки

На рис. 3.22 можна побачити сторінку для формування послуги, де є селекти з вибором категорії та підкатегорії. Якщо одне з полів для вводу буде пустим, то запит на розміщення замовлення не буде відправлено тому, що використовується валідація полів за допомогою бібліотеки «react-material-ui-form-validator». На рис. 3.22 зображено сторінку доступних замовлень, будь-хто з користувачів може зайти на неї, проте функціонал буде відрізнятись.

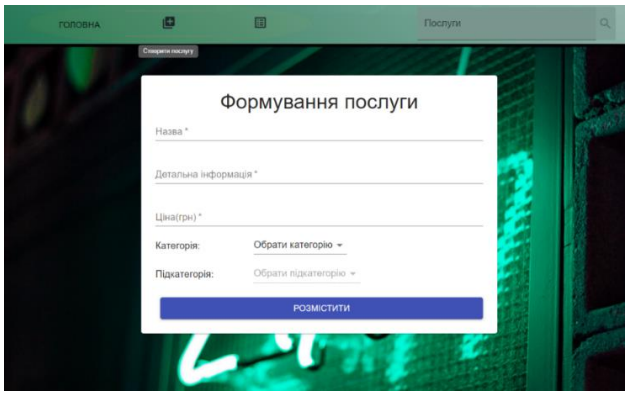

Рис. 3.22 — Сторінка для формування послуги

Різницю можна побачити на наступних рис. 3.23 та 3.24, де відрізняється кнопка текст кнопки. Для спеціаліста вивід тексту «прийняти замовлення», що дає можливість взяти дане замовлення в роботу. А для клієнта трішки інший функціонал, що дає можливість створити схоже замовлення.

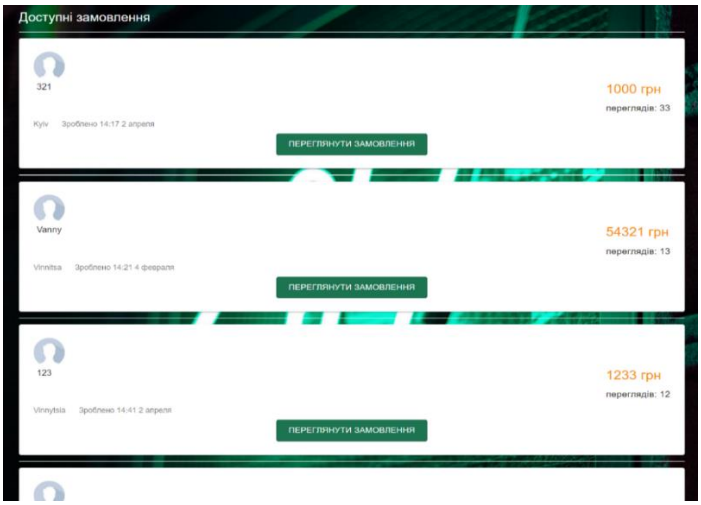

Рис. 3.23 — Сторінка доступних замовлень

Рис. 3.22 надає можливість переглянути, що спеціаліст може взяти замовлення для виконання даного замовлення. Коли спеціаліст натискає на кнопку для виконання замовлення то в нас формується PUT запит і відсилається на бекенд, де далі ми його обробляєм. Логіка нашого запиту дуже просто, ми відправляєм id спеціаліста де далі цей id записується у відповідне поле в базі даних, наступним кроком наше замовлення пропаде з основної колекції, тому що воно вже роботі.

Також це реалізовано для відсилання сповіщень у дзвіночок. Виходить так, що коли спеціаліст бере в роботу замовлення, то клієнту приходить сповіщення до дзвіночка та сповіщає його про те, що його замовлення взяв один із спеціалістів або коли спеціаліст завершив роботу.

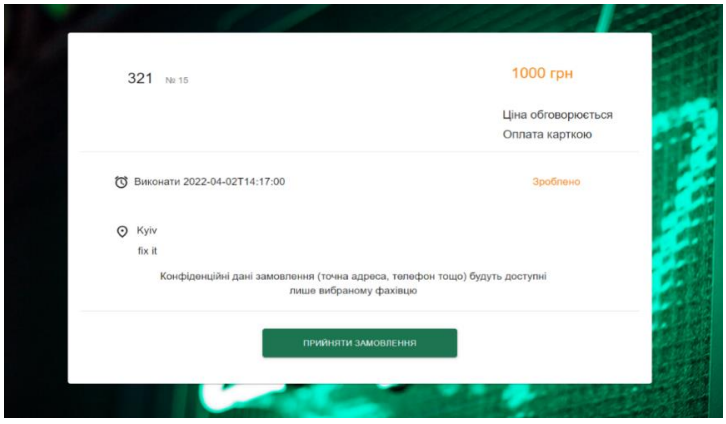

Рисунок 3.24 — Форма замовлення (вигляд зі сторони спеціаліста)

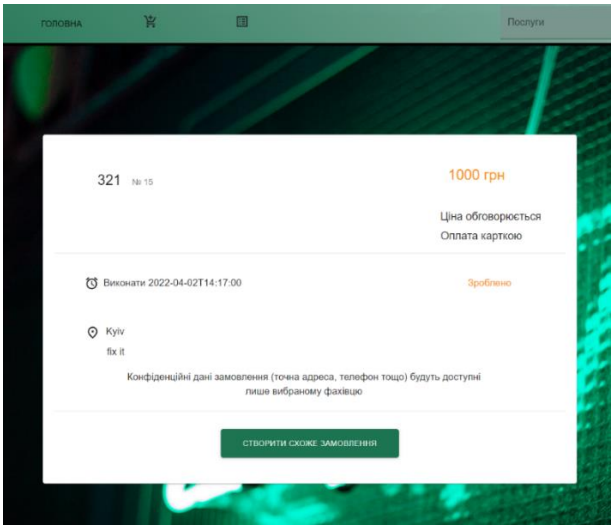

Рисунок 3.25 — Форма замовлення (вигляд зі сторони клієнта)

На рис. 3.26 можна переконатися в роботі пошуку послуги. У хедері користувач знайде поле вводу, де можна вписати назву категорії і автокомпліт підбере потрібний результат, що можна використовувати.

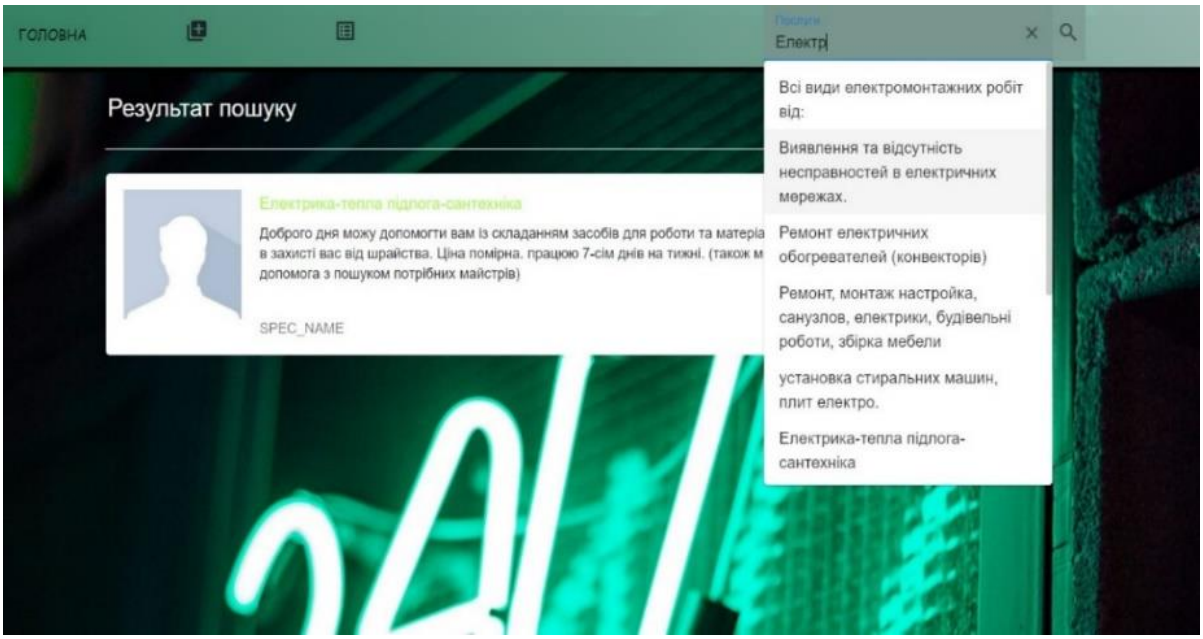

Рисунок 3.26 — Результат пошуку послуг

Даний функціонал реалізований за допомогою JavaScript. Алгоритм спрацює після ведення мінімум трьох символів далі іде зчитування даних які ми вели в поле відбувається фільтрація і вивід даних.

У футері рис.3.27 теж реалізовано великий спектр функціоналу.

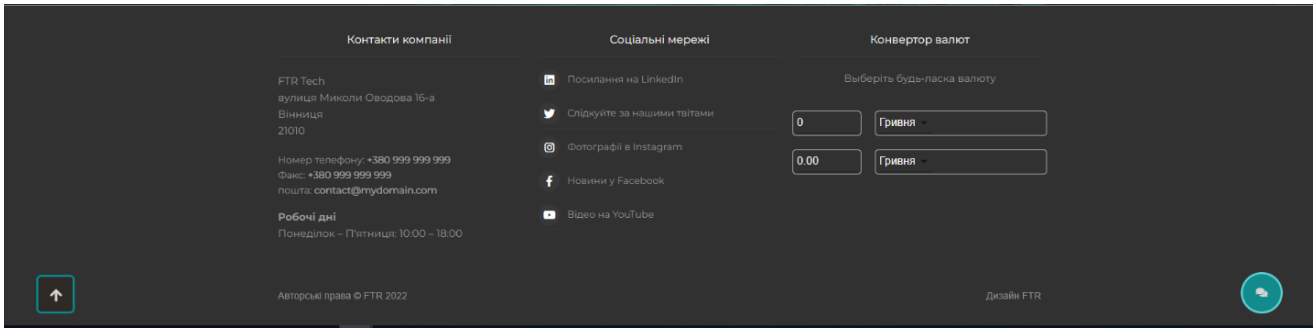

Рисунок 3.27 — Футер веб-додатку

А саме конвектор валют для користовачів який можна побачити на рис.3.26. Даний конвектор валют корисний тим що на сайті ціни за послугу можуть вказуватися в різних валютах. Це створено для того щоб користувач

не шукав якись стороній сервіс для переводу, а використав той що вже реалізований. Також у веб-сервісі реалізовано роль адміністратора на рис. 3.29 ми можем бачити сторінку входу в чат від адміністратора.

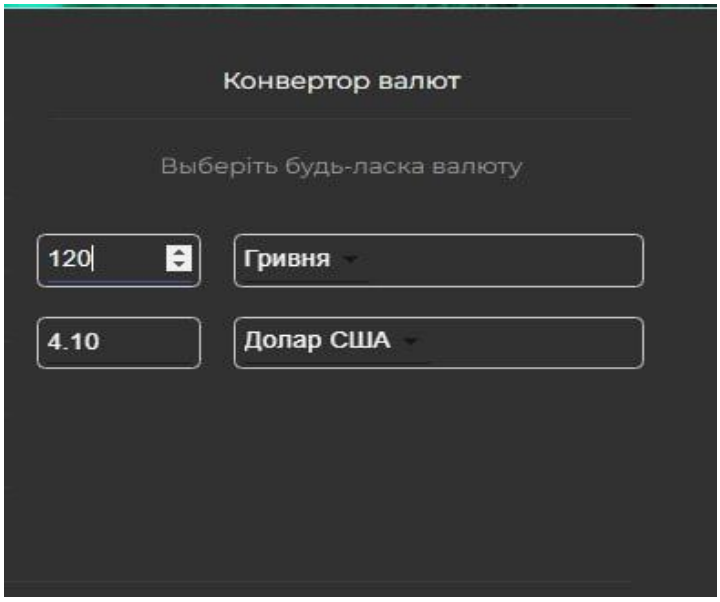

Рисунок 3.28 — Конвертор валют

Після натискання на кнопку адміністратор вводить своє ім’я під яким його зможе побачити користувач рис. 3.30, також подібний функціонал присутній для користувача рис.3.31.

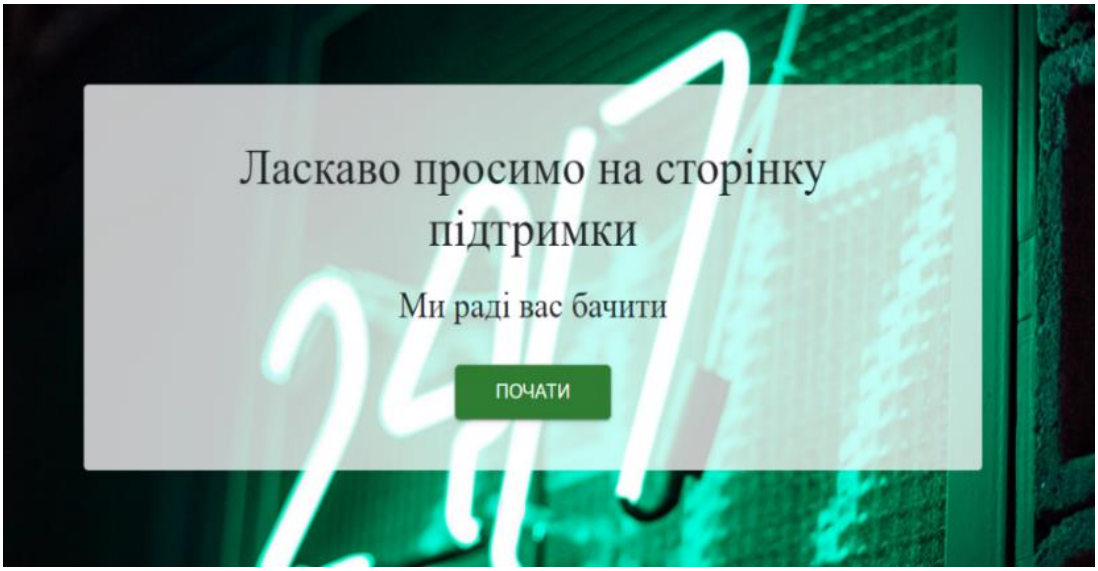

Рисунок 3.29 — Сторінка входу до чата

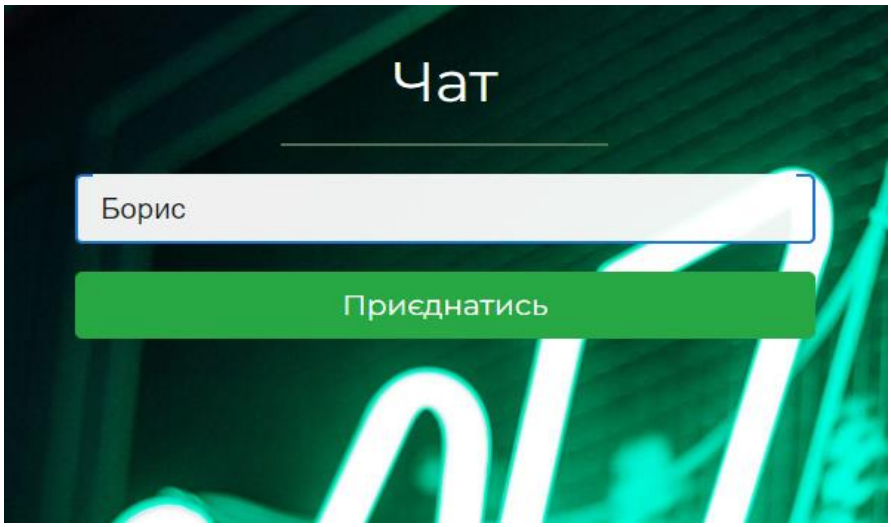

Рисунок 3.30 — Поле для ведення ім’я адміністратора

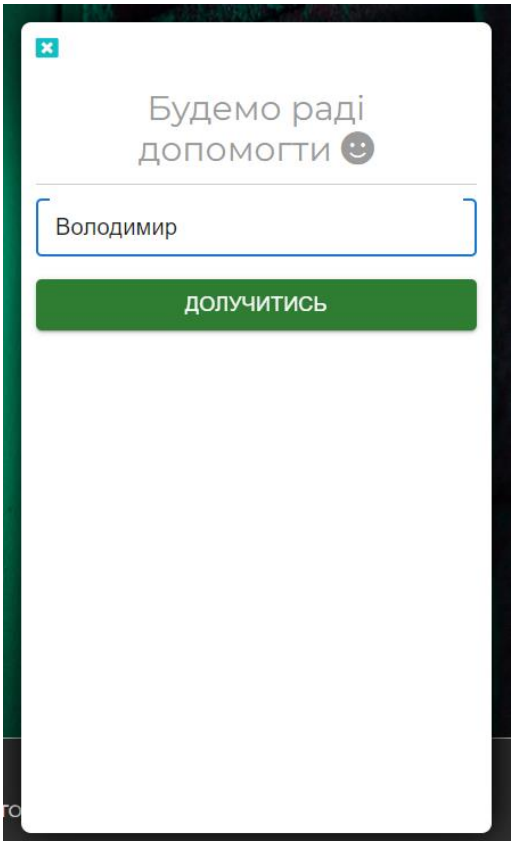

Рис. 3.31 — Поле для ведення ім’я користувачу

Наступним кроком адміністратору потрібно зайти в чат рис.32, і після того як адміністратор приєднався до чату він може спілкуватися з користувачем, що ми можем побачити на рис.3.33 з сторони адміністратора.

Рисунок 3.32 — Список доступних чатів для спілкування

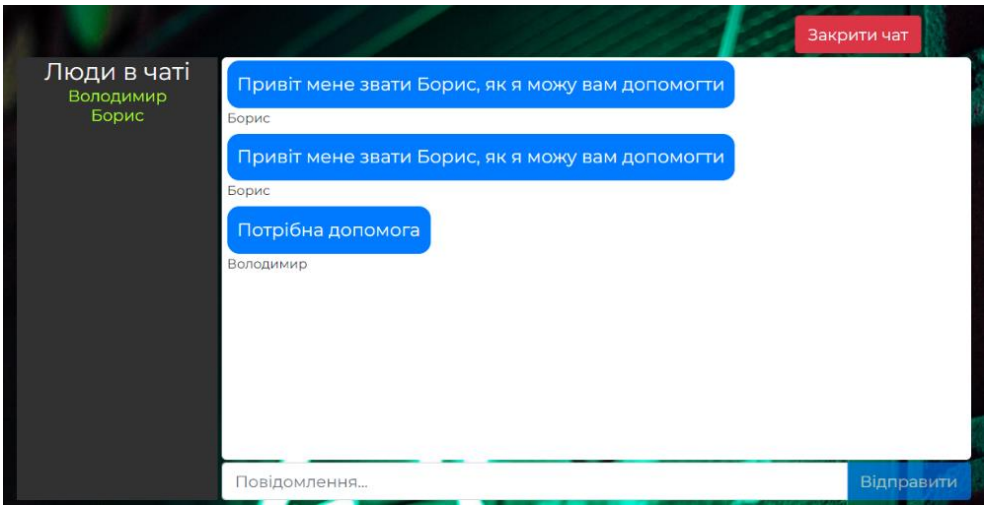

Рисунок 3.33 — Вигляд чату на стороні адміністратора

Реалізація чату відбувається за допомогою бібліотеки СігналАр. Створються два хаба — клієнт та адміністратор, хаб підключається на частині бекенд і фронтенд. На бекенд частині прописано методи, що потім посилаються на фронтенд (це методи, як наприклад приєднатись до кімнати, вийти з неї, відправити дані користувачів чи написати повідомлення).

На частині користувача реалізовано UI інтерфейс, для клієнта та адміністратора відповідно. На стороні адміністратора є форма в яку при зверненні до чату підтримки клієнт вписує своє ім’я та натискає на кнопку

«доєднатись», далі його під’єднує до хаба за допомогою методу «JoinRoom», а після цього під’єднує до кімнати, назва якої співпадає з його id.

Адміністратор у своїй формі робить те ж саме, після чого він долучається до кімнати без назви, він доєднується до хабу та отримує актуальну інформацію про абсолютно всіх користувачів, які мають бажання отримати відповідь (підтримку).

На сторінці є кнопка «рефреш», яка перепідключає користувача до хабу. Також існує кнопка для того, щоб вийти з хабу, відповідно вона від’єднує від хабу. Далі при виборі адміністратором користувача, він під’єднується до кімнати користувача, бібліотека signalr формує зв`язок між ними (по веб-сокету) і далі вони можуть обмінюватись повідомленнями. При виході кожної із сторін з чату, протилежна сторона інформується в чаті.

ASP.NET signalr є бібліотекою для розробників ASP.NET, що спрощує процес додавання веб-функцій до програми у режимі реального часу. Веб-функції (які працюють у режимі реального часу) дають можливість миттєво відправити якийсь вміст у міру доступності сервера на підключених клієнтів, а не чекати доки клієнт зробить запит на нові дані.

На рис. 3.34 можна побачити модальне вікно, за допомогою якого ви зможете перейти на сторінку профілю, що зображено на рис. 3.35. Там відкриваються можливості редагування профілю, перегляду послуг, що виконані або перебувають у процесі роботи. Також клієнт може виставити рейтинг спеціалісту, лише обравши відповідну оцінку.

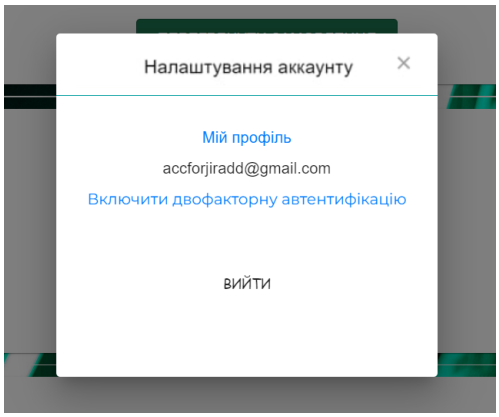

Рисунок 3.34 — Модальне вікно з користувача

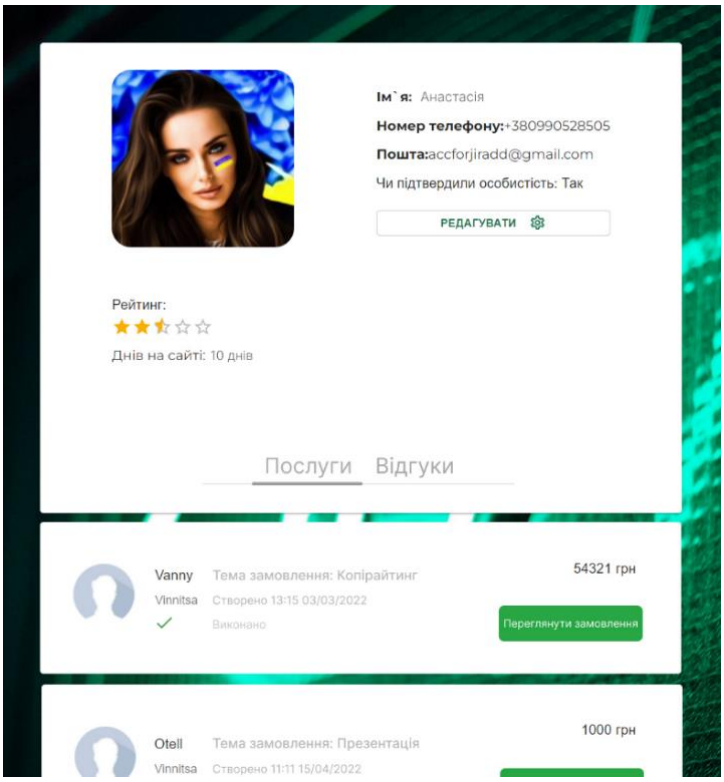

Рисунок 3.35 — Сторінка профілю спеціаліста

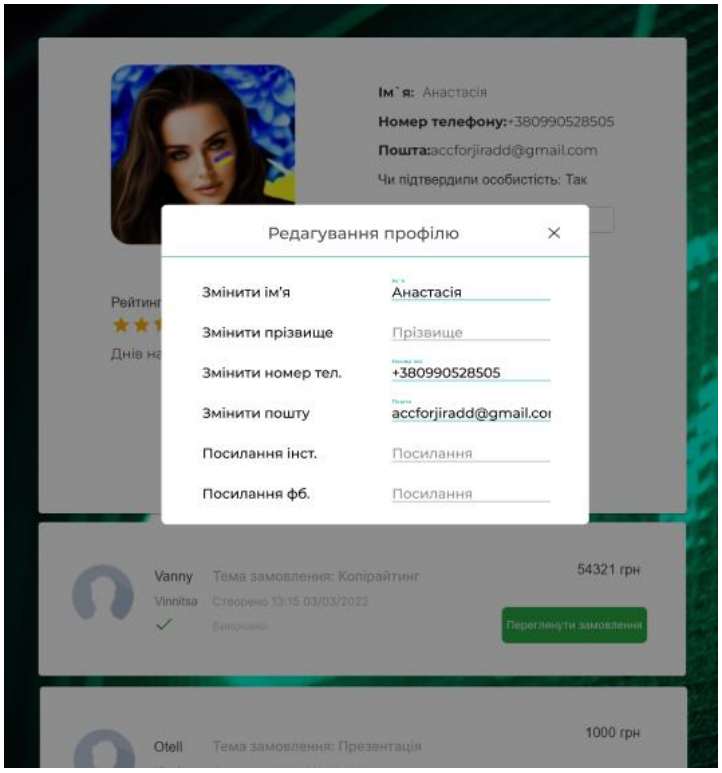

Рисунок 3.36 — Вікно редагування профілю

Рейтинг працює таким чином, що клієнт натискає на зірочку, далі з фронтенд частини відразу приймається 3 параметри: клік клієнта, кількість поставлених балів та сам рейтинг. Виходить так, що після кліку обробляються параметри і відправляються на бекенд, де через моделі та запит відправляються та зберігаються в SQL Server — базу даних. Через сторедж процедуру ділиться загальна кількість балів (від 0 до 5) на кількість кліків клієнтів, що дає можливість розрахувати правильно кількість рейтингу. Формула виглядає так: TotalPoin / TotalClick = Rating.

Далі за допомогою GET запита дані витягуються на фронтенд частині та записуються у правильному значені.

При натисканні на кнопку «редагувати», користувач має можливість побачити функціонал, що реалізований відповідно у модальному вікні рис. 3.34. На даному рисунку можна побачити, що користувачу надається можливість змінити ім’я, номер телефону та пошту, а також є можливість додати прізвище та посилання на соціальні мережі.

Також є сторінка з можливістю залишення відгуку рис.3.35, даний функціонал доступний клієнту.

Якщо розглядати профіль клієнта то у клієнта схожа сторінка профілю, проте єдине, що відрізняється так це відсутність сторінки для залишення відгуків, та відсутність рейтингу. На вкладці «послуги» у клієнта відображається список створених лише ним послуг рис 3.37.

Натомість у адміністратора є можливість перейти у чат підтримки, де він надасть допомогу користувачам, що потребують її. Також на сторінці профілю відображається роль (правий верхній кут), що дозволяє ідентифікувати користувача.

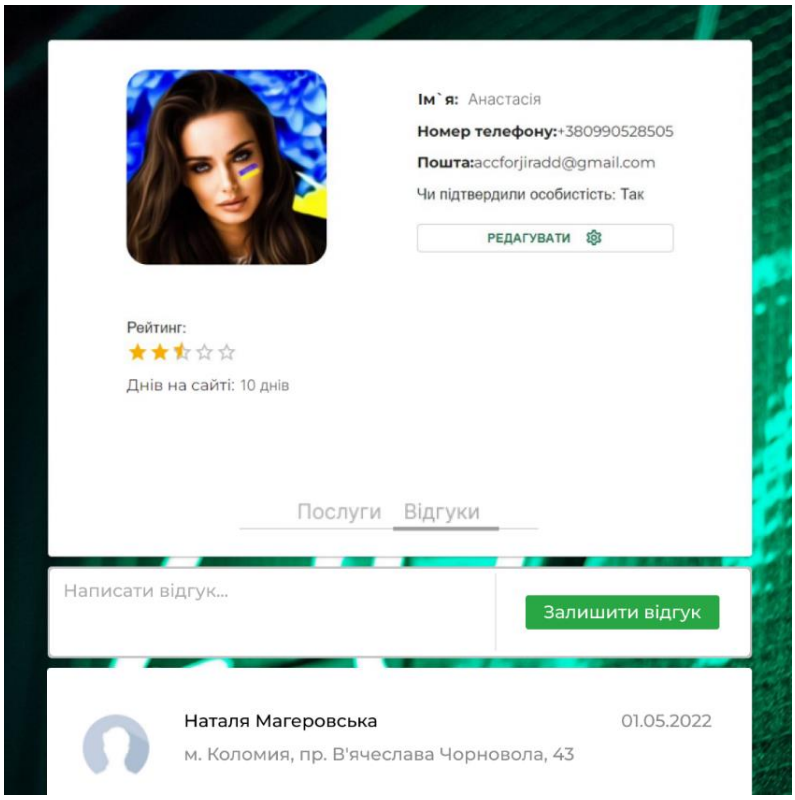

Рисунок 3.37 — Вкладка з відгуками (сторінка спеціаліста)

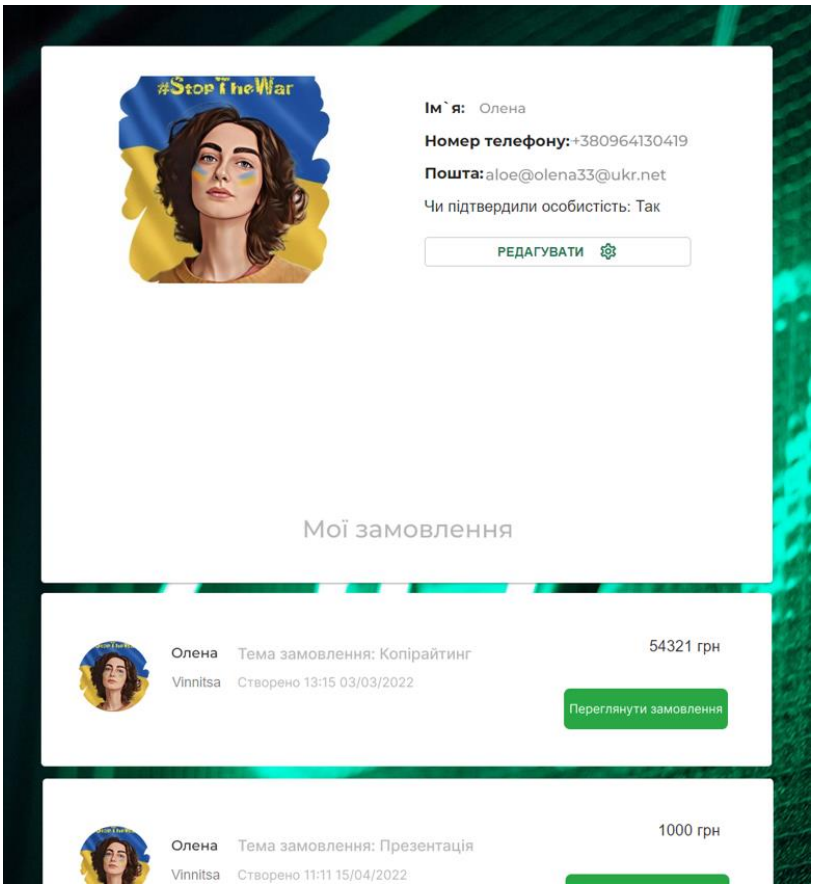

Рисунок 3.38 — Вікно з моїми замовленнями

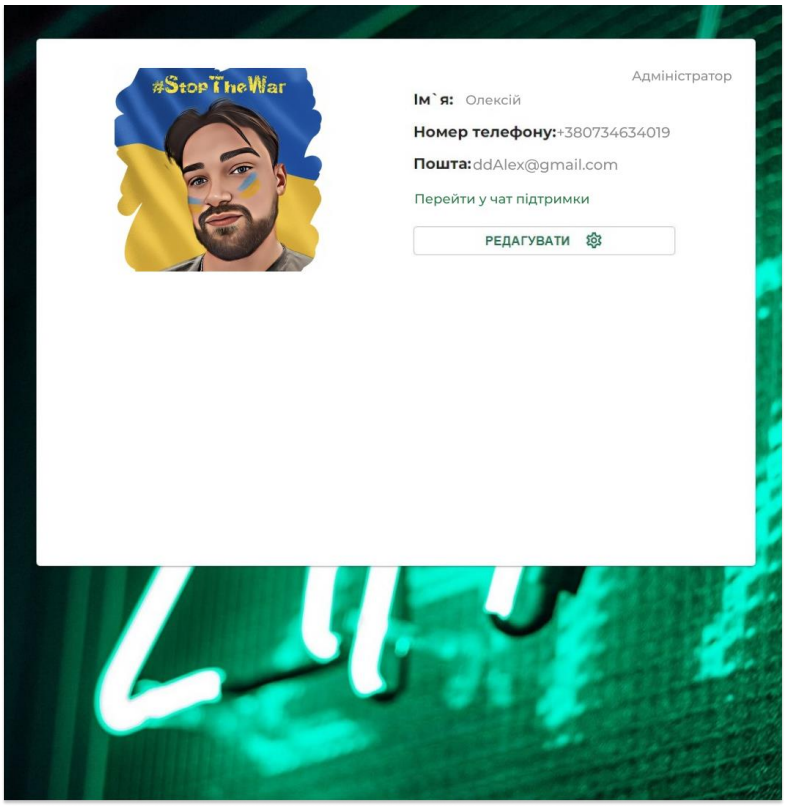

Рисунок 3.37—Вікно редагування профілю

ВИСНОВКИ

Реалізація веб-сервісу для та надання побутових послуг є доволі актуальною темою для роботи. Актуальність такої роботи полягає в тому, що станом на сьогоднішні тенденції які спричинила війна та пандемія люди проводять більше часу онлайн, тому виникають дві проблеми це безробіття та якісь побутові проблеми тому було вирішено розробити даний сервіс.

Було проаналізовано велику кількість конкурентів, з кожного сервісу вибиралися найкращий функціонал і вбудовувався в розроблений веб-сервіс. У роботі було описано ряд конкурентів з їх індивідуальними особливостями та можливостями.

Для досягнення поставленої цілі було вивчено різні методики проектування web–додатків, технічно архітектурними цілями на основі мови програмування C# та JavaScript. Постановка задачі була в розробці web–сервісу для вирішення побутових проблем, та заробітку а в деяких випадках і пошуку роботи. Для того щоб реалізувати поставлену задачу було прийнято глобальну задачу розбити на ряд менших задач. Для реалізації глобальної задачі були такі підзадачі:

—аналіз аналогів з виявленням їх переваг та недоліків;

—вивчення середовища розробки Visual Studio Code та Visual Studio.

—вивчення інструментів реалізації, а саме мов програмування C# та JavaScript, SQL, React, MaterialUI, мова розмітки HTML, каскадна мова програмування CSS.

—ряд архітектурних підходів та патернів програмування.

Головним прикладом вважається web–сервіс Kabanchik. Це дозволило оцінити структуру, загальне розуміння сервісу, головний функціонал який повинен бути доступний та обсяг роботи.

Розробка веб-сервісу, допоможе вирішити проблеми людей, щодо вирішення їхніх побутових проблем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. В.О. Кошельник Веб сервіс для надання побутових послуг; Науково-технічна конференція підрозділів Вінницького національного технічного університету (НТКП ВНТУ) м.Вінниця, Україна 31 травня 2022 року. -Вінниця ВНТУ.https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/14874

2. Побутова послуга (сайт): Веб-сайт. URL: https://www.kmu.gov.ua/npas/36531393(дата звернення: 24.04.2022).

3. Рівень безробіття в Україні (веб-сайт): Веб-сайт. URL: https://index.minfin.com.ua/ua/labour/unemploy/(дата звернення: 23.05.2022).

4. Безробіття (веб-сайт): Веб-сайт. URL: https://uk.wikipedia.org/wiki\Безробіття (дата звернення: 23.05.2022).

5. Work.ua(веб-сайт для пошуку роботи): Веб-сайт. URL: https://www.work.ua/ru/about-us/ (дата звернення: 23.05.2022).

6. Upwork (фріланс сервіс): Веб-сайт. URL: https://www.upwork.com (дата звернення: 24.04.2022).

7. Kabanchik (веб-сервіс для надання послуг): Веб-сайт. URL: https://www.kabanchik.ua (дата звернення: 24.05.2022).

8. Rabotniki (фріланс сервіс): Веб-сайт. URL: https://www. rabotniki.ua (дата звернення: 25.05.2022).

9. Freelancehunt (онлайн платформа оголошень): Веб-сайт. URL: https://www. freelancehunt.com (дата звернення: 25.05.2022).

10. OLX (онлайн платформа оголошень): Веб-сайт. URL: https://www.olx.ua (дата звернення: 24.04.2022).

11. Rabota.ua (веб-сайт для пошуку роботи): Веб-сайт. URL: https://www.rabota.ua (дата звернення: 24.04.2022).

12. Послуга.ua (веб-сервіс для надання послуг): Веб-сайт. URL: https://www. poslugy.ua (дата звернення: 25.04.2022).

13. Flagma (онлайн платформа оголошень): Веб-сайт. URL:

https://www. flagma.ua (дата звернення: 25.04.2022).

14. Рихтер, П. CLR via C# програмуваня а платформі .NET Framework / Д. Рихтер: Мир, 2013.

15..NET Framework (Електронний ресурс) – Режим доступу до ресурсу: https://training.epam.ua/#!/News/301?lang=ru

16. Боль, А. Learning SQL/ А.Боль: O'Reilly, 2009.

17. SQL (Електронний ресурс) – Веб-сайт. URL: https://uk.wikipedia.org/wiki/SQL

18. Принцип роботи технології AJAX [Електронний ресурс] – Веб-сайт. URL: https://www.hostinger.com.ua/rukovodstva/chto-takoje-ajax/

19. Опитуванням на «Stack Overflow» (Електронний ресурс) – Веб-сайт. URL: https://stackoverflow.com/questions/javascript

20. MaterialUI (Електронний ресурс) – Веб-сайт. URL: https://mui.com/

21. HTML (Електронний ресурс) – Веб-сайт. URL: https://uk.wikipedia.org/wiki/HTML

22. Google Authenticator (Електронний ресурс) – Веб-сайт. URL: https://uk.wikipedia.org/wiki/Google_Authenticator

23. Хавербеке, М. Виразний JavaScript / М. Хавербеке: Прес, 2019.

24. Фленаган, Д. JavaScript повний курс / Д. Фленаган: Реіллі, 2020.

25. JavaScript (Електронний ресурс) – Веб-сайт. URL: https://uk.wikipedia.org/wiki/JavaScript

26. Signalr (Електронний ресурс) – Веб-сайт. URL: https://docs.microsoft.com/ru-ru/aspnet/signalr/overview/getting-started/introduction-to-signalr

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ ВНТУ

д.т.н., проф.

_____О. Д. Азаров

“__” _____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

«Веб-сервіс надання побутових послуг»

08-23.БДР.035.00.000 ПЗ

Науковий керівник к.т.н., доц. каф. ОТ

_____ Колесник І.С.

Студент групи 1КІ-20мс

_____ Кошельник В. О.

Вінниця 2022

1 Найменування та область застосування

Робоча назва проекту «ФТР», розроблявся для вирішення побутових проблем та працевлаштування людей.

2 Основи для розробки

Основою для розробки є дисципліни Обєктно орієнтоване програмування, Програмування.

3 Мета та призначення розробки

Експлуатаційне призначення розробки — для пошуку і вирішення побутових проблем з якими людина не може впоратися сама.

4 Етапи БДР та очікувані результати

Робота виконується в п’ять етапів, що наведені в таблиці 4.1.

Таблиця 4.1 — Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	11.03.22	12.03.22	Вступ
2	Аналіз сучасних веб-сервісів для вирішення побутових проблем та працевлаштування	12.03.22	28.04.22	розділ 1
3	Вибір інструментів для розробки веб-сервісу для вирішення побутових проблем	28.04.22	12.05.22	Розділ 2
4	Проектування веб-сервісу для вирішення побутових проблем	13.05.22	20.05.22	Розділ 3
5	Оформлення пояснювальної записки	20.05.22	24.05.22	ПЗ, презентація

5 Матеріали, що подаються до захисту БДР

Пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відзив наукового керівника, рецензія опонента, протоколи складання державних екзаменів, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

6 Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

7 Вимоги до оформлення БДР

Вимоги до БДР наведені нище:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Единa система конструкторської документації. Основні написи»;

— Документами на які посилаються у вище вказаних.

Технічне завдання до виконання отримав _____ Кошельник В.О.

ДОДАТОК Б

Лістинг класу Header.js

```
import React, { useState, useEffect } from "react";
import { Route, BrowserRouter as Router } from "react-router-dom";
import i18n from "i18next";
import { initReactI18next } from "react-i18next";
import LanguageDetector from "i18next-browser-languagedetector";
import HttpApi from "i18next-http-backend";
import {
  SpecialistPage,
  OrderPage,
  OrderChoicePage,
  NotFound,
  ChatApp,
  GoToSupport,
} from "../pages";
import UserProfile from "../pages/User/UserProfile";
import AddService from "./Service/AddService";
import { StartOrder } from "./Order";
import {
  Home,
  SearchServices,
  SignUp,
  SignIn,
  ForgotPassword,
  ResetPassword,
  TextForForgotPassword,
  EnableTwoFactorAuth,
  DisableTwoFactorAuth,
```

```
} from "./AdminPanel";

import Navigation from "./Navigation";

import "../App.css";

i18n
  .use(initReactI18next)
  .use(LanguageDetector)
  .use(HttpApi)
  .init({
    supportedLngs: ["uk", "jp"],
    fallbackLng: "uk",
    detection: {
      data: [],
      caches: ["cookie"],
    },
    backend: {
      loadPath: "/assets/locales/{{lng}}/translation.json",
    },
    react: { useSuspense: false },
  });

function Header() {
  const [userData, setUserData] = useState({
    id: "",
    email: "",
    role: "",
    twoFactorEnabled: false,
  });
```

```
const [loginStatusFlag, setLoginStatusFlag] = useState(false);

window.localStorage.setItem("userId", userData.id);
window.localStorage.setItem("accountRole", userData.role);

const updateLoginStatus = () => {
  setLoginStatusFlag(!loginStatusFlag);
};

useEffect(() => {
  async function checkLoginStatus() {
    return await fetch(process.env.REACT_APP_API + "account/User", {
      headers: {
        Accept: "application/json",
        "Content-Type": "application/json",
      },
      credentials: "include",
    });
  }
  let isOk = false;
  checkLoginStatus()
    .then((res) => {
      isOk = res.ok;
      return res.json();
    })
    .then((data) => {
      console.log(data);
      if (isOk) {
        setUserData({
          id: data["id"],
```

```
            email: data["email"],
            role: data["role"],
            twoFactorEnabled: data["twoFactorEnabled"],
            token: data["token"],
          });
        } else {
          setUserData({
            id: undefined,
            email: undefined,
            role: undefined,
            twoFactorEnabled: false,
          });
        }
      });
  }, [loginStatusFlag]);
  return (
    <Router>
      <div className="Header">
        <Navigation
          Email={userData.email}
          Role={userData.role}
          TwoFactorEnabled={userData.twoFactorEnabled}
          updateLoginStatus={updateLoginStatus}
        />
      </div>
      <Route path="/Home" component={() => <Home Email={userData.email} />}
/>
      <Route path="/SignUp" component={SignUp} />
      <Route
        path="/SignIn"
```

```
        component={() => <SignIn updateLoginStatus={updateLoginStatus} />}
      />
      <Route path="/ForgotPassword" component={ForgotPassword} />
      <Route path="/ResetPassword/:verification?" component={ResetPassword} />
      <Route
        path="/StartOrder"
        component={() => <StartOrder Id={userData.id} />}
      />
      <Route
        path="/AddService"
        component={() => <AddService Id={userData.id} />}
      />
      <Route path="/service/search/:searchString" component={SearchServices} />
      <Route path="/TextForForgotPassword"
component={TextForForgotPassword} />
      <Route path="/Specialist" component={SpecialistPage} />
      <Route
        path="/Order"
        component={() => <OrderPage Id={userData.id} Role={userData.role} />}
      />
      <Route path="/OrderChoice" component={OrderChoicePage} />
      <Route
        path="/Account/TwoFactorAuthentication/Enable"
        component={() => (
          <EnableTwoFactorAuth updateLoginStatus={updateLoginStatus} />
        )}
      />
      <Route
        path="/Account/TwoFactorAuthentication/Disable"
        component={() => (
```

```
          <DisableTwoFactorAuth updateLoginStatus={updateLoginStatus} />
        )}
      />
      <Route
        path={"/Account"}
        render={() => <UserProfile editable myProfile role={userData.role}/>}
        exact
      />
      <Route
        path={"/Account/:accountId"}
        render={() => <UserProfile />}
      />
      <Route path="/ChatApp" component={ChatApp} />
      <Route path="/GoToSupport" component={GoToSupport} />
      <Route path="/" component={NotFound} exact />
    </Router>
  );
}

export default Header;
```

ДОДАТОК В

Лістинг класу OrderRepository

```
using Dal.Contracts;
using Dal.Models.Order;
using Dal.Models.Order.ErrorOrderModel;
using Dal.Models.ResponseModels;
using Dapper;
using Dapper.Contrib.Extensions;
using System.Collections.Generic;
using System.Data;
using System.Threading.Tasks;

namespace Dal
{
   public class OrderRepository : IOrderRepository
   {
      private IDbConnection Connection { get; }
      private IErrorManagerRepository<ErrorOrderModel> _errorAllowed { get;
set; }
      private IErrorManagerRepository<ErrorOrderModel> _orderInfo { get; set; }
      private IErrorManagerRepository<ErrorOrderModel> _acceptOrder { get; set;
}

      public OrderRepository(IDbConnection connection,
IErrorManagerRepository<ErrorOrderModel> errorAllowed,
         IErrorManagerRepository<ErrorOrderModel> orderInfo,
IErrorManagerRepository<ErrorOrderModel> acceptOrder)
      {
         Connection = connection;
```

```
        _errorAllowed = errorAllowed;
        _orderInfo = orderInfo;
        _acceptOrder = acceptOrder;
    }
    public async Task<ResponseModelBase<ErrorOrderModel>>
AcceptOrder(AcceptOrder acceptOrderForm)
    {
        _acceptOrder.model.Id = acceptOrderForm.Id;
        _acceptOrder.model.SpecialistId = acceptOrderForm.SpecialisId;
        var valid = await _acceptOrder.DetectinObjDB(_acceptOrder.model);
        if (valid.DetectinObj == true)
        {
            using (var db = Connection)
            {
                var sqlQuery = @"EXEC [dbo].[AcceptOrder]@Id, @SpecialistId";
                await db.ExecuteAsync(sqlQuery, new { Id = acceptOrderForm.Id ,
                    SpecialistId = acceptOrderForm.SpecialisId
                });
                return new
ResponseModelBase<ErrorOrderModel>(_acceptOrder.model,
                    StatusCodes.StatusCode.OK,
                    "Order was accepted");
            }
        }
        else
        {
            return new
ResponseModelBase<ErrorOrderModel>(_acceptOrder.model,
                    StatusCodes.StatusCode.ServiceUnavailable,
                    "Object is not found");
```

```
        }
    }
    public async Task<ResponseModelBase<ErrorOrderModel>>
ChangeUserInfoOrder(ChangeUserInfoOrder changeSpecialistInfoOrder)
    {
        using (var db = Connection)
        {
            var sqlQuery = @"EXEC [dbo].[ChangeUsersInfoOrder]@Id,
@UserName, @Surname, @PhoneNumber, @Email, @NormalizedUserName,
@NormalizedEmail, @InstagramLink, @FacebookLink";
                await db.ExecuteAsync(sqlQuery, new
            {
                Id = changeSpecialistInfoOrder.Id,
                UserName = changeSpecialistInfoOrder.UserName,
                Surname = changeSpecialistInfoOrder.Surname,
                PhoneNumber = changeSpecialistInfoOrder.PhoneNumber,
                Email = changeSpecialistInfoOrder.Email,
                NormalizedUserName =
changeSpecialistInfoOrder.UserName.ToUpper(),
                NormalizedEmail = changeSpecialistInfoOrder.Email.ToUpper(),
                InstagramLink = changeSpecialistInfoOrder.InstagramLink,
                FacebookLink = changeSpecialistInfoOrder.FacebookLink,
                });
            return new
ResponseModelBase<ErrorOrderModel>(StatusCodes.StatusCode.OK,
                "Specialist's info was successfully incremented");
        }
    }
    public async Task<IEnumerable<OrderInfo>> GetAllOrders()
    {
```

```
        using (var db = Connection)
        {
            return await db.QueryAsync<OrderInfo>(@"EXEC
[dbo].[GetAllOrders]");
        }
    }

    public async Task<OrderInfo> CreatedOrder(OrderInfo createdOrder)
    {
        using(var db = Connection)
        {
            var identity =  await db.InsertAsync(new OrderInfo
            {
                Id = createdOrder.Id,
                Name = createdOrder.Name,
                Details = createdOrder.Details,
                City = createdOrder.City,
                Address = createdOrder.Address,
                Fulfillment = createdOrder.Fulfillment,
                Price = createdOrder.Price,
                PaymentMethod = createdOrder.PaymentMethod,
                ClientId = createdOrder.ClientId,
                SpecialistId = createdOrder.SpecialistId,
            });

        }


        return createdOrder;
    }
```

```
public async Task<OrderInfo> GetOrder(int Id)
{
    _orderInfo.model.Id = Id;
    var valid = await _orderInfo.DetectinObjDB(_orderInfo.model);
        using (var db = Connection)
        {
            return await db.GetAsync<OrderInfo>(Id);
        }
}
public async Task<UserOrderInfo> GetUserOrder(string Id)
{
    _orderInfo.model.SpecialistId = Id;
    var valid = await _orderInfo.DetectinObjDB(_orderInfo.model);
    using  (var db = Connection)
    {
        return await db.GetAsync<UserOrderInfo>(Id);
    }
}
public async Task<ResponseModelBase<ErrorOrderModel>>
DeleteOrder(DeletetOrder deletetOrder)
{
    _errorAllowed.model.Id= deletetOrder.Id;
   var valid= await _errorAllowed.DetectinObjDB(_errorAllowed.model);
    if (valid.DetectinObj == true)
    {
        using (var db = Connection)
        {
            var sqlQuery = @"EXEC [dbo].[DeleteOrder] @Id";
            await db.ExecuteAsync(sqlQuery, new {Id = deletetOrder.Id });
        }
```

```
            return new
ResponseModelBase<ErrorOrderModel>(_errorAllowed.model,
StatusCodes.StatusCode.OK,
                "Object was successfully deleted");
        }
        else
        {
            return new
ResponseModelBase<ErrorOrderModel>(_errorAllowed.model,
                StatusCodes.StatusCode.ServiceUnavailable,
                "Object is not found");
        }
    }

    public async Task<ResponseModelBase<ErrorOrderModel>>
AddViewsOrder(AddViewsOrder addViewsOrder)
    {
        using (var db = Connection)
        {
            var sqlQuery = @"EXEC [dbo].[AddViewsOrder] @Id";
            await db.ExecuteAsync(sqlQuery, new { Id = addViewsOrder.Id });
            return new
ResponseModelBase<ErrorOrderModel>(StatusCodes.StatusCode.OK,
                "Views was successfully incremented");
        }
    }
    public async Task<ResponseModelBase<ErrorOrderModel>>
AddRatingOrder(AddRatingOrder addRatingOrder)
    {
        using (var db = Connection)
```

```
        {
            var sqlQuery = @"EXEC [dbo].[AddRatingOrder]@Id, @TotalClick,
@TotalPoint, @Rating";
            await db.ExecuteAsync(sqlQuery, new
            {
                Id = addRatingOrder.Id,
                TotalClick = addRatingOrder.TotalClick,
                TotalPoint = addRatingOrder.TotalPoint,
                Rating = addRatingOrder.TotalPoint / addRatingOrder.TotalClick
            });
            return new
ResponseModelBase<ErrorOrderModel>(StatusCodes.StatusCode.OK,
                "Rating was successfully incremented");
        }
    }

    public Task<IEnumerable<OrderInfo>> GetOrdersByClient(string userId)
    {
        var sqlQuery = @"SELECT o.*, users.*
                        FROM Orders o
                        JOIN AspNetUsers users on users.Id = o.ClientId
                        WHERE ClientId = @ClientId
                        ";
        return Connection.QueryAsync<OrderInfo, UserOrderInfo,
OrderInfo>(sqlQuery, (o, users) =>
        {
            if (o.User == null)
                o.User = users;

            return o;
```

```
        }, new { ClientId = userId });

    }

    public Task<IEnumerable<OrderInfo>> GetOrdersBySpecialist(string
specialistId)
    {
        var sqlQuery = @"SELECT o.*, users.*
                    FROM Orders o
                    JOIN AspNetUsers users on users.Id = o.ClientId
                    WHERE SpecialistId = @SpecialistId
                    ";
        return Connection.QueryAsync<OrderInfo, UserOrderInfo,
OrderInfo>(sqlQuery, (o, users) =>
        {
            if (o.User == null)
                o.User = users;

            return o;
        }, new { SpecialistId = specialistId });
    }
  }
}
```

ДОДАТОК Г

Лістинг інтерфейсу IOrderRepository

```
using Dal.Models.Order;
using Dal.Models.Order.ErrorOrderModel;
using Dal.Models.ResponseModels;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Dal.Contracts
{
   public interface IOrderRepository
   {
      Task<OrderInfo> CreatedOrder(OrderInfo createdOrder);
      Task<ResponseModelBase<ErrorOrderModel>> AcceptOrder(AcceptOrder
acceptOrder);
      Task<IEnumerable<OrderInfo>> GetAllOrders();
      Task<IEnumerable<OrderInfo>> GetOrdersBySpecialist(string Id);
      Task<IEnumerable<OrderInfo>> GetOrdersByClient(string Id);
      Task<OrderInfo> GetOrder(int Id);
      Task<UserOrderInfo> GetUserOrder(string Id);
      Task<ResponseModelBase<ErrorOrderModel>> DeleteOrder(DeletetOrder
deletetOrder);
      Task<ResponseModelBase<ErrorOrderModel>>
AddViewsOrder(AddViewsOrder addViewsOrder);
      Task<ResponseModelBase<ErrorOrderModel>>
AddRatingOrder(AddRatingOrder addRatingOrder);
      Task<ResponseModelBase<ErrorOrderModel>>
ChangeUserInfoOrder(ChangeUserInfoOrder changeSpecialistInfoOrder);
   }
}
```

ДОДАТОК Д

Структура бази даних

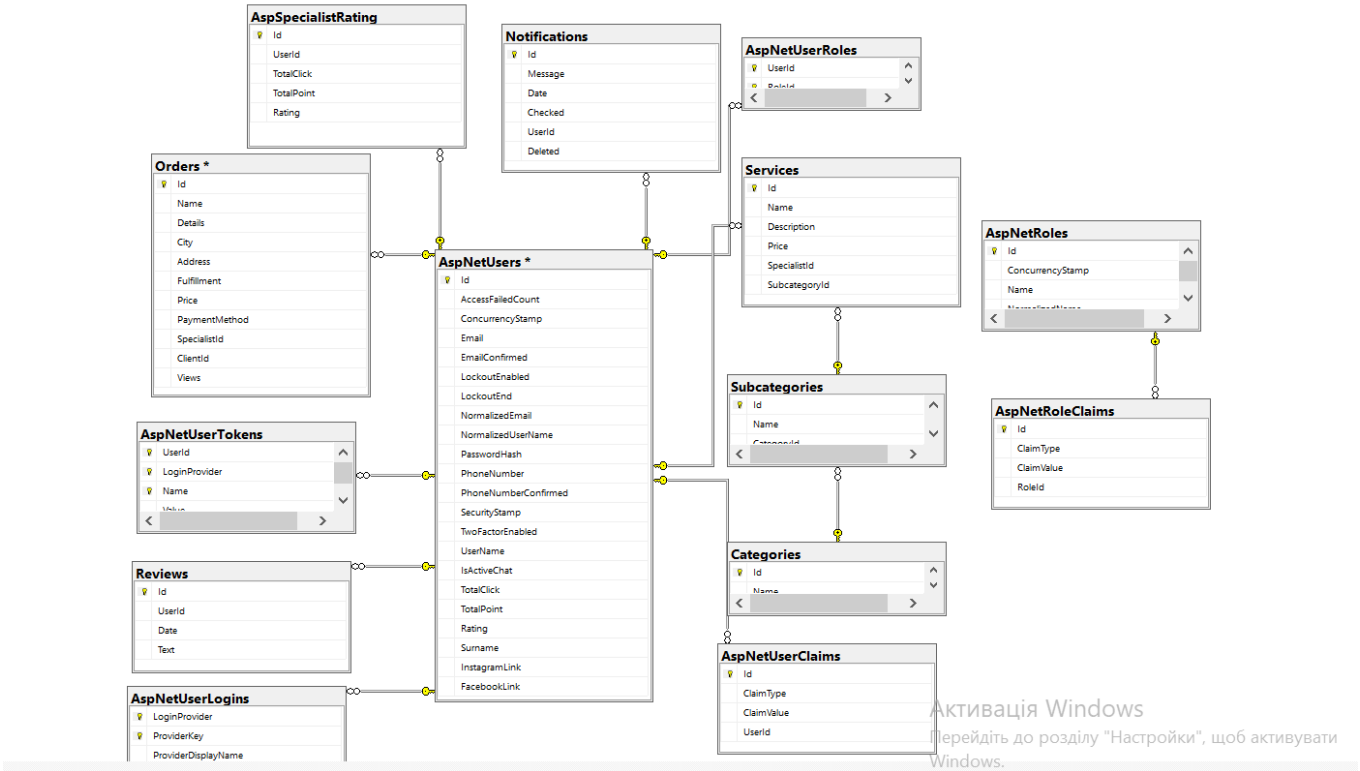

Рисунок Д—Структура бази даних

ДОДАТОК Е

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Веб-сервіс для надання послуг

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 95,0% Схожість 5,0%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Кошельник В. О.

Керівник роботи _____ Колесник І. С.