**Пояснювальна записка  
до курсової роботи**

на тему: Веб-застосунок система пошуку виконавців та замовників послуг

КПІ.ІП-1310.045440.02.81

Київ – 2024

**Зміст**

[1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 5](#_heading=h.1fob9te)

[1.1 Загальні положення 5](#_heading=h.3znysh7)

[1.2 Змістовний опис і аналіз предметної області 6](#_heading=h.2et92p0)

[1.3 Аналіз існуючих технологій та успішних IT-проєктів 8](#_heading=h.3dy6vkm)

[1.3.1 Аналіз відомих алгоритмічних та технічних рішень 10](#_heading=h.1t3h5sf)

[1.3.2 Аналіз допоміжних програмних засобів та засобів розробки 13](#_heading=h.2s8eyo1)

[1.3.3 Аналіз відомих програмних продуктів 14](#_heading=h.17dp8vu)

[1.4 Аналіз вимог до програмного забезпечення 16](#_heading=)

[1.4.1 Розроблення функціональних вимог 29](#_heading=h.35nkun2)

[1.4.2 Розроблення нефункціональних вимог 31](#_heading=h.44sinio)

[1.5 Постановка задачі 31](#_heading=h.z337ya)

[Висновки до розділу 32](#_heading=h.1y810tw)

[2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 34](#_heading=h.4i7ojhp)

[2.1 Моделювання та аналіз програмного забезпечення 34](#_heading=h.2xcytpi)

[2.2 Архітектура програмного забезпечення 36](#_heading=h.3whwml4)

[2.3 Конструювання програмного забезпечення 39](#_heading=h.qsh70q)

[2.3.1 Алгоритми 39](#_heading=)

[2.3.2 Структури даних, програмні структури 41](#_heading=)

[2.3.3 Структура бази даних 43](#_heading=)

[2.3.4 Використані бібліотеки та утиліти 50](#_heading=)

[2.3.5 Аналіз системних вимог 51](#_heading=)

[2.4 Аналіз безпеки даних 51](#_heading=h.3as4poj)

[Висновки до розділу 52](#_heading=h.49x2ik5)

[3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 54](#_heading=h.2p2csry)

[3.1 Аналіз якості ПЗ 54](#_heading=h.147n2zr)

[3.2 Опис процесів тестування 55](#_heading=h.23ckvvd)

[3.3 Опис контрольного прикладу 63](#_heading=h.ihv636)

[Висновки до розділу 76](#_heading=h.32hioqz)

[4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 77](#_heading=h.1hmsyys)

[4.1 Розгортання програмного забезпечення 77](#_heading=h.41mghml)

[4.2 Підтримка програмного забезпечення 78](#_heading=h.vx1227)

[Висновки до розділу 78](#_heading=h.1v1yuxt)

[ВИСНОВКИ 79](#_heading=h.4f1mdlm)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 80](#_heading=h.2u6wntf)

**Перелік умовних позначень**

|  |  |  |
| --- | --- | --- |
| IDE | – | Integrated Development Environment – інтегроване середовище розробки. |
| API | – | Application programming interface, прикладний програмний Інтерфейс |
| SDK | – | Software development kit |
| IT | – | Інформаційні технології |
| ER | – | Entity-Relation diagram |
| OC | – | Операційна система. |
| БД | – | База даних. |
| ORM | – | Object-Relational Mapping |
| СУБД | – | Система управління базами даних |
| CORS | – | Cross-Origin Resource Sharing |
| FK | – | Foreign Key |

**Вступ**

Сучасний світ переживає значні зміни через розвиток технологій, особливо в сфері надання та отримання послуг. Це породило потребу в системах, які полегшують процес пошуку та замовлення послуг. Така потреба підкреслює актуальність цієї роботи, яка полягає в створенні ефективної системи для цих цілей.

Глобальні тенденції свідчать про збільшення попиту на платформи, що дозволяють користувачам самостійно описувати та пропонувати свої послуги. Проте, багато з цих платформ вимагають від користувачів надання особистих даних для реєстрації, що може обмежувати їхню приватність.

Дана розробка відрізняється тим, що не вимагає від користувачів надання особистих даних для реєстрації, що забезпечує більшу приватність. Вона також дозволяє користувачам самостійно описувати та пропонувати будь-які послуги, що забезпечує більшу гнучкість. Крім того, дана система надає користувачам можливість вибрати метод комунікації, що забезпечує більшу зручність.

Отже, ця розробка представляє новий підхід до системи пошуку та замовлення послуг, який забезпечує поєднання користувачів, враховуючи їхні вимоги та уміння.

# АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Загальні положення

У історії людства як виду та як суспільства ключовим фактором виживання є об’єднання та взаємодопомога. У природі в цілому, уникнення змагання та перехід до суспільності та взаємна допомога сприяє кращому виживанню виду [1]. Так само як об’єднання тварин у групи, наприклад, стада ссавців, косяки риб, зграя птахів, люди теж почали об’єднуватись сім’ями у общини, племена, для накопичення зусиль заради виживання усіх та кожного по окремості.

Проте постійний прогрес як технологічний, так і економічний, змушує нас до виконання нових задач, які тепер мають на меті не виконання різноманітних інстинктивних дій задля виживання, а отримання певних обмежених кваліфікованих навичок, які необхідні для функціонування сучасного суспільства. Таким чином, з постійним ростом популяції суспільства, ми все менше зосереджені на прямій допомозі іншим людям в невеликому колі, і все більше на виборі власних навичок, сильних сторін, і постійного їх застосування в наданнях послуг людям з широкого кола суспільства.

Від раніше згаданої взаємодопомоги ця модель виконавців та замовників відрізняється комерційною складовою: виконавець концентрується на власних кваліфікаціях, їх поширенню серед людей для пошуку потенційних замовників, укладання угод на виконання з ними, отримуючи взамін вигоду, найчастіше грошову.

У світі, де давно переважає наймана праця, складніше зарекомендувати себе як вільного виконавця, продемонструвати себе, свої уміння та навички. З появою Інтернету та пошукових систем можливість поширення інформації на широкий загал стала як ніколи доступною – від персональних сайтів до централізованих платформ пошуку виконавців. Остання модель є найпоширенішою, за рахунок пришвидшення та зручності взаємодії виконавця та потенціального замовника.

Період пандемії вірусу COVID-19 вплинув на різноманітні аспекти життя. За даними дослідження, проведеного Harvard Business School, пандемія COVID-19 спричинила значне зростання віддаленої фріланс роботи. [2]. В умовах ізоляції Інтернет набув ще більшу роль в житті кожного, в тому числі й пошуку послуг та їх виконавців. Тому потреба в платформах, де виконавці можуть знайти необхідні послуги, та дійти до угоди щодо надання цих послуг стала як ніколи великою. Маючи велику аудиторію та можливість представлення власних послуг, люди звикли до такої моделі праці, і попри те, що у травні 2023 Всесвітня Організація Здоров’я (ВОЗ) заявила про скасування статусу пандемії коронавірусу [3], суспільство не перестало користуватись даними платформами пошуку послуг, а й надалі продовжують виконувати замовлення інших користувачів на платформах, що каже про високу актуальність розробки у цій сфері: за даними Freelancermap, цифровізація та вплив кризи 2020-х років залишили тривалі наслідки в світі праці. Тенденція явно зміщується в бік гіг-економіки та фрілансу [4].

Щодо напрямків розробки, веб-додатки є найпопулярнішим у сфері пошуку послуг, в силу найвищої доступності, відсутності необхідності встановлення окремих додатків на власний пристрій та можливості користування на різних платформах. Навіть у великих системах, що мають різноманітні клієнтські додатки на мобільні, планшетні та стільникові платформи, веб-версія користується найбільшим попитом.

## Змістовний опис і аналіз предметної області

В галузі розробки веб-застосунків для пошуку та надання різноманітних послуг виявляються певні виклики та проблеми, які можна узагальнити для аналізу та подальшого вдосконалення.

В ході конкуренції компанії, намагаючись перевершити одна одну, можуть ставити акцент на впровадження більше різноманітного функціоналу. Це може включати нові можливості, інструменти та сервіси, призначені для розширення функціональності їхніх веб-застосунків.

Проте, в процесі такої конкуренції іноді може виникати проблема неврахування балансу між розширенням функціональності та забезпеченням простоти використання для кінцевого користувача. Компанії можуть бути спритні у впровадженні нових функцій, проте це може призвести до збільшення складності інтерфейсу та ускладення процесів взаємодії з платформою. Це ускладнення означатиме нові виклики для кінцевих користувачів, погіршення досвіду використання веб-застосунку

Проблеми, пов’язані з взаємодією користувачів, неоптимальним розподілом ресурсів та неадаптованістю до змін в попиті можуть впливати на якість та ефективність веб-застосунків у даній предметній області. Однак, аналізуючи можливі плюси, варто визначити, що анонімність при реєстрації та простота процесу пошуку на сайті становлять потенційні переваги.

Анонімність при реєстрації може слугувати додатковим мотиватором для користувачів, які цінують конфіденційність своїх особистих даних. Простий процес пошуку сприяє зручності використання платформи, що може призвести до підвищення задоволеності користувачів та покращення їхнього досвіду використання.

У контексті поліпшення веб-застосунку для пошуку та надання послуг, надання можливості користувачам вказувати контактні дані для спілкування на інших платформах, такі як номер мобільного телефону чи ім’я користувача у месенджерах, може стати значущою перевагою для тих, хто вважає конфіденційність та зручність важливими аспектами.

Ця можливість дозволяє користувачам обирати спосіб зв’язку, який їм зручний, і взаємодіяти з виконавцями чи замовниками на більш особистому рівні, маючи впевненість в конфіденційності та надійності розмови. Вказання контактів на зовнішніх платформах може сприяти ефективному та надійному обміну інформацією та узгодженню деталей безпосередньо в тому середовищі, яке є зручним для кінцевого користувача.

Реалізація перелічених переваг зможе забезпечити кінцевим користувачам достатньо функціональний та зручний додаток для пошуку і надання послуг, що цінує конфіденційність користувачів та їх бажання використовувати обрані надійні платформи для зв’язку.

## Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації системи пошуку Dealoviy. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

Розглянемо для початку веб-сайт ManGoWorks. ManGoWorks – це українська платформа для пошуку виконавців послуг різного діапазону. [5].

Для реєстрації сайт пропонує вхід через обліковий запис Google, або введення особистих даних в систему, таких як ім’я та прізвище, номер телефону, дата народження, електронну пошту, а також вказати, чи реєструється фізична особа чи юридична.

Для пошуку виконавця, що задовольнить кінцевого користувача на сайті доступні різні варіанти.

Варіант вибору переходу до етапу «Створити завдання» з верхньої частини головної сторінки чи хедеру включає в себе в першу чергу багаторівневу категоризацію замовлення: спочатку обрати категорію, потім підкатегорію, у підкатегорії можуть бути власні підкатегорії, під які динамічно з’являються нові меню на формі, також, після усіх меню з обраними категоріями, очікує меню вибору міста, що включає в себе три опції: Дніпро, Київ, Харків. Після визначення усіх категорій, підкатегорій та міста користувача очікує введення назви та опис завдання, строк виконання замовлення або гнучку дату, орієнтовний бюджет чи вказати можливість вільної ціни, а також ввести контактні дані, чи використати контактні дані з авторизованого профіля, що включають в себе сталий набір полів: вибір фізичної чи юридичної особи, ім’я, номер телефону та електронну пошту – причому усі поля одночасно є обов’язковими. Останнім обов’язковим етапом є згода з правилами сервісу. Далі оголошення буде опубліковане на загальну платформу виконавців, які надсилатимуть пропозиції, і користувач обирає з надісланих пропозицій, судячи з рейтингу та запропонованою виконавцем ціною.

Іншим варіантом є пошук потрібної категорії серед багатьох на головній сторінці, після чого користувач бачитиме сторінку, наповнену інформацією та елементами різних видів. Наприклад, дві кнопки на верхній частині сайту – «Створити завдання» та «Створити оголошення». Форми, на які перенаправляють користувача ці кнопки, ідентичні, структура їх така ж, як описана вище, та інструкція біля форм однакова. Різниця між завданнями та оголошеннями не вказується на даній сторінці чи на формах створення завдань та оголошень. При першому візиті на сайт користувачу може бути незрозуміло, як краще йому опублікувати власну потребу на виконання певної задачі.

Окрім цього, нижче на сторінці категорії розміщено блок з найкращими виконавцями в поточній категорії, де в кожного виконавця відображається ім’я, статистика в вигляді кількості позитивних та негативних . В цьому блоці запропонувати роботу – третя форма, що має ідентичну попереднім форму, з такою ж інструкцією.

У висновку, для реєстрації на сайті ManGoWorks запрошується чимала кількість особистих даних, а алгоритм дій для публікації завдання не є до кінця зрозумілим.

Наступним IT-проектом на огляді стане Freelancer.com. Freelancer є широко визнаною та глобальною платформою, яка об’єднує замовників та фрілансерів. [6].

При реєстрації на Freelancer.com користувач може вибрати між реєстрацією через обліковий запис Facebook або пройти форму реєстрації. Форма реєстрації включає електронну пошту або юзернейм – добре, що в користувача на старті є вибір в сторону конфіденційності.

Щодо механізму опису своєї задачі, Freelancer дозволяє створити проект, описати його, обрати категорію та визначити бюджет. Також Freelancer пропонує функцію пошуку фрілансерів на основі категорій, навичок та рейтингів. В такій моделі фрілансери подають пропозиції на виконання. Для комунікації з фрілансером замовник використовуватиме внутрішню систему комунікації. Така імплементація вимагає великих ресурсів і зусиль на забезпечення конфіденційності і цілісності даних розмови, і користувачі можуть мати різний рівень довіри до комунікації на сайті, незалежно від рівня безпеки та захищеності даних.

В загальному, видно, що Freelancer – сайт, в реалізацію якого було вкладено багато думок, і конфіденційність користувача цінується на етапі реєстрації.

В підсумку бачимо, що незалежно від масштабу і вкладених в розробку ресурсів не кожен IT-проект вважає критичною потребою завдоволення можливості конфіденційно користуватись сайтом, та алгоритм використання кінцевим користувачем повинен розроблятись докладніше. До того ж, незалежно від наявності власної платформи комунікації, в цілях тої ж конфіденційності було б добре надати можливість спілкуватись на інших платформах комунікації.

### Аналіз відомих алгоритмічних та технічних рішень

Серед існуючих рішень в сфері пошуку послуг та їх надання маємо різні підходи, алгоритми та рішення задля «зустрічі» та потенційоного проведення угоди між замовником та виконавцем.

Публікація замовлень на загальний майданчик для виконавців: найпоширенішим алгоритмом пошуку замовником виконавця, що зможе вирішити його проблему, є створення замовником власного замовлення з описом проблеми, вибором категорії завдання та іншою інформацією на спільну платформу для виконавців, які розглядатимуть ці публікації та вігукуватись на них. Також зустрічається можливість публікації замовлення, яке буде напряму надіслане обраному виконавцю. Переглянувши ТОП-5 фріланс-бірж 2020 року, можна помітити, що найбільші та найстаріші з актуальних платформ на момент написання використовують таку модель [7]. Такий підхід дає замовникам можливість бачити зацікавлених в виконанні власного завдання, проте при реалізації такого алгоритму пошуку виконавця та замовника варто утримувати баланс в функціональності та зручності і інтуїтивного використання користувачем.

Пошук послуг на платформі та залишення заявок на послугу: іншим можливим підходом є пошук замовником послуг за заданими критеріями, вибір найбільш підходящої послуги за описом, рейтингом та подання заявки на неї. Цей підхід нагадує підхід, за яким виконавці знаходять своїх замовників за межами Інтернету: як приклад, в телевізійних новинах, чи газетах є блок реклами, що виступає платформою для публікації, що змушує виконавця бути зацікавленим в тому, щоб замовники побачили оголошення про його послугу, а не навпаки. Виконавець демонструє свої навички на платформі, а виконавці оцінюють їх, та роблять вибір щодо звернення до даного виконавця за наданими контактними даними. Як приклад, компанія SmartTender – офіційний майданчик державних тендерів Prozorro – має платформу, на якій демонструє потенційним клієнтам різні тарифні плани, в які включені ті чи інші види тендерів на закупівлю. Для детальнішої інформації та подання заявки компанія надає контактні дані гарячої лінії та посилання на сторінку для заповнення заявок [8]. Такий підхід, дає можливість зробити вибір найоптимальнішої послуги, власноруч оцінивши та порівнявши продемонстровані навички та показники виконавців.

Розділення на категорії: ще одним поширеним принципом серед систем пошуку та надання послуг є категоризація послуг, замовлень тощо по предметній області. Такий підхід дає можливість розділяти інформацію на сайті по критерію напрямку, предметної області послуги, проте зі зростанням аудиторії та кількості послуг, що вони надають – структура цих категорій збільшується та навіть йде у підкатегорії та під-підкатегорії, що ускладнює організацію даних в них та може доставити незручності у використанні кінцевим користувачем.

Демонстрація рейтингу: це рішення, що дозволяє замовникам оцінити якість наданої виконавцем послуги, аби порекомендувати чи, навпаки, застережити інших замовників про виконавця даної послуги. Це дозволяє замовнику зробити легший вибір серед кількох знайдених послуг, що зближує виконавця та замовника до потенційно укладеної угоди.

Комунікація користувачів безпосередньо на сайті: рішення, що притаманне великим платформам, забезпечує можливість моментального спілкування при знаходженні на даному сайті. Але розробка такого рішення вимагає чіткості в вимогах, ресурсів, і вхід на сайт для перевірки наявності нових повідомлень може не завжди бути найзручнішим та найшвидшим варіантом для кінцевого користувача.

Проаналізувавши усі ці алгоритми та технічні рішення, в цілях забезпечення конфіденційності користувачів та створення простого і зрозумілого процесу знаходження особи для вирішення власної задачі, було обрано підхід публікації послуг виконавцями та їх пошук замовниками. Цей підхід є найменш вимогливим для користувача-замовника, який тільки що зареєструвався або давно є користувачем сайту, бо алгоритм дій прямолінійний, без розгалужень та великої кількості кроків. Таким чином, обравши оптимальну послугу, користувач лишатиме заявку саме на неї. Задля комунікації виконавця та потенційного замовника виконавець має внести один або декілька варіантів зв’язку з ним, і при перегляді сторінки послуги замовники не бачитимуть власне контактних даних виконавця, а радше методи зв’язку з ним. При створенні заявки замовник обиратиме один з доступних методів зв’язку і вводитиме свої контактні дані для цього методу, що їх побачить лиш виконавець на поданій заявці. Замовник же на даній заявці побачить контактні дані замовника, того методу зв’язку, який він обрав. Також користувачі-замовники матимуть можливість оцінити роботу виконавця на сайті, тим самим вплинути на його рейтинг належним чином та допомогти іншим користувачам зробити свій вибір на користь того чи іншого виконавця.

### Аналіз допоміжних програмних засобів та засобів розробки

Сфера розробки веб-додатків налічує величезну кількість інструментів, програм, бібліотек, фреймворків для розробки як для серверної, так і клієнтської частин.

Для розробки серверної частини було обрано платформу .NET, що включає сімейство мов програмування та інструментів, проте саме розробка на мові C# на даній платформі відрізняється багатим досвідом розробника, можливістю встановлення пакетів, розроблених іншими розробниками завдяки системі NuGet packages - платформа, яка налічує більше 400 мільярдів завантажень пакетів [9]; компіляцією та швидким виконанням; платформа має постійно розширювану підтримку як від компанії Microsoft, так і спільноти розробників, оскільки інструменти розробки від Microsoft мають відкритий доступ до вихідного коду. Функціонал для реалізації прикладного програмного інтерфейсу (API) веб-серверу забезпечує фреймворк ASP.NET Core – він має достатній та багатий інструментарій для створення веб-сервера, легкого опису обробки запитів ендпоінтами. Для роботи з базою даних використовується фреймворк Entity Framework Core, що забезпечує підтримку під’єднання до баз даних різного виду, роботу з записами з бази даних в коді за допомогою ORM, багатий функціонал створення LINQ-запитів для отримання, фільтрації, сортування даних замість використання SQL-запитів.

Для даної розробки в якості СУБД було обрано Microsoft SQL Server через його інтеграцію з іншими продуктами розробки Microsoft, високі стандарти надійності, продуктивності та широкі можливості маніпуляції даними.

Для виконання читання даних про області та міста з файлу (дані про географічні області та міста взяті з джерела SimpleMaps [10]) з розширенням .csv для подальшого завантаження даних в БД було обрано програмний пакет з платформи NuGet під назвою CsvHelper [11]. Даний програмний пакет надає зручні інструменти для ефективного читання файлів з розширенням .csv та можливість отримання конкретних даних з рядку за назвою колонки.

Для фронтенд-розробки було обрано фреймворк Vue, що відомий своєю простотою в освоєнні, швидкою роботою, наявністю усіх необхідних інструментів для початку роботи в базових модулях. Простий, функціональний та швидкий – це робить Vue прекрасним вибором для розробки клієнтського додатку в веб-застосунку.

Для безпосереднього процесу розробки коду необхідні різні програмні засоби, в першу чергу – інтегровані середовища розробки (IDE), це є ключовим компонентом в зручності та швидкості розробки продукту.

Для розробки бекенд-частини застосунку було обрано IDE JetBrains Rider, що має переваги для розробників усіх рівнів: Rider працює швидше за Visual Studio [12], Rider, як і усі програмні продукти компанії JetBrains, має зручний досвід користування, можливість встановлення плагінів, та займає менший об’єм пам’яті, що займає на диску, включно з додатковими інструментами, плагінами для розробки.

У фронтенд-розробці було обрано Visual Studio Code – текстовий редактор з можливістю розширення різноманітними плагінами для розробок у різних цілях – є найкращим варіантом для розробки в сфері фронтенд: легкий, швидкий, високо розширюваний, має багатий набір офіційних плагінів для роботи з різними мовами програмування та фреймворками.

### Аналіз відомих програмних продуктів

Розглянемо функціонал давно існуючих рішень у сфері пошуку послуг, та проведемо порівняльний аналіз основних функціональних можливостей між курсовою роботою та такими аналогами, як ManGoWorks та FreeLancer.

Для порівняння курсової роботи з аналогами можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функціонал | Курсова робота (Dealoviy) | ManGoWorks | Freelancer | Пояснення |
| Анонімна система реєстрації | + | - | + | Реєстрація на сайті не вимагає введення персональних даних, таких як ім’я, прізвище, електронна пошта, номер телефону, поштовий індекс |
| Пошук послуг за різними критеріями | + | + | + | Можливість пошуку послуг за ключовим словом та фільтрами |
| Перегляд рейтингу та оглядів інших від інших користувачів | + | + | + | Користувачі можуть переглядати відгуки від інших користувачів на роботу виконавця. |
| Можливість стати виконавцем на сайті | + | + | + | Механізм для отримання ролі виконавця, можливості надавати послугу чи виконувати завдання інших користувачів |
| Вибір бажаного методу комунікації з виконавцем | + | - | - | Користувачеві надається вибір зв’язку з виконавцем, який вважає для себе необхідним |
| Можливість вільного вибору укладання угоди з різними користувачами | + | + | + | Свобода вибору для замовників та для виконавців щодо виконання певного замовлення |
| Управління станом замовлень | + | + | + | Оновлення стану замовлення |

Додатково наведу порівняння алгоритмів укладення угоди на аналозі ManGoWorks та власній розробці Dealoviy:

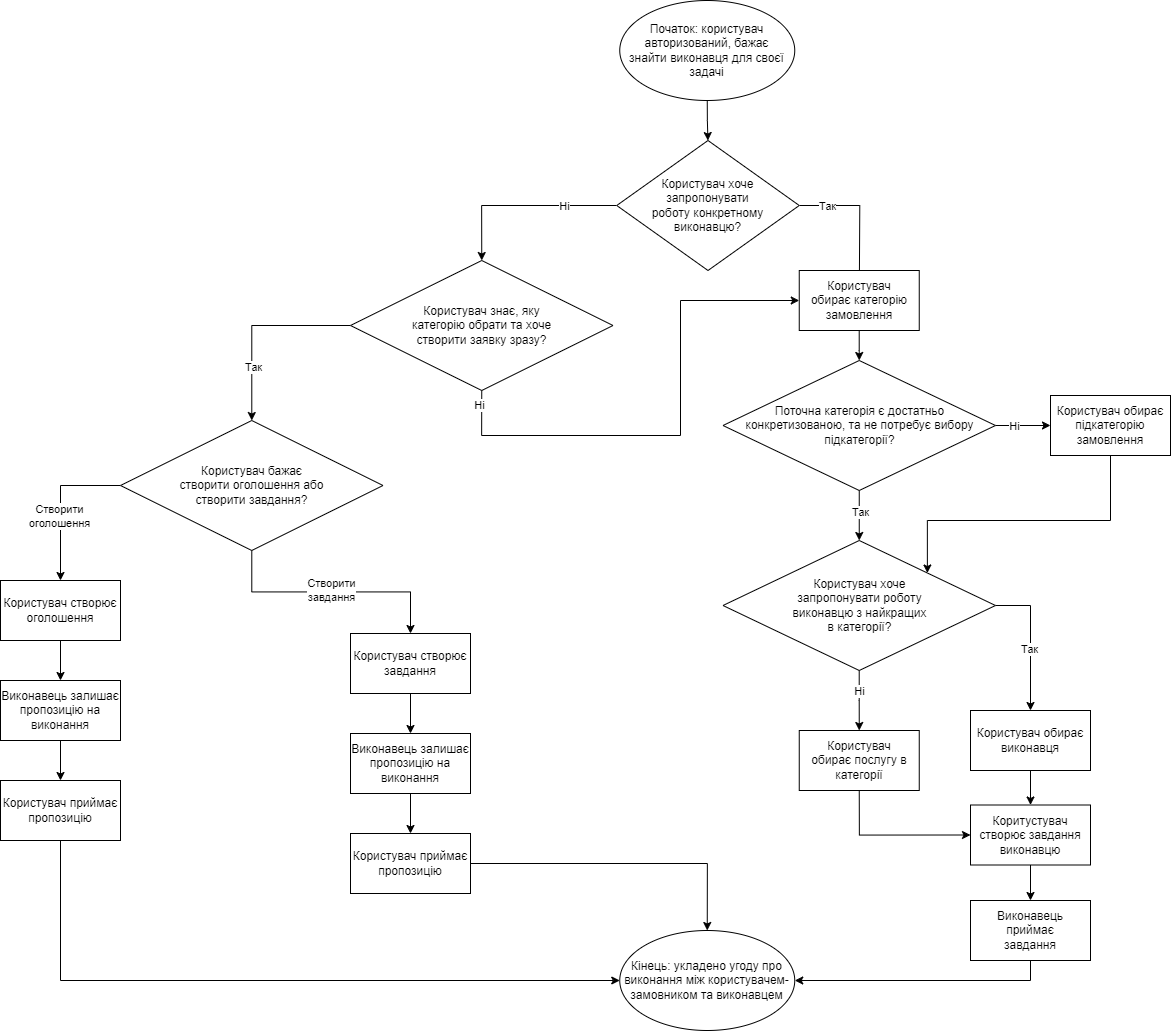


Рисунок 1.1 – алгоритм укладення угоди на ManGoWorks



Рисунок 1.2 – алгоритм укладення угоди на Dealoviy

Як бачимо, алгоритм аналогу містить багато ситуацій вибору для користувача, що виникають через різні напрямки створення заявок, різні види заявок, поділ на категорії та підкатегорії – усе це створює розгалуження і додає інформаційне навантаження для користувача сайту.

Розробка курсової роботи - навпаки, фокусується не на широкому виборі напрямків, а балансуванню і оптимізації єдиного шляху, який користувач зможе одразу зрозуміти і запам’ятати.

## Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є пошук послуги, що найкраще підходить для вирішення задачі замовника, більше функцій можна побачити на рисунку 1.3.

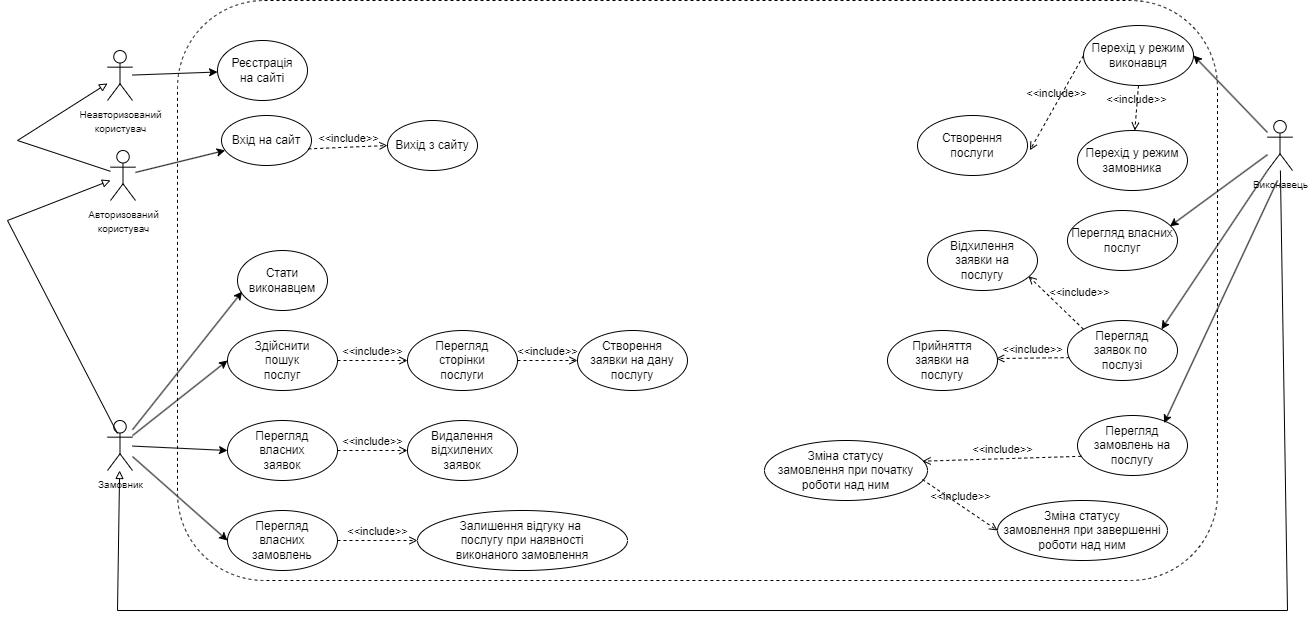


Рисунок 1.3 – Діаграма варіантів використання

В таблицях 1.2 - 1.23 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

|  |  |
| --- | --- |
| Use case name | Реєстрація на сайті |
| Use case ID | UC-01 |
| Goals | Реєстрація нового користувача в системі |
| Actors | Неавторизований користувач |
| Trigger | Користувач бажає зареєструватися |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: унікальне ім’я користувача в системі, ім’я для відображення на сайті (опціонально), пароль. Після заповнення даних користувач натискає кнопку реєстрації. Після цього користувач перенаправляється на головну сторінку. |
| Extension | - |
| Post-Condition | Створення сторінки користувача, перехід на головну сторінку |

Таблиця 1.3 - Варіант використання UC-2

|  |  |
| --- | --- |
| Use case name | Реєстрація на сайті |
| Use case ID | UC-02 |
| Goals | Вхід користувача у систему |
| Actors | Авторизований користувач |
| Trigger | Користувач бажає авторизуватися |
| Pre-conditions | Користувач знаходиться на сторінці реєстрації |
| Flow of Events | Користувач переходить на сторінку входу в систему. В поля для авторизації вводяться відповідні дані: унікальне ім’я користувача в системі, та пароль. Після заповнення даних користувач натискає кнопку входу. Після цього користувач перенаправляється на головну сторінку |
| Extension | - |
| Post-Condition | Авторизація користувача, перехід на головну сторінку |

Таблиця 1.4 - Варіант використання UC-3

|  |  |
| --- | --- |
| Use case name | Здійснити пошук послуг |
| Use case ID | UC-03 |
| Goals | Видача користувачеві пошукових результатів по обраним критеріям |
| Actors | Замовник |
| Trigger | Користувач бажає здійснити пошук послуг |
| Pre-conditions | Користувач авторизований та знаходиться на головній сторінці |
| Flow of Events | Користувач знаходиться на головній сторінці. В поле текстового запиту користувач вводить ключовий запит для пошуку послуг по назві, після чого в першому меню обирає географічну область, далі в наступному меню обирає місто. Після заповнення даних користувач натискає кнопку пошуку. Після цього на сторінці нижче користувач побачить пошукові результати – послуги, назви яких містять ключовий запит, і місто співпадає з обраним містом для пошуку. |
| Extension | Якщо в системі не буде знайдено співпадінь з пошуковими запитами – користувач не побачить жодного пошукового результату |
| Post-Condition | На сторінці відображено пошукові результати за заданим запитом |

Таблиця 1.5 - Варіант використання UC-4

|  |  |
| --- | --- |
| Use case name | Перегляд сторінки послуги |
| Use case ID | UC-04 |
| Goals | Виведення сторінки послуги користувачеві |
| Actors | Замовник |
| Trigger | Користувач бажає дізнатися більше про послугу |
| Pre-conditions | Користувач авторизований, здійснив пошук на головній сторінці та бачить пошукові результати |
| Flow of Events | Користувач натискає на певний з пошукових результатів. Після цього користувача переносить на сторінку даної послуги. |
| Extension | - |
| Post-Condition | Користувач знаходиться на сторінці послуги |

Таблиця 1.6 - Варіант використання UC-5

|  |  |
| --- | --- |
| Use case name | Створення заявки на дану послугу |
| Use case ID | UC-05 |
| Goals | Додавання нової заявки до послуги |
| Actors | Замовник |
| Trigger | Користувач бажає лишити заявку на виконання даної послуги |
| Pre-conditions | Користувач знаходиться на сторінці послуги |
| Flow of Events | Користувач знаходиться на сторінці послуги. На ній користувач натискає кнопку створення заявки. Користувачу стає доступна форма для створення заявки. У формі користувач має увести опис задачі, обрати метод комунікації з доступних в меню радіобоксів, ввести свої контактні дані для обраного методу комунікації, та ввести бажану суму оплати. Після цього користувач натискає кнопку надсилання заявки. Після цього користувач знов знаходиться на сторінці послуги |
| Extension | - |
| Post-Condition | Створення заявки на послугу, перехід на сторінку послуги |

Таблиця 1.7 - Варіант використання UC-6

|  |  |
| --- | --- |
| Use case name | Перегляд власних заявок |
| Use case ID | UC-06 |
| Goals | Відображення користувачу надісланих ним заявок |
| Actors | Замовник |
| Trigger | Користувач бажає переглянути свої заявки |
| Pre-conditions | Користувач авторизований |
| Flow of Events | Користувач натискає на елемент для перегляду власних заявок та замовлень в хедері сайту. Після цього користувача перенаправлено на сторінку власних заявок та замовлень, на якій у відповідній частині відображатимуться надіслані даним користувачем заявки |
| Extension | Якщо користувач не надсилав заявок, користувач не побачить заявок на даній сторінці |
| Post-Condition | Користувач знаходиться на сторінці власних заявок та замовлень і бачить свої заявки. |

Таблиця 1.8 - Варіант використання UC-7

|  |  |
| --- | --- |
| Use case name | Видалення відхилених заявок |
| Use case ID | UC-07 |
| Goals | Видалення відхиленої заявки |
| Actors | Замовник |
| Trigger | Користувач бажає видалити відхилену заявку |
| Pre-conditions | Користувач авторизований, знаходиться на сторінці власних заявок та замовлень та має відхилену заявку |
| Flow of Events | Користувач натискає на кнопку видалення на компоненті заявки зі статусом “Declined”. Після цього користувач більше не бачитиме дану заявку на поточній сторінці, а заявка буде видалена з системи |
| Extension | - |
| Post-Condition | Відхилену заявку видалено з системи |

Таблиця 1.9 - Варіант використання UC-8

|  |  |
| --- | --- |
| Use case name | Перегляд власних замовлень |
| Use case ID | UC-08 |
| Goals | Відображення користувачу його власних замовлень |
| Actors | Замовник |
| Trigger | Користувач бажає переглянути власні замовлення |
| Pre-conditions | Користувач авторизований |
| Flow of Events | Користувач натискає на елемент для перегляду власних заявок та замовлень в хедері сайту. Після цього користувача перенавлено на сторінку власних заявок та замовлень, де у відповідній частині сторінки відображатимуться замовлення, в яких користувач є замовником. |
| Extension | Якщо жодна заявка користувача не була прийнята виконавцями, то не буде створено жодного замовлення і користувач не побачить жодного замовлення на відповідній сторінці |
| Post-Condition | Користувач знаходиться на сторінці власних заявок і замовлень і бачить свої замовлення |

Таблиця 1.10 - Варіант використання UC-9

|  |  |
| --- | --- |
| Use case name | Залишення відгуку на послугу |
| Use case ID | UC-09 |
| Goals | Додавання відгуку до послуги, що надавалась у замовленні |
| Actors | Замовник |
| Trigger | Користувач бажає лишити відгук про надану послугу |
| Pre-conditions | Користувач знаходиться на сторінці власних заявок та замовлень та має завершене замовлення |
| Flow of Events | Користувач натискає на кнопку для створення відгуку на компоненті замовлення, що має статус “Finished”. Після цього з’являється спливаюче вікно з формою створення відгуку. У дану форму користувач повинен увести свою оцінку послузі від 1 до 5, та, опціонально, ввести текст відгуку. Після цього користувач натискає на кнопку відправлення відгуку. Після цього спливаюче вікно зникає, і з компоненту замовлення зі статусом “Finished” зникає кнопка для створення відгуку |
| Extension | Користувач має можливість закрити спливаюче вікно без створення відгуку. |
| Post-Condition | Створено відгук щодо наданої в замовленні послуги, користувач знаходиться на сторінці власних заявок та замовлень |

Таблиця 1.11 - Варіант використання UC-10

|  |  |
| --- | --- |
| Use case name | Стати виконавцем на сайті |
| Use case ID | UC-10 |
| Goals | Додавання профілю виконавця для користувача |
| Actors | Замовник |
| Trigger | Користувач бажає стати виконавцем |
| Pre-conditions | Користувач авторизований |
| Flow of Events | Користувач натискає на елемент в хедері сайту, де відображається поточна роль користувача – “Customer”. Після цього відкривається спливаюче меню, в якому користувач має натиснути на елемент “Become a contractor”. Після цього користувача буде перенаправлено на сторінку створення профіля виконавця. На цій сторінці у формі користувач повинен ввести короткий опис про себе, та ввести від 1 до 4х одиниць контактних даних доступних видів. Після цього користувач натискає на кнопку для створення профілю виконавця. Після цього профіль буде створено, спливаюче меню надалі не міститиме елемент “Become a contractor”, а користувача перенаправлено на головну сторінку. |
| Extension | Користувач не може створити профіль виконавця, не ввівши жодний з елементів контактних даних |
| Post-Condition | Створення профіля виконавця, користувач став виконавцем, користувач знаходиться на головній сторінці |

Таблиця 1.12 - Варіант використання UC-11

|  |  |
| --- | --- |
| Use case name | Перехід в режим виконавця |
| Use case ID | UC-11 |
| Goals | Зміна хедеру сайту для відображення потрібних виконавцю елементів |
| Actors | Виконавець |
| Trigger | Користувач бажає перейти в режим виконавця |
| Pre-conditions | Користувач має режим замовника в хедері |
| Flow of Events | Користувач натискає на елемент у хедері, що відображає його поточну роль – “Customer”. Після цього відкривається спливаюче меню. В даному меню користувач повинен натиснути на елемент з назвою “Contractor”, після чого хедер зміниться, буде відображатись роль “Contractor” та в меню хедера з’являться елементи, пов’язані з потребами виконавця |
| Extension | - |
| Post-Condition | Перехід в режим виконавця, відображення відповідних елементів у хедері |

Таблиця 1.13 - Варіант використання UC-12

|  |  |
| --- | --- |
| Use case name | Перехід в режим замовника |
| Use case ID | UC-12 |
| Goals | Зміна хедеру сайту для відображення потрібних замовнику |
| Actors | Виконавець |
| Trigger | Користувач бажає перейти в режим замовника |
| Pre-conditions | Користувач має режим виконавця в хедері |
| Flow of Events | Користувач натискає на елемент у хедері, що відображає його поточну роль – “Contractor”. Після цього відкривається спливаюче меню. В даному меню користувач повинен натиснути на елемент з назвою “Customer”, після чого хедер зміниться, буде відображатись роль “Customer ” та в меню хедера з’являться елементи, пов’язані з потребами замовника |
| Extension | - |
| Post-Condition | Перехід в режим замовника, відображення відповідних елементів у хедері |

Таблиця 1.14 - Варіант використання UC-13

|  |  |
| --- | --- |
| Use case name | Створення послуги |
| Use case ID | UC-13 |
| Goals | Створення виконавцем послуги |
| Actors | Виконавець |
| Trigger | Користувач бажає створити послугу |
| Pre-conditions | Користувач має режим виконавця в хедері |
| Flow of Events | Користувач натискає на елемент у хедері для перегляду власних послуг. Після цього користувача перенаправлено на сторінку власних послуг. На даній сторінці користувач натискає на кнопку створення послуги. Після цього користувача перенаправлено на сторінку створення послуги. На даній сторінці у формі користувач повинен увести назву послуги, опис послуги, обрати в першому випадаючому меню область, далі в другому випадаючому меню – місто, потім ввести ціновий діапазон оплат власної послуги. Після цього користувач натискає на кнопку для створення послуги. Після цього послуга створюється, а користувача перенаправлено на сторінку власних послуг. |
| Extension | - |
| Post-Condition | Створення послуги користувача, користувач знаходиться на сторінці перегляду власних послуг. |

Таблиця 1.15 - Варіант використання UC-14

|  |  |
| --- | --- |
| Use case name | Перегляд власних послуг |
| Use case ID | UC-14 |
| Goals | Користувач бачить список власних послуг |
| Actors | Виконавець |
| Trigger | Користувач бажає переглянути власні послуги |
| Pre-conditions | Користувач має режим виконавця в хедері та користувач створив хоча б одну послугу |
| Flow of Events | Користувач натискає на елемент у хедері для перегляду власних послуг. Після цього користувача перенаправлено на сторінку власних послуг. На даній сторінці користувач бачитиме список власних послуг. |
| Extension | Якщо користувач не створив жодної послуги, список буде порожній |
| Post-Condition | Користувач знаходиться на сторінці перегляду послуг та бачить список власних послуг |

Таблиця 1.16 - Варіант використання UC-15

|  |  |
| --- | --- |
| Use case name | Перегляд заявок по послузі |
| Use case ID | UC-15 |
| Goals | Користувач бачить список заявок по обраній послузі |
| Actors | Виконавець |
| Trigger | Користувач бажає переглянути заявки по власній послузі |
| Pre-conditions | Користувач має режим виконавця в хедері та створив хоча б одну послугу |
| Flow of Events | Користувач натискає на елемент у хедері для перегляду власних задач на виконання. Після цього користувача перенаправлено на сторінку, на якій користувач бачитиме список власних послуг з кількістю заявок, що очікують відповіді, та кількістю замовлень, що не є завершеними. Користувач натискає на необхідну послугу зі списку. Після цього користувача перенаправляє на сторінку заявок та замовлень, де у відповідній частині сторінки будуть відображатись заявки, лишені на дану послугу |
| Extension | Якщо користувач не створив жодної послуги, список буде порожній |
| Post-Condition | Користувач знаходиться на сторінці перегляду заявок та замовлень по послузі та бачить список заявок, поданих на послугу |

Таблиця 1.17 - Варіант використання UC-16

|  |  |
| --- | --- |
| Use case name | Відхилення заявки на послугу |
| Use case ID | UC-16 |
| Goals | Користувач відхиляє обрану заявку |
| Actors | Виконавець |
| Trigger | Користувач бажає відхилити заявку на послугу |
| Pre-conditions | Користувач знаходиться на сторінці перегляду заявок та замовлень по послузі |
| Flow of Events | Користувач натискає на кнопку, що відповідає за відхилення заявки. Після цього заявка приховується зі списку заявок по послузі, а її статус оновлюється на “Declined”. |
| Extension | - |
| Post-Condition | Користувач знаходиться на сторінці перегляду заявок та замовлень по послузі та не бачить відхилену заявку |

Таблиця 1.18 - Варіант використання UC-17

|  |  |
| --- | --- |
| Use case name | Прийняття заявки на послугу |
| Use case ID | UC-17 |
| Goals | Користувач приймає обрану заявку |
| Actors | Виконавець |
| Trigger | Користувач бажає прийняти заявку на послугу |
| Pre-conditions | Користувач знаходиться на сторінці перегляду заявок та замовлень по послузі |
| Flow of Events | Користувач натискає на кнопку, що відповідає за прийняття заявки. Після цього на основі заявки створюється замовлення, а заявка приховується і видаляється. |
| Extension | - |
| Post-Condition | Користувач знаходиться на сторінці перегляду заявок та замовлень по послузі та не бачить прийняту заявку |

Таблиця 1.19 - Варіант використання UC-18

|  |  |
| --- | --- |
| Use case name | Перегляд замовлень по послузі |
| Use case ID | UC-18 |
| Goals | Користувач бачить список замовлень по обраній послузі |
| Actors | Виконавець |
| Trigger | Користувач бажає відхилити заявку на послугу |
| Pre-conditions | Користувач має режим виконавця в хедері та створив хоча б одну послугу |
| Flow of Events | Користувач натискає на елемент у хедері для перегляду власних задач на виконання. Після цього користувача перенаправлено на сторінку, на якій користувач бачитиме список власних послуг з кількістю заявок, що очікують відповіді, та кількістю замовлень, що не є завершеними. Користувач натискає на необхідну послугу зі списку. Після цього користувача перенаправляє на сторінку заявок та замовлень, де у відповідній частині сторінки будуть відображатись заявки, лишені на дану послугу. |
| Extension | - |
| Post-Condition | Користувач знаходиться на сторінці перегляду заявок та замовлень по послузі та бачить замовлення по обраній послузі |

Таблиця 1.20 - Варіант використання UC-19

|  |  |
| --- | --- |
| Use case name | Зміна статусу замовлення при початку роботи над ним |
| Use case ID | UC-19 |
| Goals | Користувач змінює статус замовлення з “Not started” в “In progress” |
| Actors | Виконавець |
| Trigger | Користувач бажає оновити статус замовлення при початку роботи над ним |
| Pre-conditions | Користувач знаходиться на сторінці перегляду заявок та замовлень по послузі та має хоч одне замовлення в статусі “Not started” |
| Flow of Events | Користувач натискає на кнопку на елементі замовлення зі статусом “Not started”, що відповідає за зміну статусу при початку роботи над замовленням. Після цього статус замовлення змінюється на “In progress” |
| Extension | - |
| Post-Condition | Користувач знаходиться на сторінці перегляду заявок та змінив статус замовлення з “Not started” в “In progress” |

Таблиця 1.21 - Варіант використання UC-20

|  |  |
| --- | --- |
| Use case name | Зміна статусу замовлення при завершенні роботи над ним |
| Use case ID | UC-20 |
| Goals | Користувач змінює статус замовлення з “In progress” в “Finished” |
| Actors | Виконавець |
| Trigger | Користувач бажає оновити статус замовлення при завершенні роботи над ним |
| Pre-conditions | Користувач знаходиться на сторінці перегляду заявок та замовлень по послузі та має хоч одне замовлення в статусі “In progress” |
| Flow of Events | Користувач натискає на кнопку на елементі замовлення зі статусом “In progress”, що відповідає за зміну статусу при завершенні роботи над замовленням. Після цього статус замовлення змінюється на “Finished” |
| Extension | - |
| Post-Condition | Користувач знаходиться на сторінці перегляду заявок та змінив статус замовлення з “In progress ” в “Finished” |

Таблиця 1.22 - Варіант використання UC-21

|  |  |
| --- | --- |
| Use case name | Вихід з сайту |
| Use case ID | UC-21 |
| Goals | Користувач виходить із системи |
| Actors | Авторизований користувач |
| Trigger | Користувач бажає вийти з сайту |
| Pre-conditions | Користувач авторизований та відкрив спливаюче меню користувача |
| Flow of Events | Користувач натискає на кнопку в спливаючому меню користувача, що відповідає за вихід з системи. |
| Extension | - |
| Post-Condition | Користувач більше не авторизований в системі |

### Розроблення функціональних вимог

В таблицях 1.23 – 1.40 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.4.

Таблиця 1.23 – Функціональна вимога FR-1

|  |  |
| --- | --- |
| Назва | Реєстрація користувача |
| Опис | Система повинна надавати можливість реєстрації користувачеві шляхом введення унікального імені користувача та паролю |

Таблиця 1.24 – Функціональна вимога FR-2

|  |  |
| --- | --- |
| Назва | Вхід |
| Опис | Система повинна надавати можливість входу користувачеві шляхом введення унікального імені користувача та паролю |

Таблиця 1.25 – Функціональна вимога FR-3

|  |  |
| --- | --- |
| Назва | Пошук послуг |
| Опис | Система повинна надавати можливість користувачеві здійснити пошук послуг за ключовим запитом та містом знаходження |

Таблиця 1.26 – Функціональна вимога FR-4

|  |  |
| --- | --- |
| Назва | Перегляд сторінки послуги |
| Опис | Система повинна надавати можливість переглядати сторінку послуги |

Таблиця 1.27 – Функціональна вимога FR-5

|  |  |
| --- | --- |
| Назва | Залишення заявки на послугу |
| Опис | Система повинна надавати можливість залишити заявку на надання послуги |

Таблиця 1.28 – Функціональна вимога FR-6

|  |  |
| --- | --- |
| Назва | Перехід на головну сторінку |
| Опис | Система повинна надавати можливість перейти на головну сторінку шляхом натискання відповідного елементу в лівій частині хедера |

Таблиця 1.29 – Функціональна вимога FR-7

|  |  |
| --- | --- |
| Назва | Відкриття спливаючого меню користувача |
| Опис | Система повинна надавати можливість відкриття спливаючого меню користувача шляхом натискання відповідного елементу в правій частині хедера |

Таблиця 1.30 – Функціональна вимога FR-8

|  |  |
| --- | --- |
| Назва | Можливість стати виконавцем |
| Опис | Система повинна надавати можливість користувачеві стати виконавцем |

Таблиця 1.31 – Функціональна вимога FR-9

|  |  |
| --- | --- |
| Назва | Переключення між режимами замовника та виконавця |
| Опис | Система повинна надавати можливість перемикатись між режимами замовника та виконавця |

Таблиця 1.32 – Функціональна вимога FR-10

|  |  |
| --- | --- |
| Назва | Перегляд власних заявок та замовлень |
| Опис | Система повинна надавати можливість користувачеві як замовнику переглядати власні заявки та замовлення |

Таблиця 1.33– Функціональна вимога FR-11

|  |  |
| --- | --- |
| Назва | Видалення власних відхилених заявок |
| Опис | Система повинна надавати можливість користувачеві як замовнику видаляти власну відхилену заявку |

Таблиця 1.34 – Функціональна вимога FR-12

|  |  |
| --- | --- |
| Назва | Можливість залишити огляд на надану послугу |
| Опис | Система повинна надавати можливість користувачеві як замовнику залишити огляд на надану послугу |

Таблиця 1.35 – Функціональна вимога FR-13

|  |  |
| --- | --- |
| Назва | Створення послуг |
| Опис | Система повинна надавати можливість користувачеві як виконавцю створити послугу |

Таблиця 1.36 – Функціональна вимога FR-14

|  |  |
| --- | --- |
| Назва | Перегляд власних послуг |
| Опис | Система повинна надавати можливість користувачеві як виконавцю переглядати власні послуги |

Таблиця 1.37 – Функціональна вимога FR-15

|  |  |
| --- | --- |
| Назва | Перегляд сторінки заявок та замовлень по послузі |
| Опис | Система повинна надавати можливість користувачеві як виконавцю переглядати сторінку заявок та замовлень по послузі |

Таблиця 1.38 – Функціональна вимога FR-16

|  |  |
| --- | --- |
| Назва | Можливість відхиляти та приймати заявки по послузі |
| Опис | Система повинна надавати можливість користувачеві як виконавцю відхиляти та приймати заявки по послузі |

Таблиця 1.39 – Функціональна вимога FR-17

|  |  |
| --- | --- |
| Назва | Зміна статусу замовлення |
| Опис | Система повинна надавати можливість користувачеві як виконавцю змінювати статус замовлення |

Таблиця 1.40 – Функціональна вимога FR-18

|  |  |
| --- | --- |
| Назва | Перегляд сторінки заявок та замовлень по послузі |
| Опис | Система повинна надавати можливість користувачеві як виконавцю переглядати сторінку заявок та замовлень по послузі |

A grid of lines with different colored dots

Description automatically generated

Рисунок 1.4 – Матриця трасування вимог

### Розроблення нефункціональних вимог

* Додаток повинен мати зрозумілий у використанні інтерфейс;
* Додаток повинен коректно працювати на браузерах, що підтримують роботу JavaScript;
* Додаток повинен працювати на платформі Windows 10 або вище;
* Додаток повинен забезпечувати цілісність інформації в базі даних;
* Додаток повинен забезпечувати конфіденційність контактних даних користувачів.

## Постановка задачі

У рамках роботи планується розробити веб-застосунок, який матиме на меті реалізацію конфіденційного та простого використання кінцевим користувачем.

Першою задачею постає анонімна реєстрація. Реєстрація на сайті не вимагатиме персональних даних, а лиш вибір унікального імені користувача в системі, за бажання вибору імені, яке відображатиметься на сайті, для кращого представлення себе іншим користувачам, та паролю.

Механізм знаходження послуги, яку користувач хотів би отримати, складатиметься з пошуку по ключовому запиту та місту знаходження, яке цікавить користувача. Отримавши пошукові результати, користувач може переглянути сторінку послуги, яка його зацікавила, дізнатися більше про виконавця, побачити, які методи комунікації він використовує з замовниками, та переглянути відгуки.

Користувач зможе лишити заявку на послугу, яку він для себе обрав, увівши опис задачі та очікуваний бюджет оплати, та обравши один з методів комунікації, що доступні у виконавця, і ввести свої контактні дані відповідного типу. Після надсилання заявки замовник бачитиме контактні дані виконавця обраного типу, переглядаючи дану заявку. Аналогічно виконавець при перегляді заявки, що надійшла, бачитиме контактні дані замовника. Таким чином досягається конфіденційність, мінімальне поширення контактних даних як замовника, так і виконавця, і користувачі мають свободу вибору комунікації на платформах, що вважають підходящими та надійними.

Виконавець матиме простий механізм управління заявками та замовленнями, що дозволятиме керувати задачами і організовувати робочий процес.

Користувачі зможуть лишати відгуки на надану послугу після факту виконання даного замовлення. Ці відгуки допоможуть іншим користувачам враховувати досвід надання послуг попередніх замовників, та покращуватиме процес вибору оптимальної послуги.

## Висновки до розділу

У цьому розділі було розглянуто предметну область надання послуг: звідки взялись, як сформувалась концепція, які блага та які виклики сформували ситуацію, яку маємо на сьогодні. Визначили, що ця сфера в IT і досі є актуальною.

Звернули увагу на тенденції в сфері пошуку та надання послуг. Аналіз предметної області в загальному та аналіз існуючих IT-проектів, визначення і оцінка алгоритмічних і технічних рішень допомогли сформувати чітку картину стану речей, оцінити досвід використання, з яким стикаються люди щодня.

Проведений в розділі порівняльний аналіз допоміг визначити як переваги, так і недоліки аналогів в сферах функціональності, конфіденційності та зручного досвіду використання. Це зміцнило уявлення та переконання важливості урахування аспектів, визначених раніше.

Було побудувано і детально досліджено діаграми використання, визначено функціональні вимоги та співставлено їх з варіантами використання у матриці трасування. Це все допомогло у розумінні необхідності функціоналу в проекті, яку роль грає кожна функціональна вимога у картині загального проекту.

Визначено інструменти розробки, бібліотеки, фреймворки, програмні пакети, СУБД, та IDE, які разом становлять екосистему поєднаних інструментів для вирішення задачі розробки веб-застосунку системи пошуку та надання послуг.  
,

# МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу програмного забезпечення використовуються BPMN моделі (рисунки 2.1-2.2).

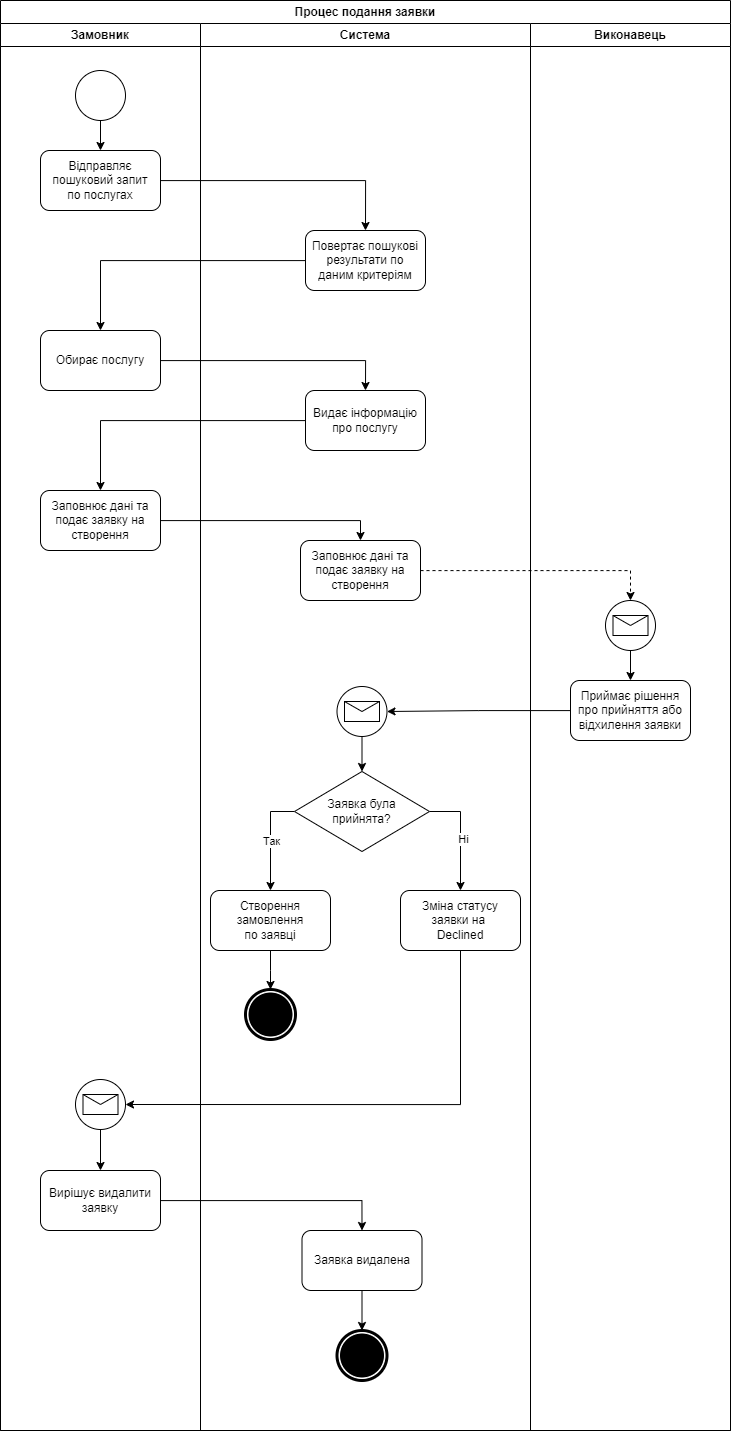
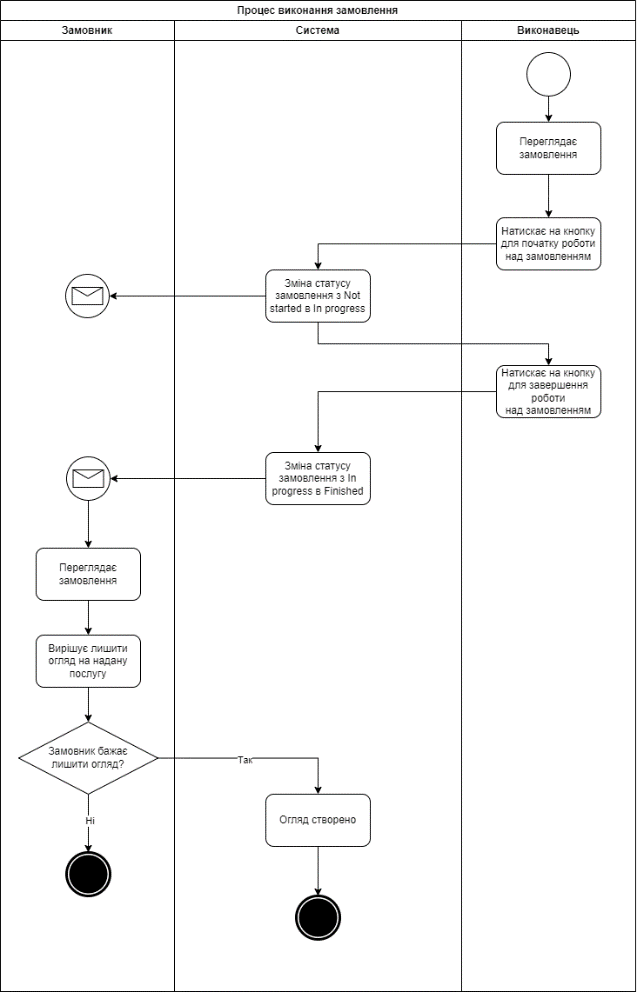


Рисунок 2.1 – BPMN-модель процесу подання заявки

Опис послідовності створення заявки на виконання послуги:

* замовник вводить критерії для пошуку послуг;
* замовник здійснює пошук;
* замовник обирає послугу з пошукових результатів;
* замовник переходить на сторінку послуги;
* замовник подає заявку на послугу;
* виконавець переглядає заявки по послузі;
* якщо виконавець відхиляє заявку, то статус заявки змінюється на Declined;
* замовник переглядає заявки;
* замовник видаляє заявку зі статусом Declined, чим закінчує процес;
* якщо виконавець прийняв заявку, то по даній заявці буде створено замовлення, що закінчує процес.



* Рисунок 2.2 – BPMN-модель процесу виконання замовлення

Опис послідовності процесу виконання замовлення:

* виконавець переглядає замовлення;
* виконавець натискає кнопку для початку роботи над замовленням, після чого система змінює статус замовлення з Not started в In progress;
* виконавець натискає кнопку для завершення роботи над замовленням, після чого система змінює статус замовлення з In progress в Finished;
* замовник переглядає замовлення в статусі Finished;
* якщо замовник бажає створити замовлення, він вводить дані і відправляє їх на створення, після чого система створює замовлення, чим закінчує процес;
* якщо замовник не бажає створити замовлення, процес закінчено.

## Архітектура програмного забезпечення

Для розробки веб-застосунку системи пошуку послуг було використано клієнт-серверну та монолітну архітектури.

Клієнт-серверна архітектура – це відношення, в якому один компонент, клієнт, запрошує послугу чи ресурс від іншого компоненту, сервера [13]. Це дозволяє розділити функціональні обов’язки між частинами системи. Клієнтська частина реалізує взаємодію з користувачем, отримання введеної інформації та відображення результатівю. Сервер відповідає за обробку запитів від клієнтської частини, виконання бізнес-правил, збереження та маніпуляцією даних в БД. Таке архітектурне рішення надає чітке розділення відповідальностей у системі, що полегшує підтримку та майбутнє розширення системи.

Монолітна архітектура – підхід, в якому весь споріднений функціонал системи виконується одним програмним додатком. Монолітна архітектура є оптимальним варіантом для швидкої розробки додатку, та має перевагу швидкої роботи, тому що всі запити робляться до єдиної бази коду, без навантаження на передачу повідомлень між процесами користувацького рівня. [14]. Проте, при великому зрості попиту на використання веб-застосунку прийдеться розглядати варіант мікросервісної архітектури.

Дані архітектурні патерни задовольняють потреби розробки, забезпечуючи розділення функціональних відповідальностей системи, швидкість розробки та простір для розширення функціоналу в подальшому розвитку проекту.

Для візуалізації архітектури використано діаграму C4 в трьох рівнях деталізації.

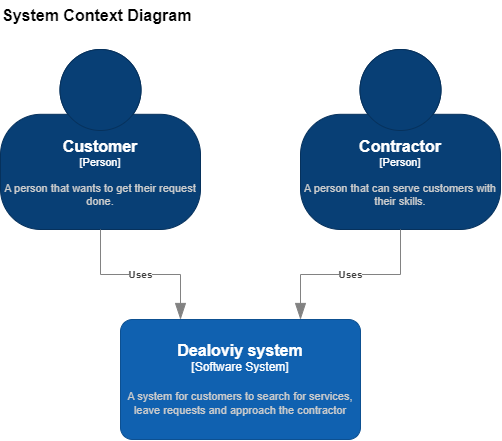


Рисунок 2.3 – Діаграма системного контексту

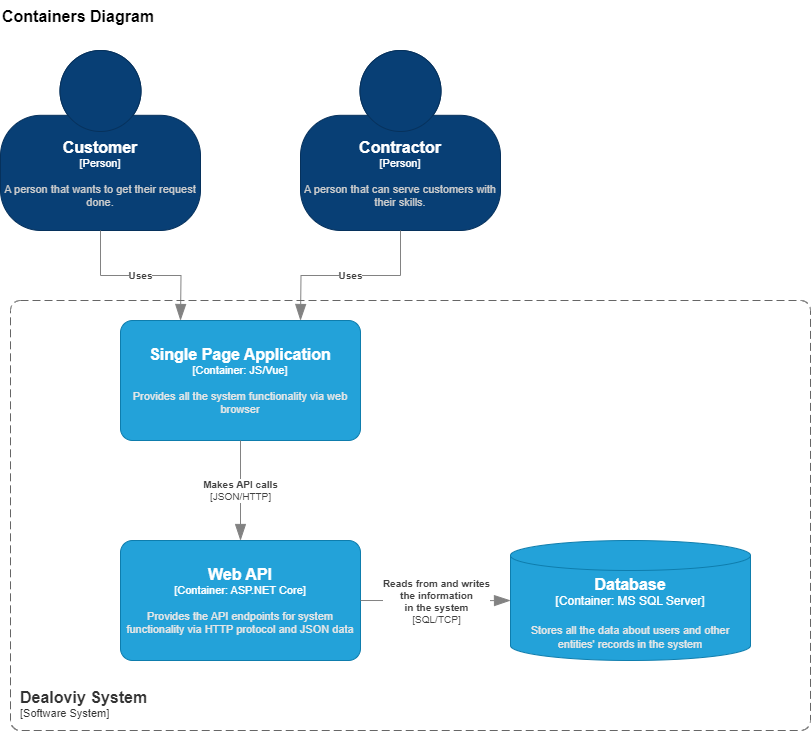


Рисунок 2.4 – Діаграма контейнерів

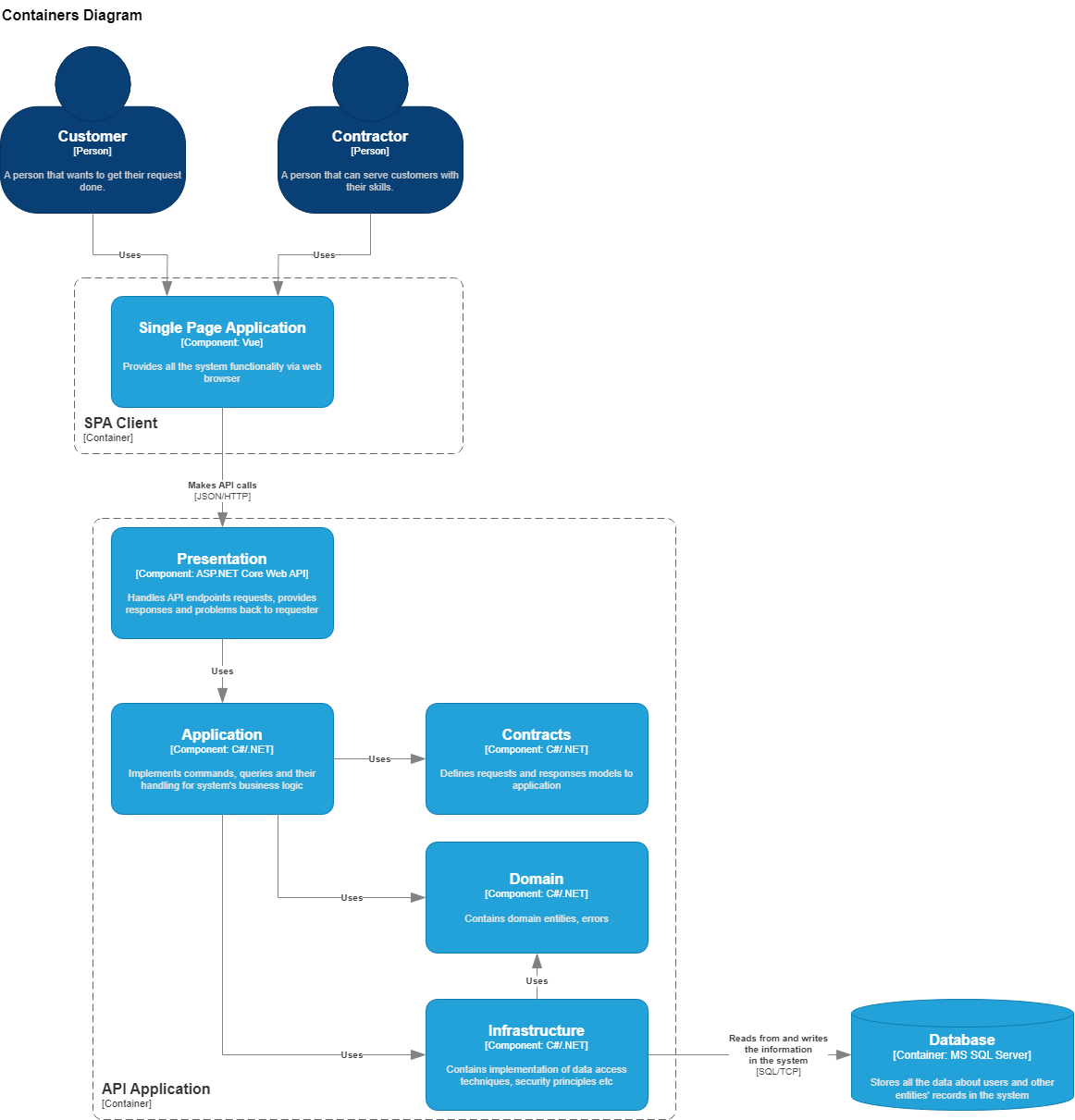


Рисунок 2.5 – Діаграма компонент

## Конструювання програмного забезпечення

В наступному підрозділі будуть описані алгоритми, структури даних, програмні структури, бази даних, програмні утиліти, що необхідні для конструювання програмного забезпечення курсової роботи.

### Алгоритми

Алгоритм використання веб-застосунку наступний. Користувач здійнює пошук послуг шляхом введення ключового запиту, вибору географічної області, та вибору міст в межах обраної області. При пошуку послуг за даними критеріями здійснюється запит до записів послуг, та фільтрація по наявності ключового запиту в назві послуги та співпадіння ідентифікатору міста в послузі та в запиті. Після отримання даних про послугу, також надсилається запит на записи міст. З цих двох наборів інформації формуємо записи пошукових результатів послуг, поєднуючи власні дані послуги з назвою міста для відображення. Тоді сформовані записи сортуються по середньому значенню рейтингу послуг, та повертаються у відповіді разом з кількістю знайдених пошукових результатів, ключовим словом та назвою міста, по яким здійснявся пошук.

Отримані пошукові результати користувач може використати, аби перейти на сторінку послуги, що його зацікавила. На сторінці послуги користувач, окрім основної інформації, бачитиме інформацію про виконавця: ім’я на сайті, опис про себе, та які методи комунікації надає замовникам. До того ж, на сторінці відображатиметься список відгуків на послугу від інших користувачів в порядку актуальності дати. На даній сторінці завдяки кнопці користувач може перейти до створення заявки на надання послуги, де повинен ввести необхідну інформацію для замовника: опис проблеми, очікуваний бюджет та контактні дані, для яких користувач сам обирає один з доступних методів комунікації.

В подальшому користувач-замовник може спостерігати за станом власних заявок, а користувач-виконавець матиме механізм відхилення або прийняття заявок. Якщо виконавець відхилить заявку – він більше її не бачитиме, а замовник помітить, що заявка в відхиленому стані, та матиме можливість видалити її. Виконавець бачитиме заявки, що надходять на послугу, в порядку актуальності дати подання заявки. В випадку прийняття виконавцем заявки – вона зникатиме для обох користувачів, а на її основі буде сформовано нове замовлення. В заявках і замовленнях виконавець та користувач бачитимуть контактні дані одне одного для зв’язку. Виконавець матиме функціонал для послідовного переведення статусу замовлення по станам “Not started”, “In progress”, “Finished”. Замовник матиме можливість переглядати власні замовлення та бачити, в якому статусі вони знаходяться. При наявності замовлення в завершеному стані замовник має можливість лишити відгук про надану йому послугу, оцінити її по п’ятибальній шкалі та надати текстовий коментар.

Для того, щоб стати виконавцем на сайті користувачу не потрібно створювати окремий акаунт під цю роль, йому треба буде додати короткий опис себе як виконавця на сторінці створення профіля виконавця, та внести контактні дані, якими зможуть користуватись замовники для комунікації. При наявності профіля виконавця користувач може перемикатись між режимами виконавця та замовника, аби мати можливість використовувати сайт в різних цілях.

Дані алгоритми забезпечують просте та зрозуміле використання користувачами, як замовниками, так і виконавцями, і надають зручні механізми для надання послуг та виконання замовлень.

### Структури даних, програмні структури

В процесі конструювання програмного забезпечення дані були структуровані в класичні структури даних: списки, колекції, об’єкти.

Наступні діаграми класів покращать розуміння програмної структури.

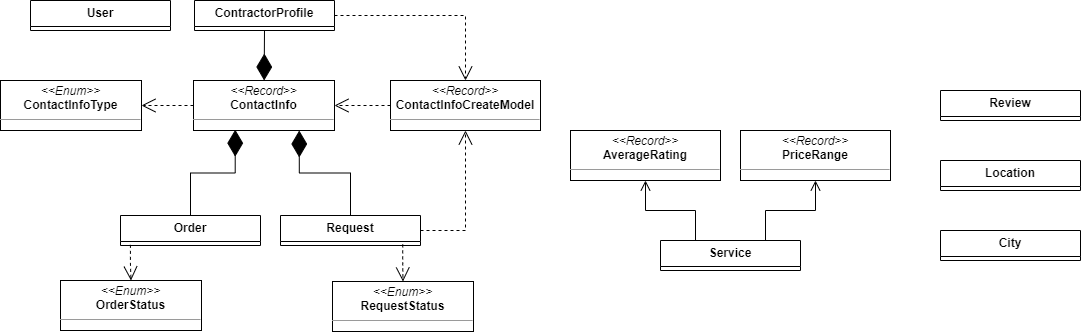


Рисунок 2.6 – Діаграма класів шару Domain

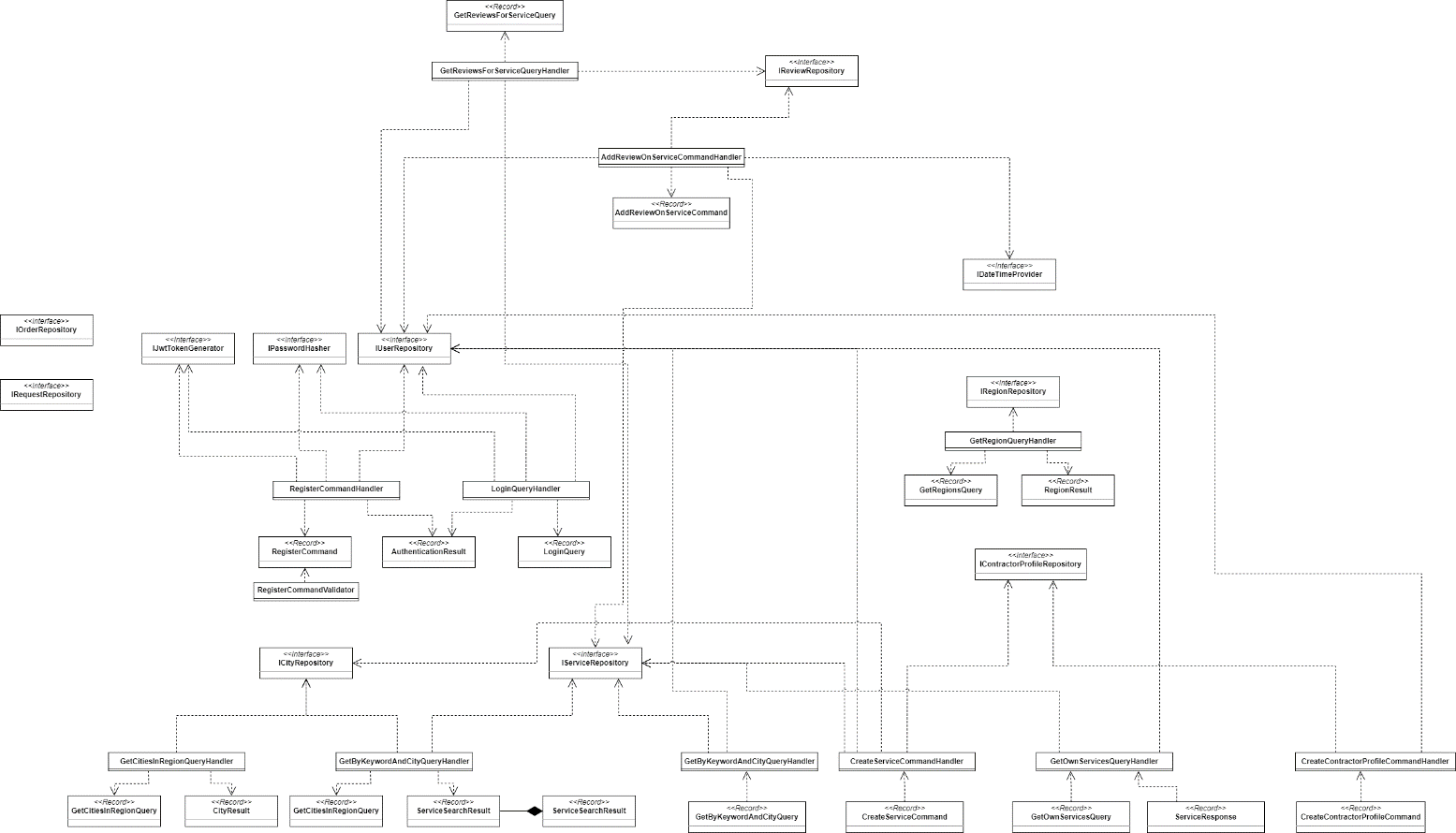


Рисунок 2.7 – Діаграма класів шару Application

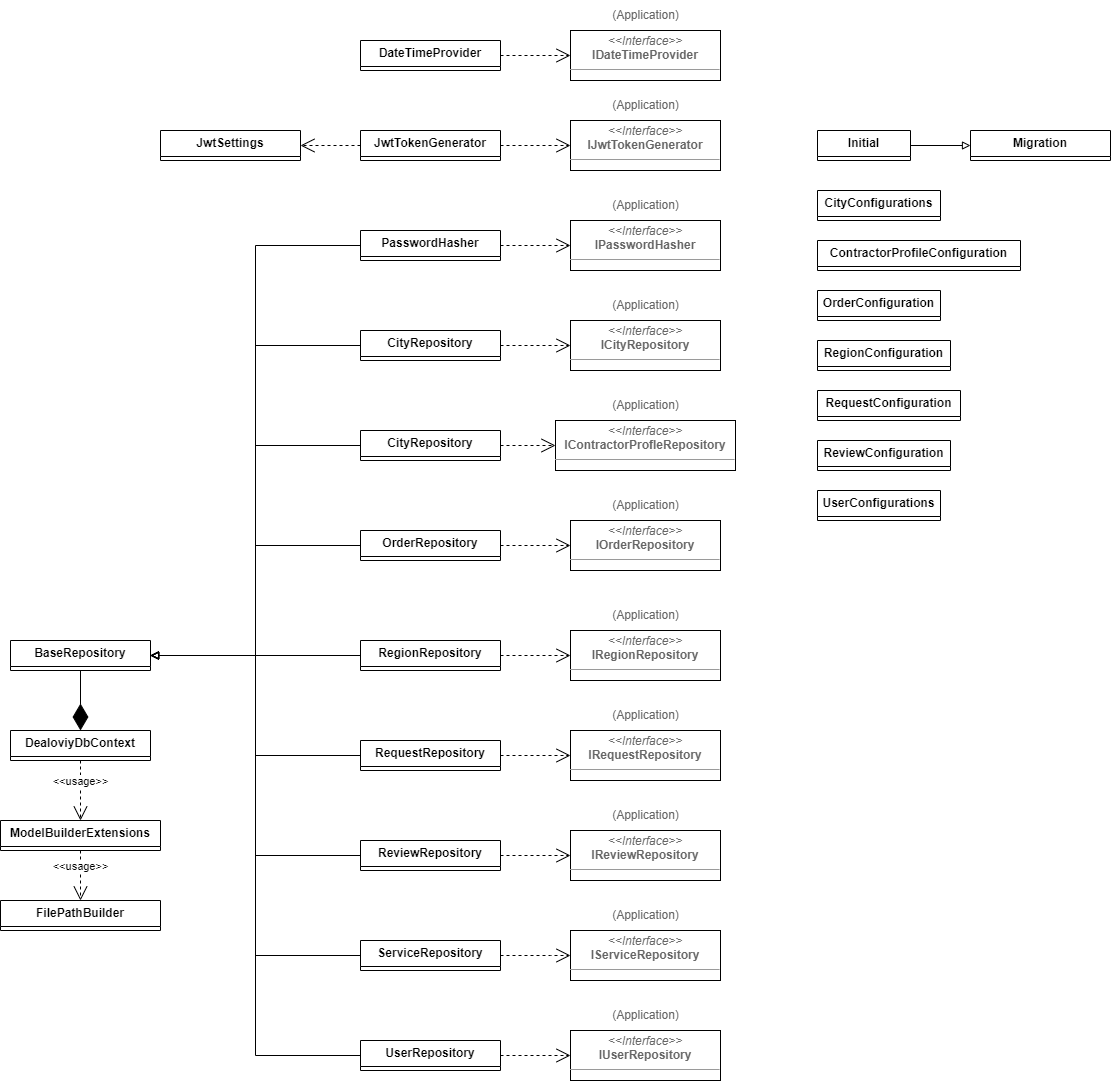


Рисунок 2.8 – Діаграма класів шару Infrastructure

### Структура бази даних

В якості системи управління базами даних використовується Microsoft SQL Server. База даних серверу призначена для зберігання користувачів, а також даних про їх профілі виконавців, полуги, заявки та замовлення. Також зберігаються статичні дані про список областей та міст. ER-діаграма бази даних зображена на рисунку 2.9. Опис таблиць бази даних наведено у таблицях 2.11 - 2.14.

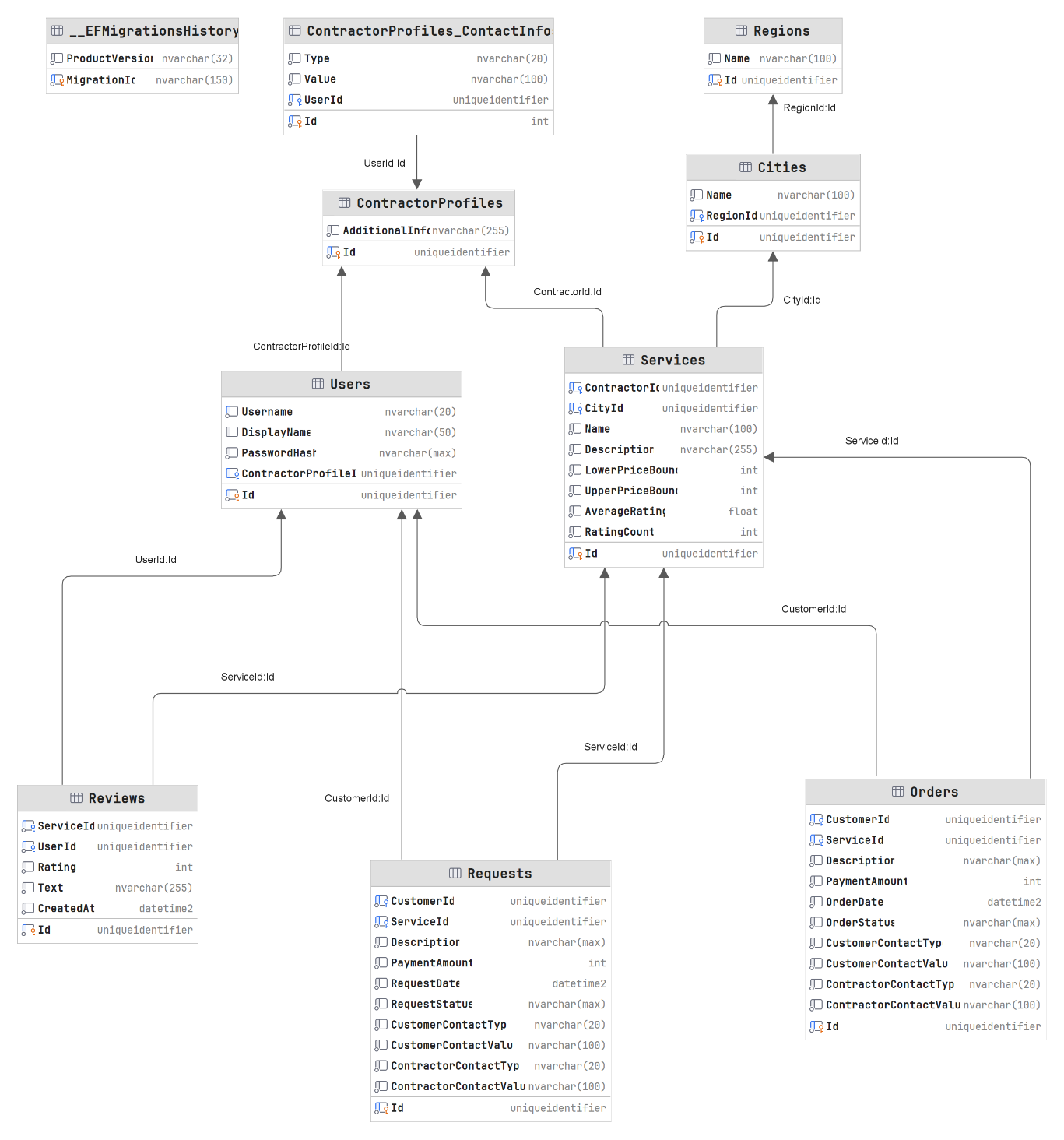


Рисунок 2.9 – ER-діаграма бази даних

Таблиця 2.1 – Опис таблиці Users

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Users | Id | uniqueidentifier | унікальний ідентифікатор користувача |
| Username | nvarchar | унікальне ім’я користувача в системі |
| DisplayName | nvarchar | ім’я, яке користувач може використовувати для відображення на сайті замість унікального юзернейму |
| PasswordHash | nvarchar | хешований пароль користувача |
| ContractorProfileId | uniqueidentifier | FK на таблицю сутності профіля виконавця |

Таблиця 2.2 – Опис таблиці ContractorProfiles

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| ContractorProfiles | Id | uniqueidentifier | унікальний ідентифікатор профіля виконавця |
| AdditionalInfo | nvarchar | інформація про виконавця |

Таблиця 2.3 – Опис таблиці ContractorProfiles\_ContactInfo

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| ContractorProfiles\_ContactInfo | Id | uniqueidentifier | унікальний ідентифікатор контактних даних |
| Type | nvarchar | метод комунікації |
| Value | nvarchar | контактна інформація для даного методу комунікації |
| UserId | uniqueidentifier | FK на таблицю сутності профіля виконавця |

Таблиця 2.4 – Опис таблиці Services

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Services | Id | uniqueidentifier | унікальний ідентифікатор послуги |
| ContractorId | uniqueidentifier | FK на таблицю сутності профіля виконавця |
| CityId | uniqueidentifier | FK на таблицю сутності міста |
| Name | nvarchar | назва послуги |
| Description | nvarchar | опис послуги |
| LowerPriceBound | int | нижня межа цінового діапазону |
| UpperPriceBound | int | верхня межа цінового діапазону |
| AverageRating | float | середнє значення оцінок послуги |
| RatingCount | int | кількість оцінок послуги |

Таблиця 2.5 – Опис таблиці Cities

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Cities | Id | uniqueidentifier | унікальний ідентифікатор міста |
| RegionId | uniqueidentifier | FK на таблицю сутності області |
| Name | nvarchar | географічна назва міста |

Таблиця 2.6 – Опис таблиці Regions

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Regions | Id | uniqueidentifier | унікальний ідентифікатор користувача |
| Name | nvarchar | географічна назва області |

Таблиця 2.7 – Опис таблиці Requests

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Requests | Id | uniqueidentifier | унікальний ідентифікатор заявки |
| CustomerId | uniqueidentifier | FK на таблицю сутності користувача; ідентифікатор користувача, що лишив заявку |
| ServiceId | uniqueidentifier | FK на таблицю сутності послуги; ідентифікатор послуги, на яку створено заявку |
| Description | nvarchar | опис задачі, яку замовник бажає виконати наданням даної послуги |
| PaymentAmount | int | бюджет замовника на дану заявку |
| RequestDate | datetime2 | дата подачі заявки |
| RequestStatus | nvarchar | статус заявки |
| CustomerContactType | nvarchar | метод комунікації замовника |
| CustomerContactValue | nvarchar | контактні дані замовника |
| ContractorContactType | nvarchar | метод комунікації виконавця |
| ContractorContactValue | nvarchar | контактні дані виконавця |

Таблиця 2.8 – Опис таблиці Orders

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Orders | Id | uniqueidentifier | унікальний ідентифікатор замовлення |
| CustomerId | uniqueidentifier | FK на таблицю сутності користувача; ідентифікатор користувача, чиє замовлення виконується |
| ServiceId | uniqueidentifier | FK на таблицю сутності послуги; ідентифікатор послуги, по якій виконується замовлення |
| Description | nvarchar | опис задачі, яку замовник бажає виконати наданням даної послуги |
| PaymentAmount | int | бюджет замовника на дане замовлення |
| OrderDate | datetime2 | дата створення замовлення |
| OrderStatus | nvarchar | статус замовлення |
| CustomerContactType | nvarchar | метод комунікації замовника |
| CustomerContactValue | nvarchar | контактні дані замовника |
| ContractorContactType | nvarchar | метод комунікації виконавця |
| ContractorContactValue | nvarchar | контактні дані виконавця |

Таблиця 2.9 – Опис таблиці Reviews

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Reviews | Id | uniqueidentifier | унікальний ідентифікатор огляду |
| ServiceId | uniqueidentifier | FK на таблицю сутності послуги; ідентифікатор послуги, на яку залишено огляд |
| UserId | uniqueidentifier | FK на таблицю сутності користувача; ідентифікатор користувача, що залишив огляд на послугу |
| Rating | int | оцінка наданої послуги |
| Text | nvarchar | текстовий опис огляду на надану послугу |
| CreatedAt | datetime2 | Дата додавання огляду |

### Використані бібліотеки та утиліти

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.10.

Таблиця 2.10 – Опис утиліт

|  |  |  |
| --- | --- | --- |
| №  п/п | Назва утиліти | Опис застосування |
| 1 | JetBrains Rider | Головне середовище розробки програмного забезпечення серверної частини курсової роботи. |
| 2 | Visual Studio Code | Текстовий редактор, що завдяки розширенням стає швидким середовищем розробки клієнтської частини курсової роботи. |
| 3 | Postman | Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів. |
| 4 | CsvHelper | Програмний пакет з платформи NuGet, що допомагає в читанні та роботі з даними файлів формату .csv |

### Аналіз системних вимог

Програмне забезпечення повинно бути сумісним з ОС Windows 10 та вище.

Для роботи веб-застосунку необхідно мати встановлені .NET 6 для роботи бекенду та Vue CLI для роботи фронтенду.

Веб-застосунок повинен підтримувати актуальні браузери, що підтримують JavaScript, такі як Google Chrome, Opera, Microsoft Edge, тощо

Для роботи застосунку необхідно мати стабільне Інтернет-з’єднання.

## Аналіз безпеки даних

Задля забезпечення конфіденційності цілісності і захищеності даних веб-застосунку проведемо аналіз безпеки даних, де визначимо можливі вразливості і методи їх запобігання.

Аутентифікація та авторизація: задля запобігання несанкціонованого доступу реалізовано механізми автентифікації та авторизації, що працюють на порівнянні креденціалів та JWT-токенах.

Перевірка даних на коректність: задля цілісності даних в системі необхідно валідувати дані за визначеними правилами.

Налаштування CORS: CORS на серверній частині налаштований таким чином, щоб дозволяти робити запити лиш фронтенд-додатку.

## Висновки до розділу

У даному розділі було проведено багато важливих етапів з моделювання та конструювання програмного забезпечення веб-застосунку системи пошуку послуг.

Першим етапом у розділі стало моделювання бізнес-процесів. Було використано BMPN-моделі для моделювання основних бізнес-процесів в системі: подання заявки та виконання замовлення.

Наступним кроком став аналіз та визначення архітектурних підходів. Для розробки було обрано два архітектурні підходи: клієнт-серверний, що допомагає розділяти обов’язки системи, такі як взаємодія з користувачем, відправка запитів, виведення інформації – відповідальність клієнтської частини; монолітний – реалізація основного функціоналу системи в одній базі коду, що є швидким у розробці та у роботі. Для детальнішого аналізу і визначення архітектури системи було спроектовано діаграму C4 трьох рівнів деталізації. На рівні контексту системи ми зобразили, як наша монолітна система розміщується у світі та як взаємодіє з користувачами. На рівні контейнерів ми визначили, що в нашій системі за взаємодію з користувачем відповідає клієнтський фронтенд-застосунок, а обробка запитів, реалізація бізнес-логіки та взаємодія з базою даних виконуються серверною частиною. На рівні компонентів визначили, що серверна частина складається з різних шарів, де кожен має свою роль.

Було визначено й описано алгоритми, визначено структури даних, програмна структура була зображена в діаграмах класів.

Структура бази даних була зображена на ER-діаграмі та описана у таблицях сутностей, що покращує уявлення про те, в якому виді зберігаються дані, та як пов’язані між собою.

Також було визначено бібліотеки, програмні утиліти і інструменти, які будуть потрібні у розробці програмного забезпечення. Було проведено аналіз системних вимог веб-застосунку.

Аналіз безпеки даних виявив потенційні вразливості та методи їх запобігання у наступних напрямках: автентифікація та авторизація, перевірка даних на коректність, налаштування CORS. Ці заходи допоможуть забезпечити конфіденційність, цілісність та захищеність інформації, що знаходиться в системі.

Багатокроковий процес моделювання та конструювання програмного забезпечення допоміг в візуалізації різних аспектів системи, що допоможе в подальшій розробці веб-застосунку, зробивши його якісним, ефективним та надійним.

# АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Аналіз якості ПЗ

У розробці програмного забезпечення, як і у будь-якій іншій інженерній сфері, недостатньо рішення, яке просто виконує поставлену задачу, потрібно слідкувати і за якістю написаного коду – це забезпечує зручність продовження розробки в тій же базі коду, власний код стає легшим для розуміння іншими учасниками команди, допомагає виявляти та усувати помилки на етапах розробки. Для виявлення можливих помилок у коді скористаємось драйвером dotnet в інструментах .NET CLI, а саме команду dotnet build:

A close-up of a computer screen

Description automatically generated

Рисунок 3.1 – результат виконання команди dotnet build

Як бачимо, проект збирається, не має помилок або попереджень щодо процесу збірки.

Додатково скористаємось вбудованим в Rider аналізатором проблем:

A screenshot of a computer

Description automatically generated

Рисунок 3.2 – результат аналізатору коду Rider

Бачимо, що не виникає жодної помилки, а попередження здебільшого не носять критичний характе

р, та більшість з цих попереджень викликані кодом, що був розроблений з наміром роботи саме таким чином (застосування pattern matching техніки для приведення результату, що може містити null, у змінну певного типу, задля подальшого використання), або ж для покращення розуміння коду (явне вираження конструктору, який компілятор теж генерує).

## Опис процесів тестування

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.30.

Таблиця 3.1 – Тест 1.1

|  |  |
| --- | --- |
| Тест | Реєстрація користувача |
| Номер тесту | 1.1 |
| Початковий стан системи | Користувач знаходиться на сторінці реєстрації |
| Вхідні дані | Юзернейм, ім’я для відображення на сайті, пароль |
| Опис проведення тесту | У відповідні поля вводяться: унікальний юзернейм, який до цього не був зареєстрований в системі, ім’я для відображення (опціональне поле), пароль від 8 символів. Після цього натискається кнопка Sign Up. |
| Очікуваний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |
| Фактичний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |

Таблиця 3.2 – Тест 1.2

|  |  |
| --- | --- |
| Тест | Вхід |
| Номер тесту | 1.2 |
| Початковий стан системи | Користувач знаходиться на сторінці входу, є зареєстрованим на сайті |
| Вхідні дані | Юзернейм, пароль |
| Опис проведення тесту | У відповідні поля вводяться: коректний юзернейм, який був зареєстрований в системі користувачем, відповідний пароль від 8 символів. Після цього користувач натискає кнопку Sign In |
| Очікуваний результат | Авторизація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |
| Фактичний результат | Авторизація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |

Таблиця 3.3 – Тест 1.3

|  |  |
| --- | --- |
| Тест | Пошук послуг |
| Номер тесту | 1.1 |
| Початковий стан системи | Користувач авторизований, знаходиться на головній сторінці |
| Вхідні дані | Ключовий запит: “Photography”, область: “Kyiv, Misto”, місто: “Kyiv” |
| Опис проведення тесту | У поле текстового вводу користувач вводить ключовий запит “Photography”, потім обирає в випадаючому меню областей “Kyiv, Misto”, після цього в меню міст обирає “Kyiv” , після цього натискає кнопку Search. |
| Очікуваний результат | Запит оброблюється, на сторінці користувач бачитиме результати пошуку по ключовому запиту “Photography” в місті “Kyiv”. |
| Фактичний результат | Запит оброблюється, на сторінці користувач бачитиме результати пошуку по ключовому запиту “Photography” в місті “Kyiv”. |

Таблиця 3.4 – Тест 1.4

|  |  |
| --- | --- |
| Тест | Подання заявки |
| Номер тесту | 1.4 |
| Початковий стан системи | Користувач знаходиться на сторінці послуги, відкрив форму створення заявки. |
| Вхідні дані | Опис задачі: “I need to do a professional photoshoot on 05.02 16:00”, обраний тип контактних даних: “Telegram”, введені контактні дані: “@JohnDoe” |
| Опис проведення тесту | У відповідні поля форми вводяться: опис задачі, що є непорожнім полем – користувач вводить “I need to do a professional photoshoot on 05.02 16:00”, обирається тип контактних даних з меню радіобоксів – користувач обирає тип “Telegram”. Вводяться дані валідного виду для кожного типу: для “Phone” – мають вводитись лише цифри, номер повинен мати довжину 12 цифр, та починатись з 380, для інших типів-месенджерів – повинні починатись з символу “@” та мати довжину не більше 100 символів. Користувач вводить у поле контактних даних “@JohnDoe”. Введення даних в очікуваний бюджет оплати повинно валідуватись у ціле число додатного виду. Після введення даних користувач натискає кнопку Send. |
| Очікуваний результат | Створення заявки проходить успішно – створюється заявка з описом “I need to do a professional photoshoot on 05.02 16:00”, типом контактних даних “Telegram” з контактними даними “@JohnDoe”. |
| Фактичний результат | Створення заявки проходить успішно – створюється заявка з описом “I need to do a professional photoshoot on 05.02 16:00”, типом контактних даних “Telegram” з контактними даними “@JohnDoe”. |

Таблиця 3.5 – Тест 1.5

|  |  |
| --- | --- |
| Тест | Створення профіля виконавця |
| Номер тесту | 1.5 |
| Початковий стан системи | Користувач авторизований, знаходиться на сторінці створення профіля виконавця |
| Вхідні дані | Текстовий опис себе як виконавця: “Professional photographer in Kyiv”, контактні дані: тип “Phone”: “380951234567”, тип “Telegram”: “@PhotographerInKyiv”. |
| Опис проведення тесту | У відповідні поля вводяться: опис виконавця – має бути непорожнім полем та довжиною до 255 символів. Користувач вводить в поле опису виконавця “Professional photographer in Kyiv”. Для валідації користувач повинен ввести контактні дані хоча б одного виду. Вводяться дані валідного виду для кожного типу: для “Phone” – мають вводитись лише цифри, номер повинен мати довжину 12 цифр, та починатись з 380, для інших типів-месенджерів – повинні починатись з символу “@” та мати довжину не більше 100 символів. В контактні дані користувач вводить: у тип “Phone” – “380951234567”, у тип “Telegram” – “@PhotographerInKyiv”. Після цього користувач натискає кнопку Submit. |
| Очікуваний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |
| Фактичний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |

Таблиця 3.6 – Тест 1.6

|  |  |
| --- | --- |
| Тест | Створення профіля виконавця |
| Номер тесту | 1.6 |
| Початковий стан системи | Користувач авторизований, знаходиться на сторінці створення профіля виконавця |
| Вхідні дані | Текстовий опис себе як виконавця: “Professional photographer in Kyiv”, контактні дані: тип “Phone”: “380951234567”, тип “Telegram”: “@PhotographerInKyiv”. |
| Опис проведення тесту | У відповідні поля вводяться: опис виконавця – має бути непорожнім полем та довжиною до 255 символів. Користувач вводить в поле опису виконавця “Professional photographer in Kyiv”. Для валідації користувач повинен ввести контактні дані хоча б одного виду. Вводяться дані валідного виду для кожного типу: для “Phone” – мають вводитись лише цифри, номер повинен мати довжину 12 цифр, та починатись з 380, для інших типів-месенджерів – повинні починатись з символу “@” та мати довжину не більше 100 символів. В контактні дані користувач вводить: у тип “Phone” – “380951234567”, у тип “Telegram” – “@PhotographerInKyiv”. Після цього користувач натискає кнопку Submit. |
| Очікуваний результат | Створення профіля виконавця проходить успішно. |
| Фактичний результат | Створення профіля виконавця проходить успішно. |

Таблиця 3.7 – Тест 1.7

|  |  |
| --- | --- |
| Тест | Перемикання на режим виконавця в плаваючому меню |
| Номер тесту | 1.7 |
| Початковий стан системи | Користувач авторизований, створив профіль виконавця, знаходиться на сайті |
| Вхідні дані | - |
| Опис проведення тесту | В хедері сайту користувач натискає на елемент, що відображає поточний режим – “Customer”, після чого відображається плаваюче меню. На ньому користувач повинен натиснути на елемент “Contractor” |
| Очікуваний результат | Змінюється режим в хедері на Contractor, в хедері відображаються відповідні для ролі виконавця елементи. |
| Фактичний результат | Змінюється режим в хедері на Contractor, в хедері відображаються відповідні для ролі виконавця елементи. |

Таблиця 3.8 – Тест 1.8

|  |  |
| --- | --- |
| Тест | Відхилення заявки |
| Номер тесту | 1.8 |
| Початковий стан системи | Користувач авторизований, має профіль виконавця, знаходиться на сторінці заявок та замовлень на виконання по послузі, має хоч одну заявку, що очікує розгляду |
| Вхідні дані | - |
| Опис проведення тесту | Користувач натискає на кнопку для відхилення заявки |
| Очікуваний результат | Заявка приховується зі сторінки виконавця. Статус заявки змінюється на Declined. |
| Фактичний результат | Заявка приховується зі сторінки виконавця. Статус заявки змінюється на Declined. |

Таблиця 3.9 – Тест 1.9

|  |  |
| --- | --- |
| Тест | Прийом заявки |
| Номер тесту | 1.9 |
| Початковий стан системи | Користувач авторизований, має профіль виконавця, знаходиться на сторінці заявок та замовлень на виконання по послузі, має хоч одну заявку, що очікує розгляду |
| Вхідні дані | - |
| Опис проведення тесту | Користувач натискає на кнопку для прийняття заявки |
| Очікуваний результат | Заявка приховується зі сторінки виконавця. Створюється замовлення на основі даної заявки зі статусом Not started. |
| Фактичний результат | Заявка приховується зі сторінки виконавця. Створюється замовлення на основі даної заявки зі статусом Not started. |

Таблиця 3.10 – Тест 1.10

|  |  |
| --- | --- |
| Тест | Видалення відхиленої заявки |
| Номер тесту | 1.10 |
| Початковий стан системи | Користувач авторизований, створив хоч одну заявку, знаходиться на сторінці власних заявок та замовлень, має заявку в статусі Declined |
| Вхідні дані | - |
| Опис проведення тесту | Користувач натискає на кнопку для видалення заявки |
| Очікуваний результат | Заявка приховується зі сторінки, на якій знаходиться зазмовник. Заявка видаляється з системи. |
| Фактичний результат | Заявка приховується зі сторінки, на якій знаходиться замовник. Заявка видаляється з системи. |

Таблиця 3.11 – Тест 1.11

|  |  |
| --- | --- |
| Тест | Зміна статусу замовлення при початку роботи над ним |
| Номер тесту | 1.11 |
| Початковий стан системи | Користувач авторизований, має профіль виконавця, знаходиться на сторінці заявок та замовлень на виконання по послузі, має хоч одне замовлення в статусі Not started. |
| Вхідні дані | - |
| Опис проведення тесту | Користувач натискає на кнопку Start на елементі замовлення зі статусом Not Started. |
| Очікуваний результат | Статус замовлення змінюється з Not started в In progress |
| Фактичний результат | Замовлення приховується зі сторінки виконавця. Створюється замовлення на основі даної заявки зі статусом Not started. |

Таблиця 3.12 – Тест 1.12

|  |  |
| --- | --- |
| Тест | Зміна статусу замовлення при початку роботи над ним |
| Номер тесту | 1.12 |
| Початковий стан системи | Користувач авторизований, має профіль виконавця, знаходиться на сторінці заявок та замовлень на виконання по послузі, має хоч одне замовлення в статусі In progress. |
| Вхідні дані | - |
| Опис проведення тесту | Користувач натискає на кнопку Finish на елементі замовлення зі статусом In progress. |
| Очікуваний результат | Статус замовлення змінюється з Not started в In progress |
| Фактичний результат | Замовлення приховується зі сторінки виконавця. Створюється замовлення на основі даної заявки зі статусом Not started. |

Таблиця 3.13 – Тест 1.13

|  |  |
| --- | --- |
| Тест | Створення заявки на послугу |
| Номер тесту | 1.13 |
| Початковий стан системи | Користувач авторизований, знаходиться на сторінці власних заявок та замовлень, має хоч одне замовлення в статусі Finished. |
| Вхідні дані | Оцінка послуги: 5, коментар: “They did good photos!” |
| Опис проведення тесту | Користувач натискає на кнопку Leave a rate на елементі замовлення зі статусом Finished. У формі, що з’явилась, користувач вводить оцінку – 5, та коментар: “They did good photos!”. Після введення даних користувач натискає кнопку Submit. |
| Очікуваний результат | Огляд успішно створюється. |
| Фактичний результат | Огляд успішно створюється. |

## Опис контрольного прикладу

Почнемо тестування з реєстрації акаунта для замовника. На сторінці реєстрації вводимо креденціали і натискаємо кнопку Sign up:

A screen shot of a login form

Description automatically generated

Рисунок 3.3 - Сторінка реєстрації користувача

Перенаправлені на головну сторінку. Далі зайдемо в меню з хедеру:

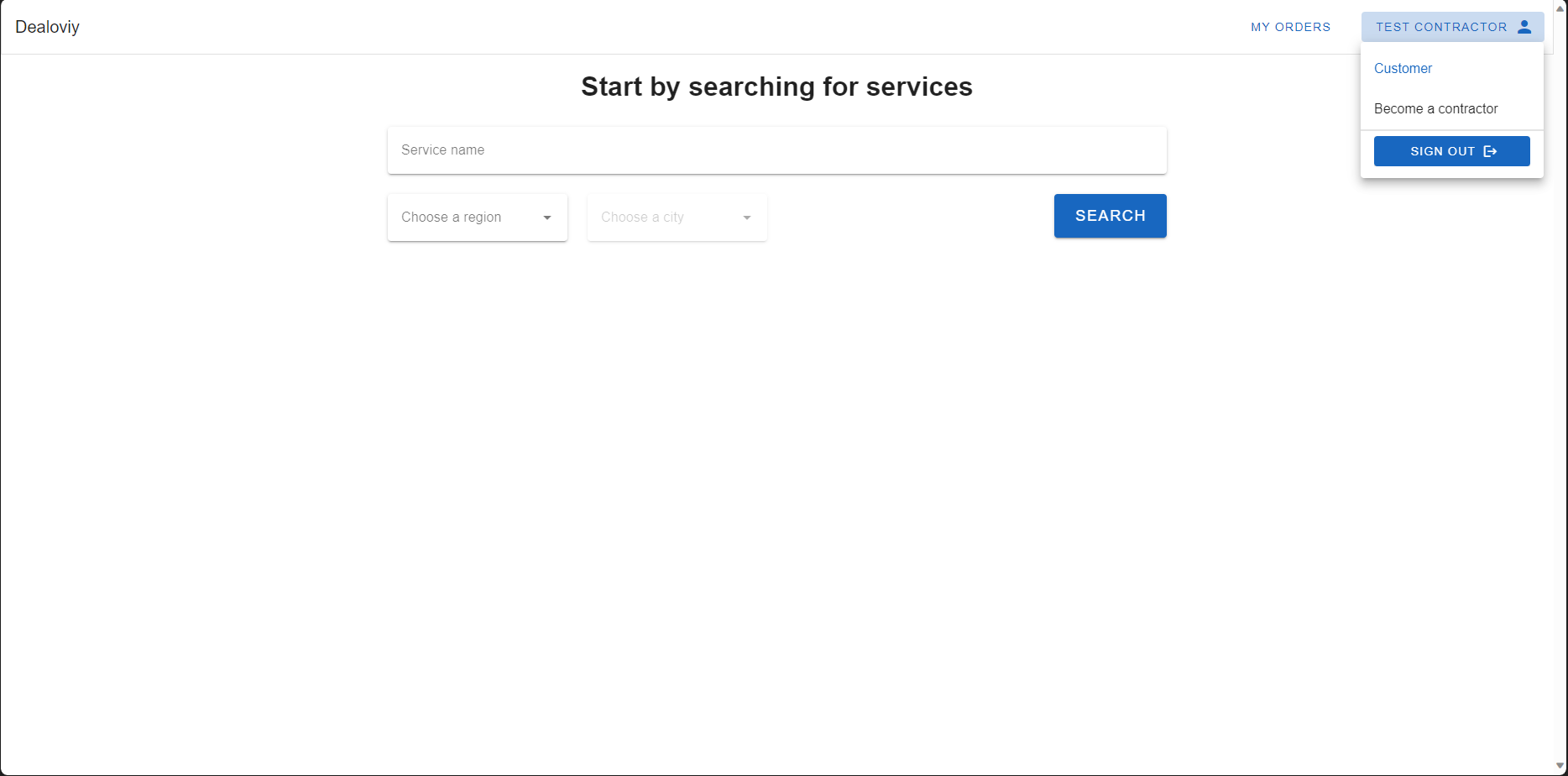


Рисунок 3.4 - Головна сторінка з відкритим меню користувача

В меню натискаємо Become a contractor - переходимо на сторінку створення профіля виконавця:

A screenshot of a computer

Description automatically generated

Рисунок 3.5 - Сторінка створення профіля користувача

Заповнили опис, натискаємо Add contact - відкривається форма:

A screenshot of a computer

Description automatically generated

Рисунок 3.6 - Форма додавання контактних даних

Додамо два види контактних даних:

A screenshot of a computer

Description automatically generated

Рисунок 3.7 - Сторінка створення профіля виконавця з доданими контактними даними

Після створення профіля користувача система перенаправляє нас на сторінку логіну. Входимо, використувуючи попередні креденціали, і знаходимо себе на головній сторінці.

A screenshot of a computer

Description automatically generated

Рисунок 3.8 - Головна сторінка

Бачимо нові елементи в хедері. Натискаємо на My Services.

A screenshot of a computer

Description automatically generated

Рисунок 3.9 - Сторінка власних послуг

На сторінці бачимо кнопку для створення послуги, натискаємо – відкривається форма.

A screenshot of a computer

Description automatically generated

Рисунок 3.10 - Форма створення послуги

Після введення в форму даних натискаємо Add service – форма закривається. Оновлюємо сторінку і бачимо власну послугу:

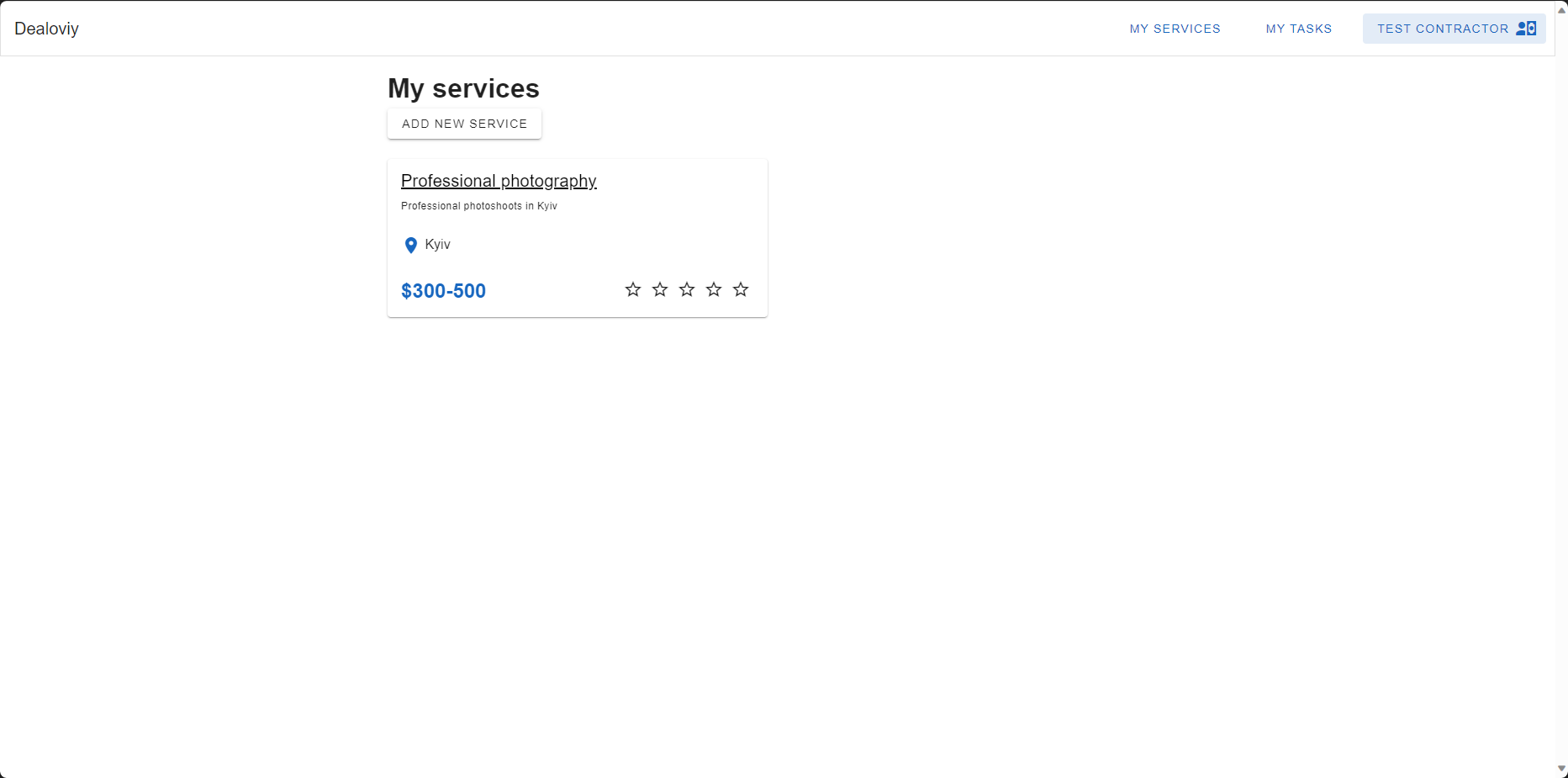


Рисунок 3.11 - Відображення створеної послуги

Тепер зареєструємо замовника:

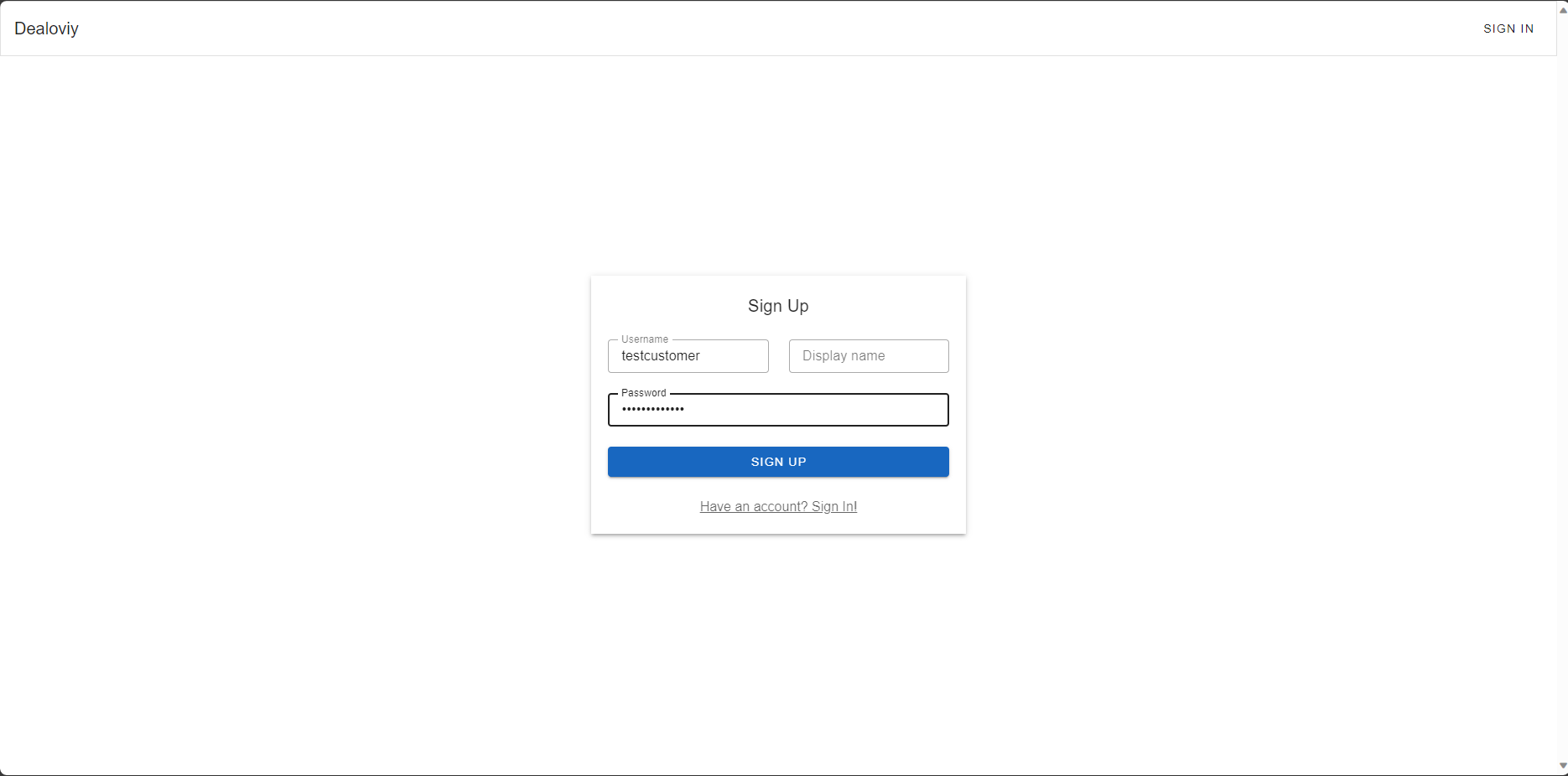


Рисунок 3.12 - Сторінка реєстрації

Після реєстрації ми перенаправлені на головну сторінку. Здійснимо пошук за критеріями, щоб знайти послугу виконавця, бачимо пошукові результати:

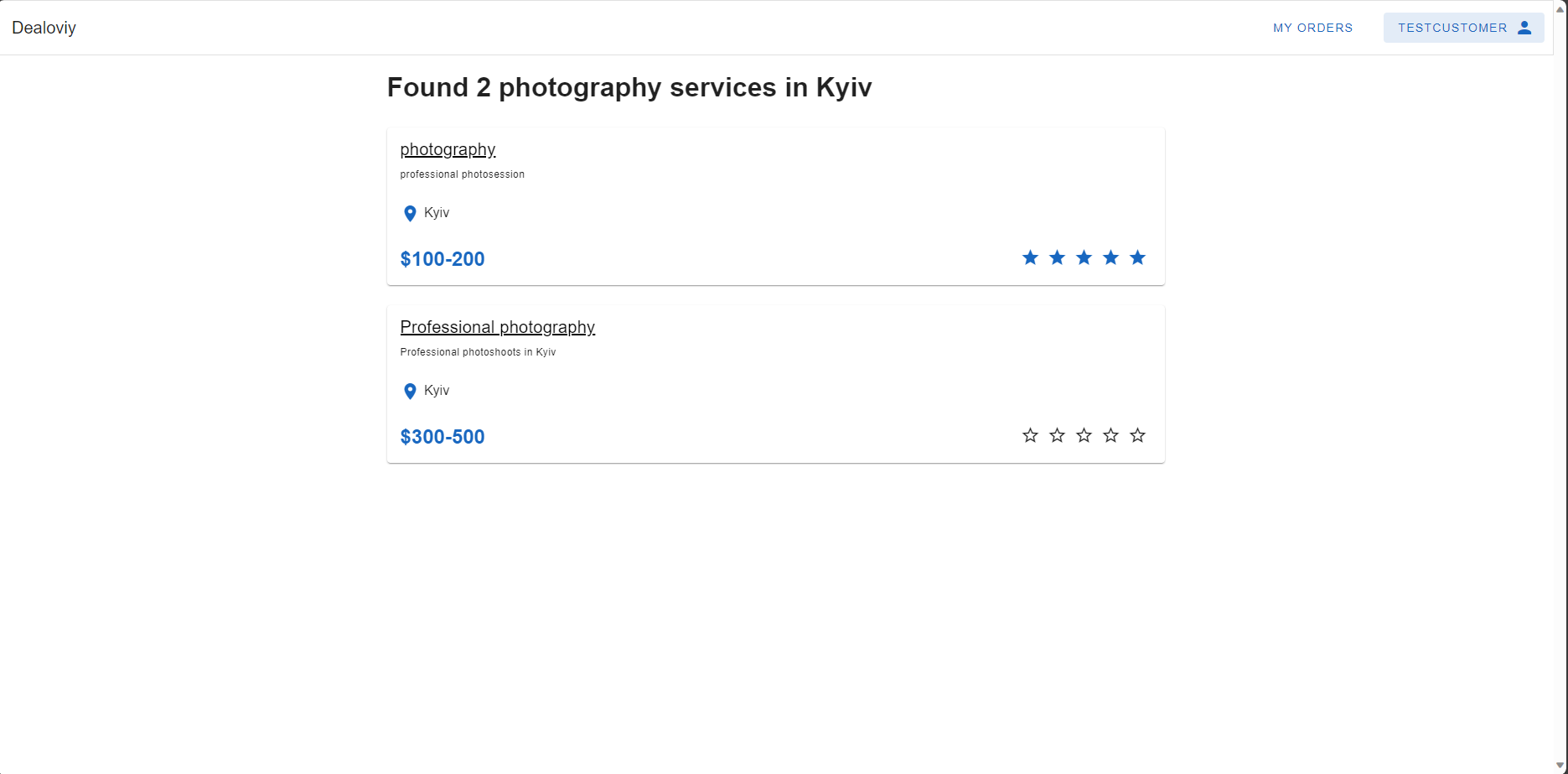


Рисунок 3.13 - сторінка пошукових результатів

Бачимо пошукові результати, переходимо на послугу Professional photography:

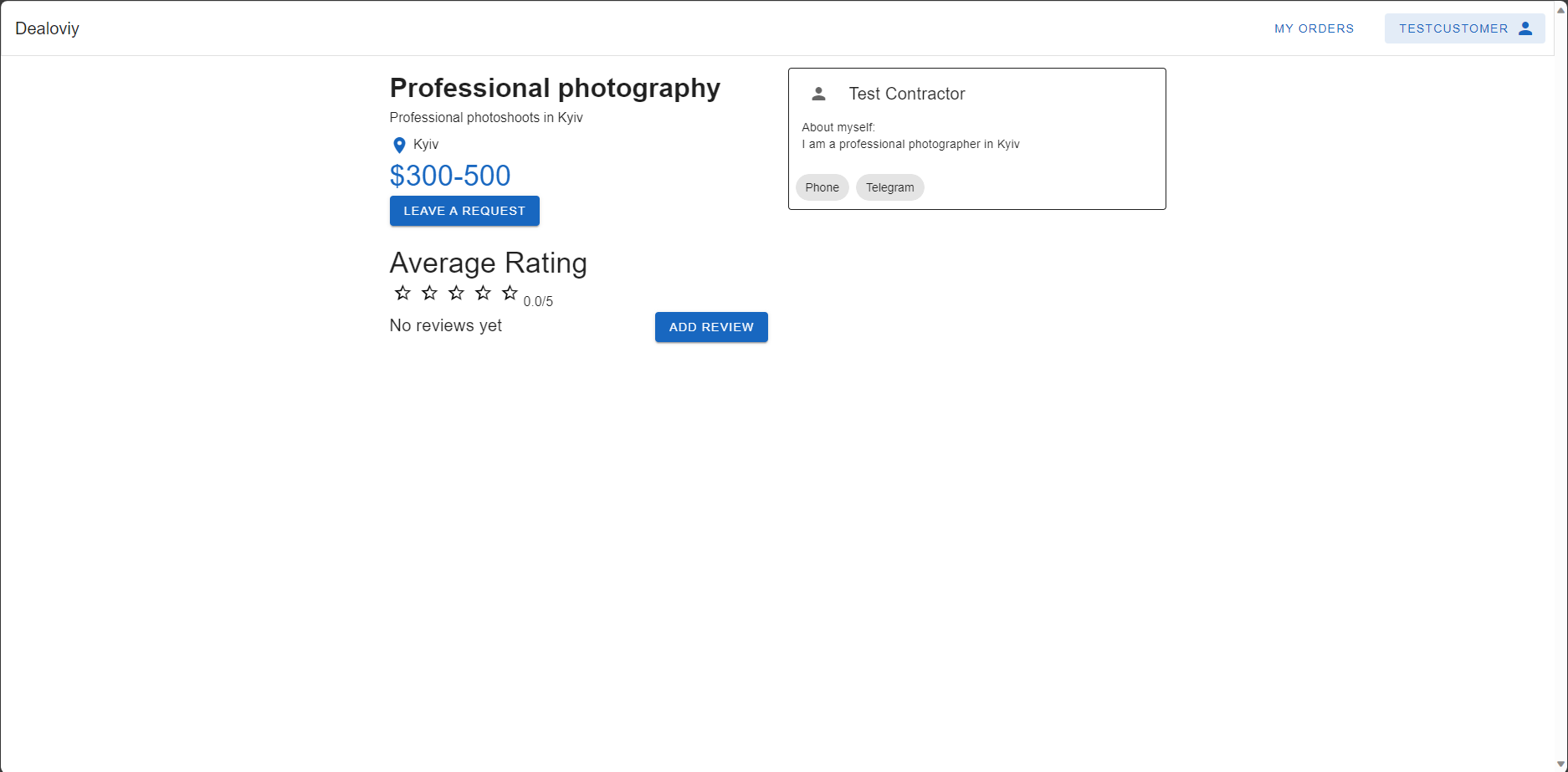


Рисунок 3.14 - Сторінка послуги

Бачимо сторінку послуги. Натискаємо кнопку Leave a request - відкривається форма:

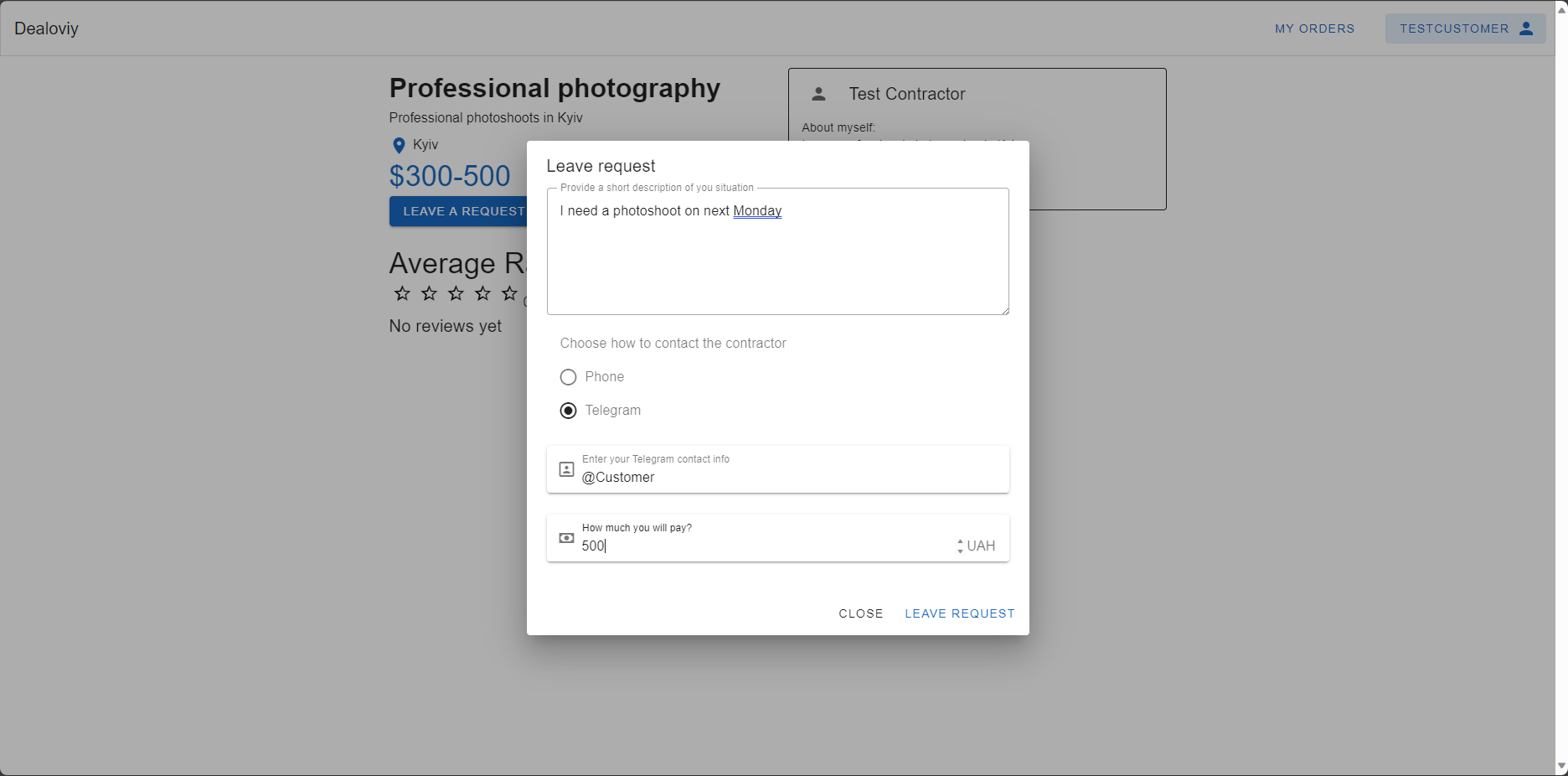


Рисунок 3.15 - Форма подання заявки

Вводимо дані для заявки, і натискаємо Leave request - форма закривається. Тепер на хедері натиснемо на My orders - переходимо на сторінку власних заявок і замовлень:

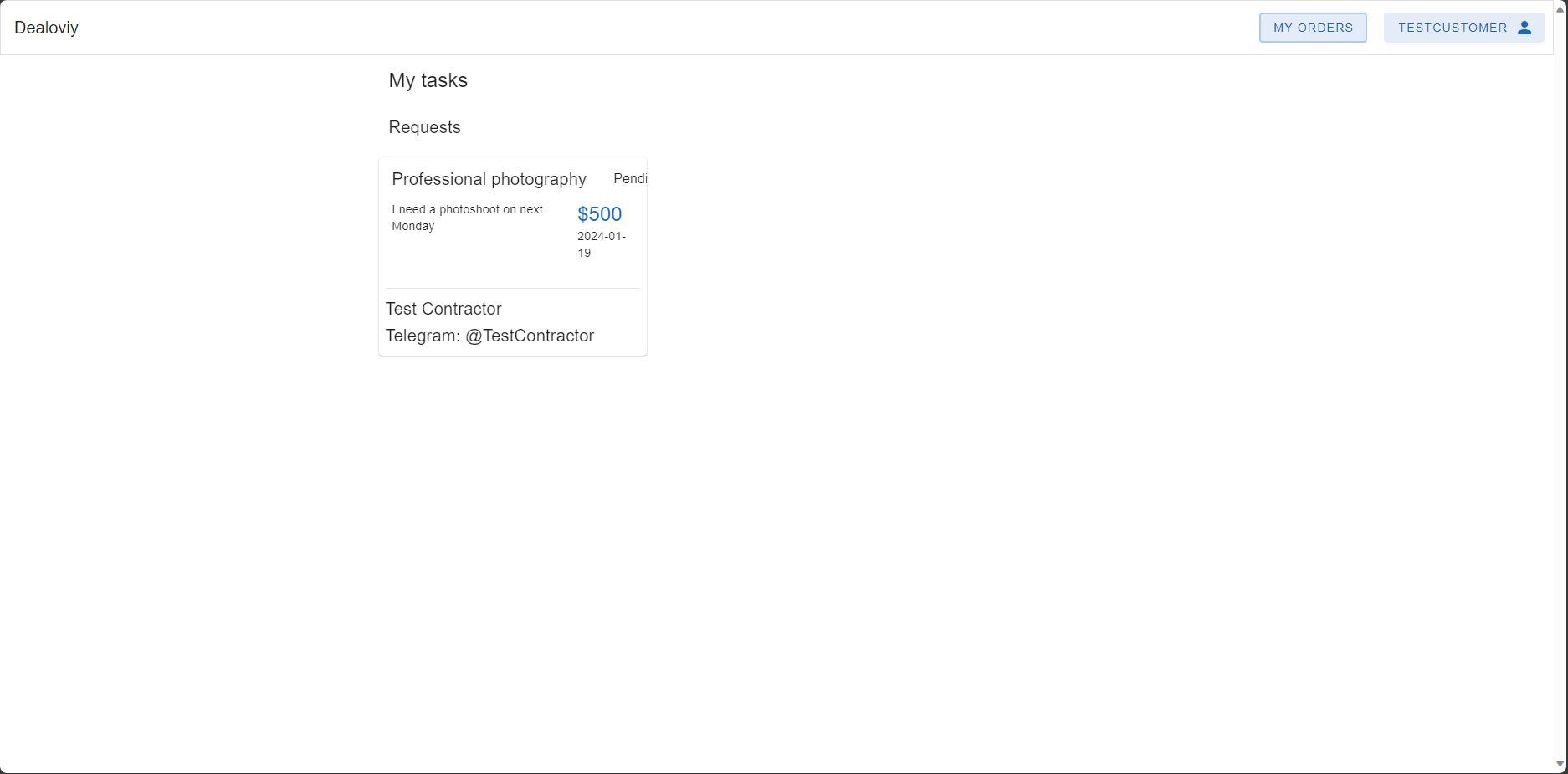


Рисунок 3.16 - Сторінка власних заявок та замовлень

Бачимо щойно створену заявку, контактні дані виконавця обраного типу, і статус Pending.

Повернемось до акаунту виконавця, натиснемо на хедері на елемент My tasks - переходимо на сторінку:

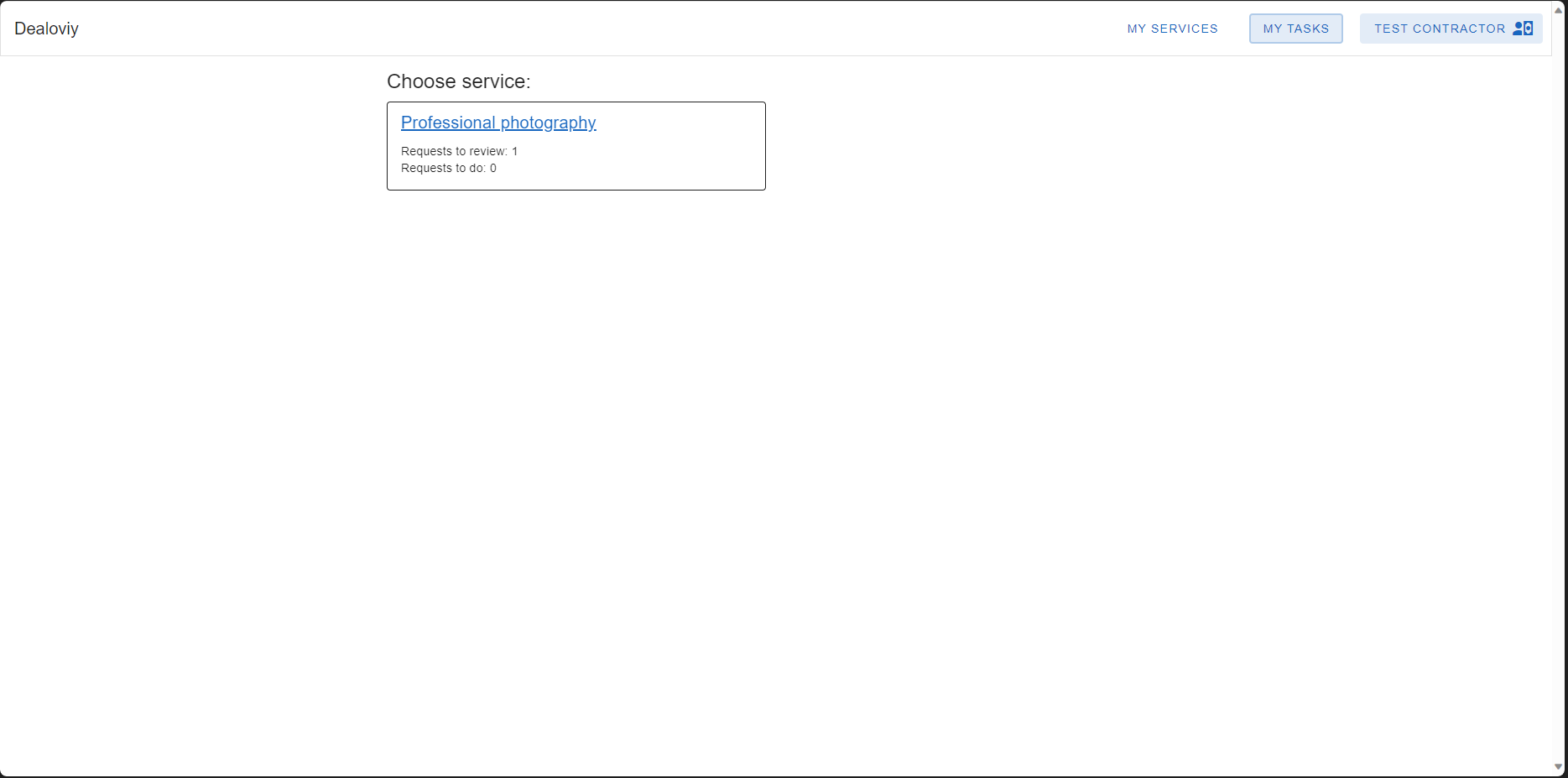


Рисунок 3.17 - Сторінка вибору послуги для перегляду заявок і замовлень

На сторінці бачимо елемент з нашою послугою, і що там нас очікує 1 заявка. Натискаємо:

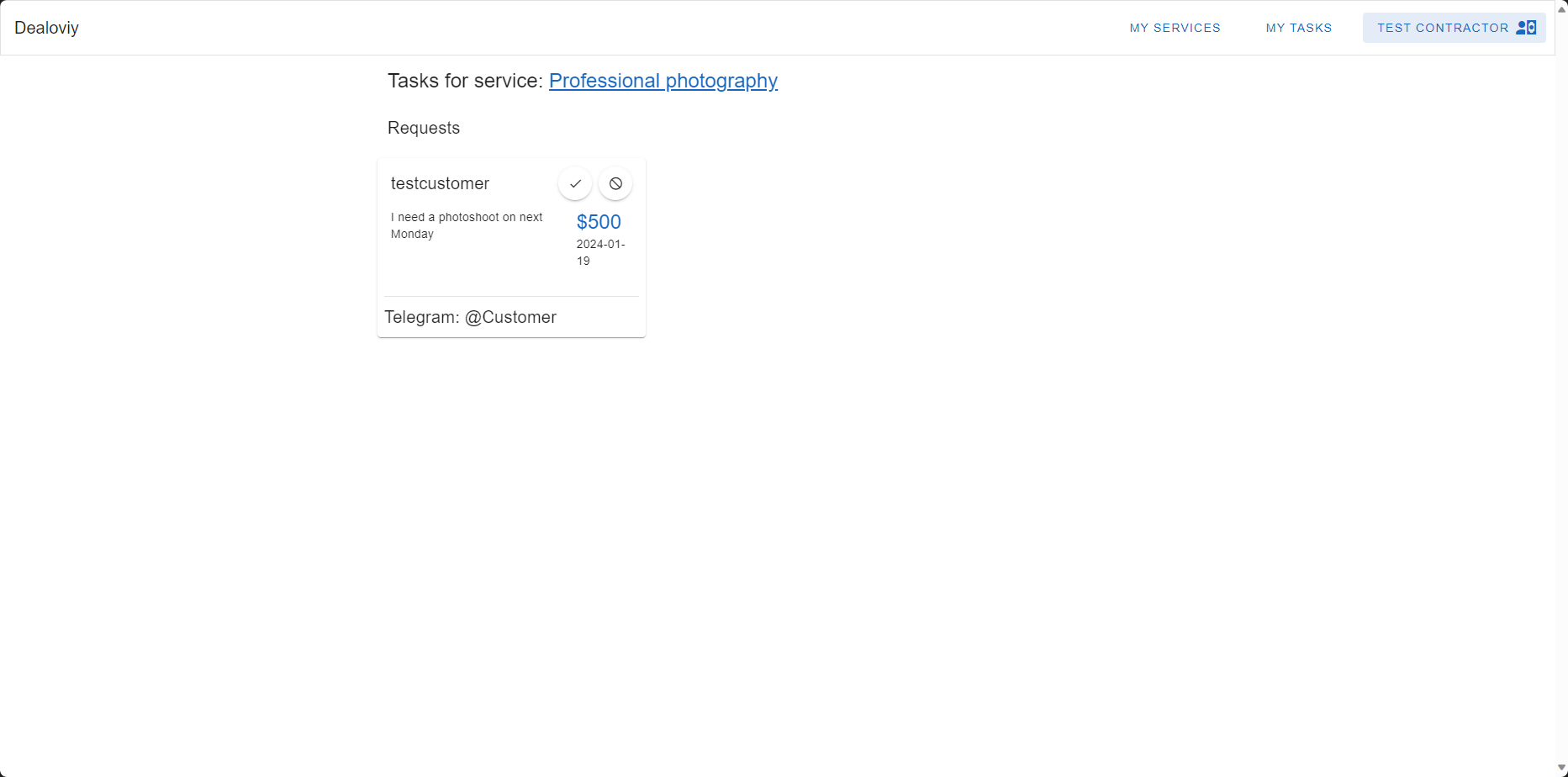


Рисунок 3.18 - Сторінка заявок і замовлень по послузі виконавця

Знаходимось на сторінці заявок і замовлень по даній послузі, і бачимо заявку - приймемо її.

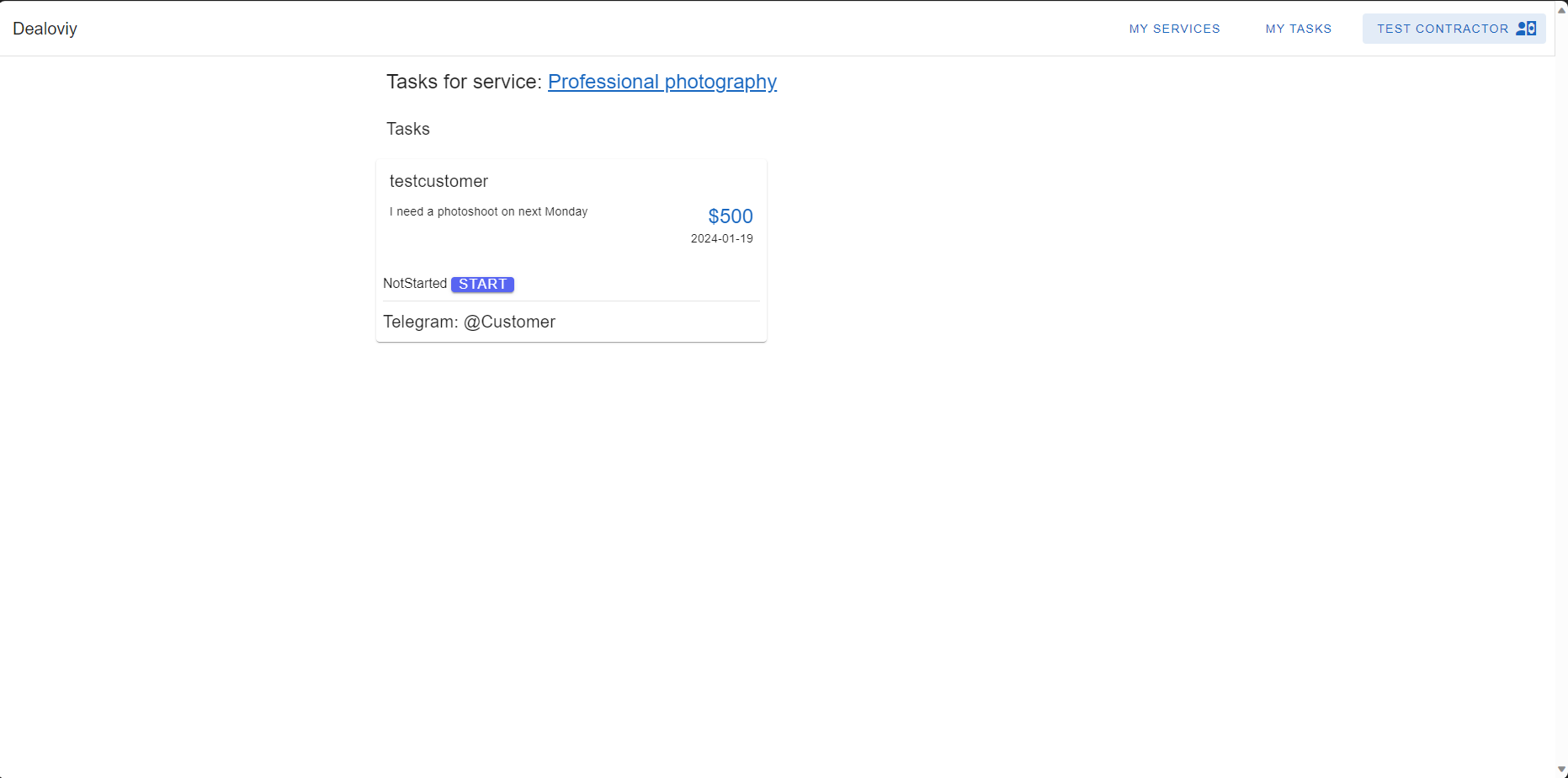


Рисунок 3.19 - Сторінка заявок та замовлень по послузі з новим замовленням на ній

Тепер на сторінці маємо замовлення, що ще не почате. На сторінці у замовника буде видно замовлення:

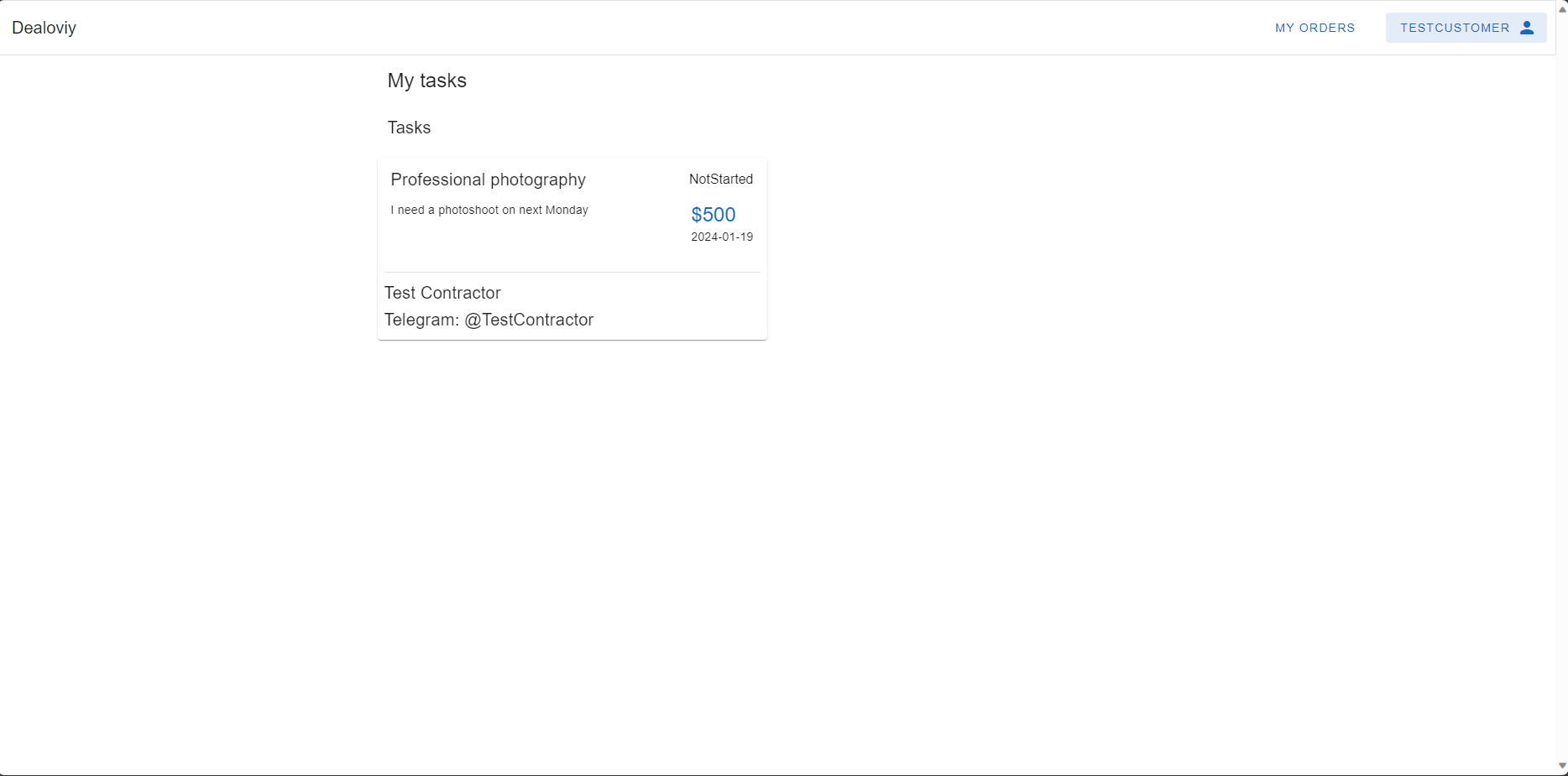


Рисунок 3.20 - Сторінка заявок у замовника, нове замовлення на ньому

Повернемось до виконавця, на замовленні натиснемо Start - зміниться статус замовлення:

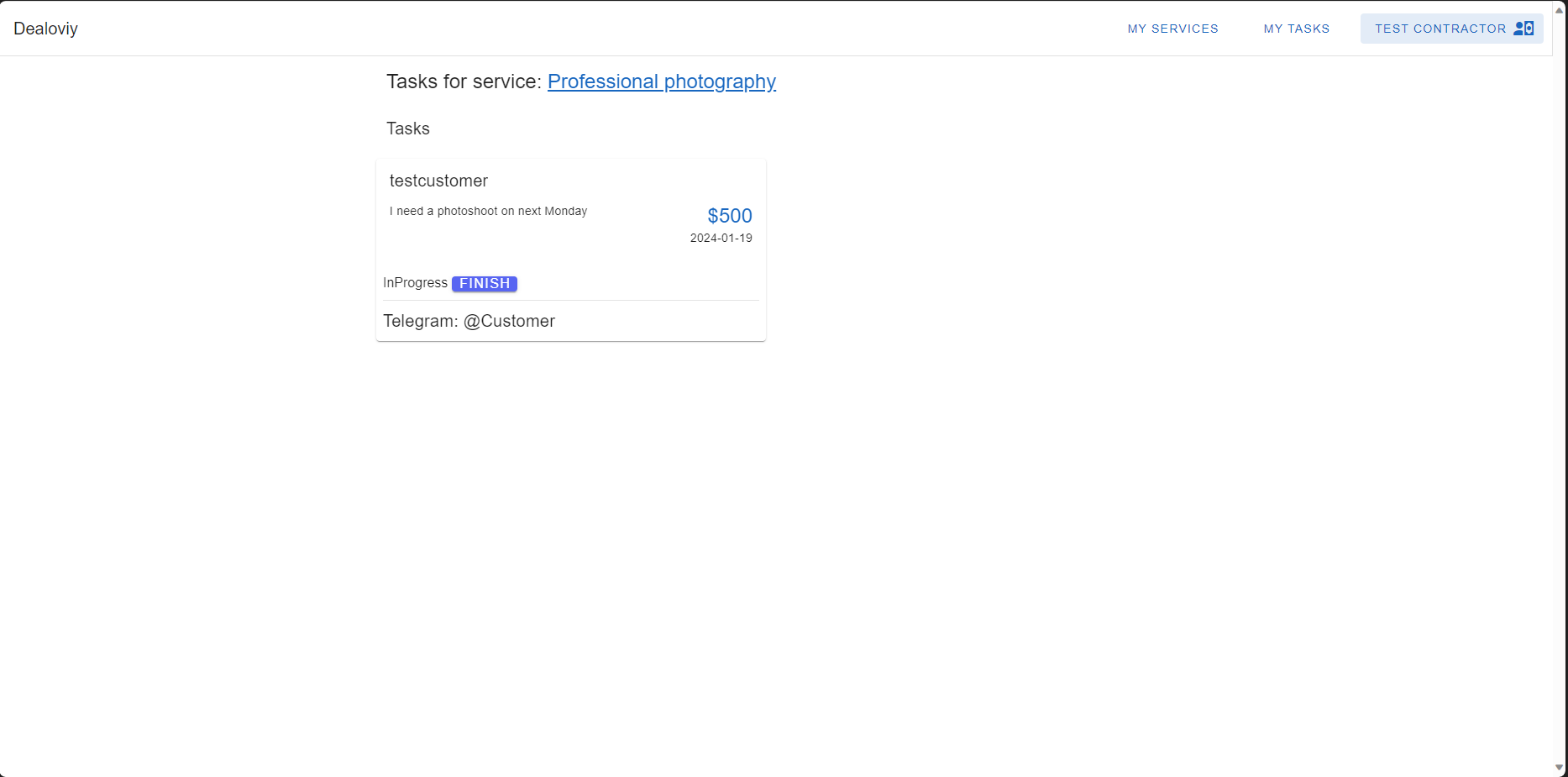


Рисунок 3.21 - Сторінка заявок виконавця, замовлення в процесі

Як бачимо, статус змінився. У виконавця на сторінці статус теж оновлено:

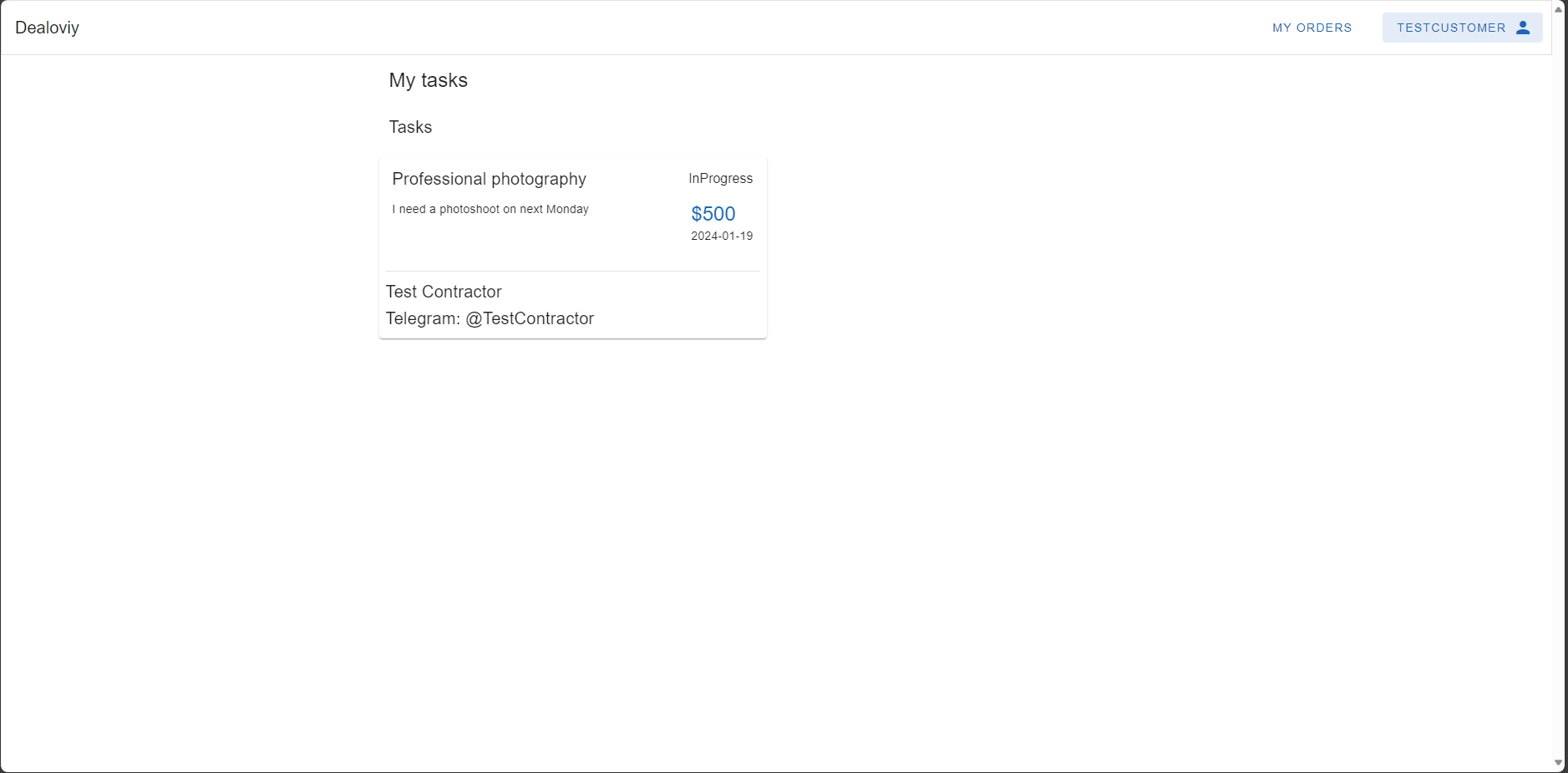


Рисунок 3.22 - Сторінка заявок замовника, замовлення в процесі

Знов на сторінці виконавця, натиснемо на Finish:

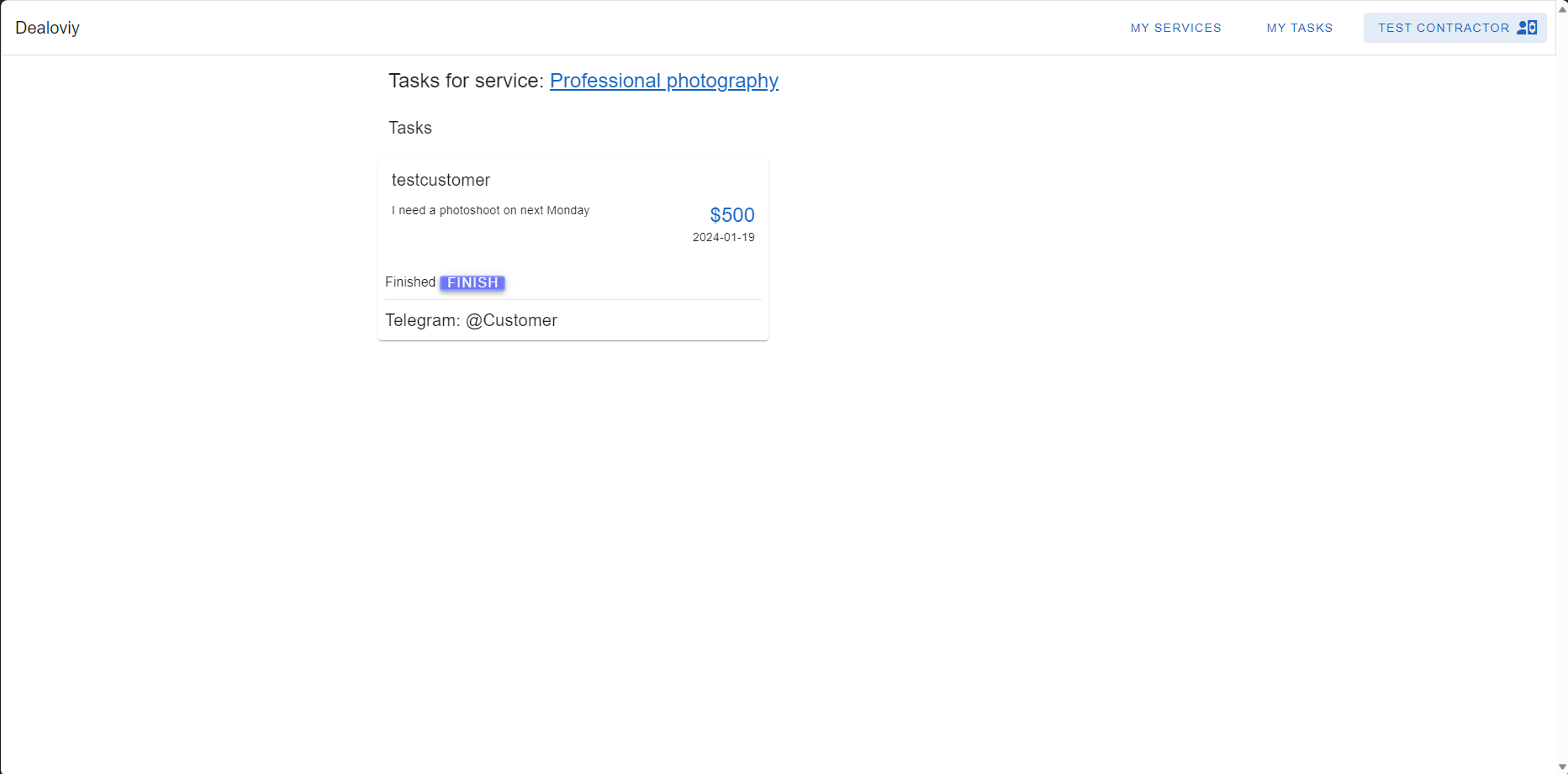


Рисунок 3.23 - Сторінка заявок виконавця, замовлення завершено

Статус змінився, переглянемо на сторінці замовника:

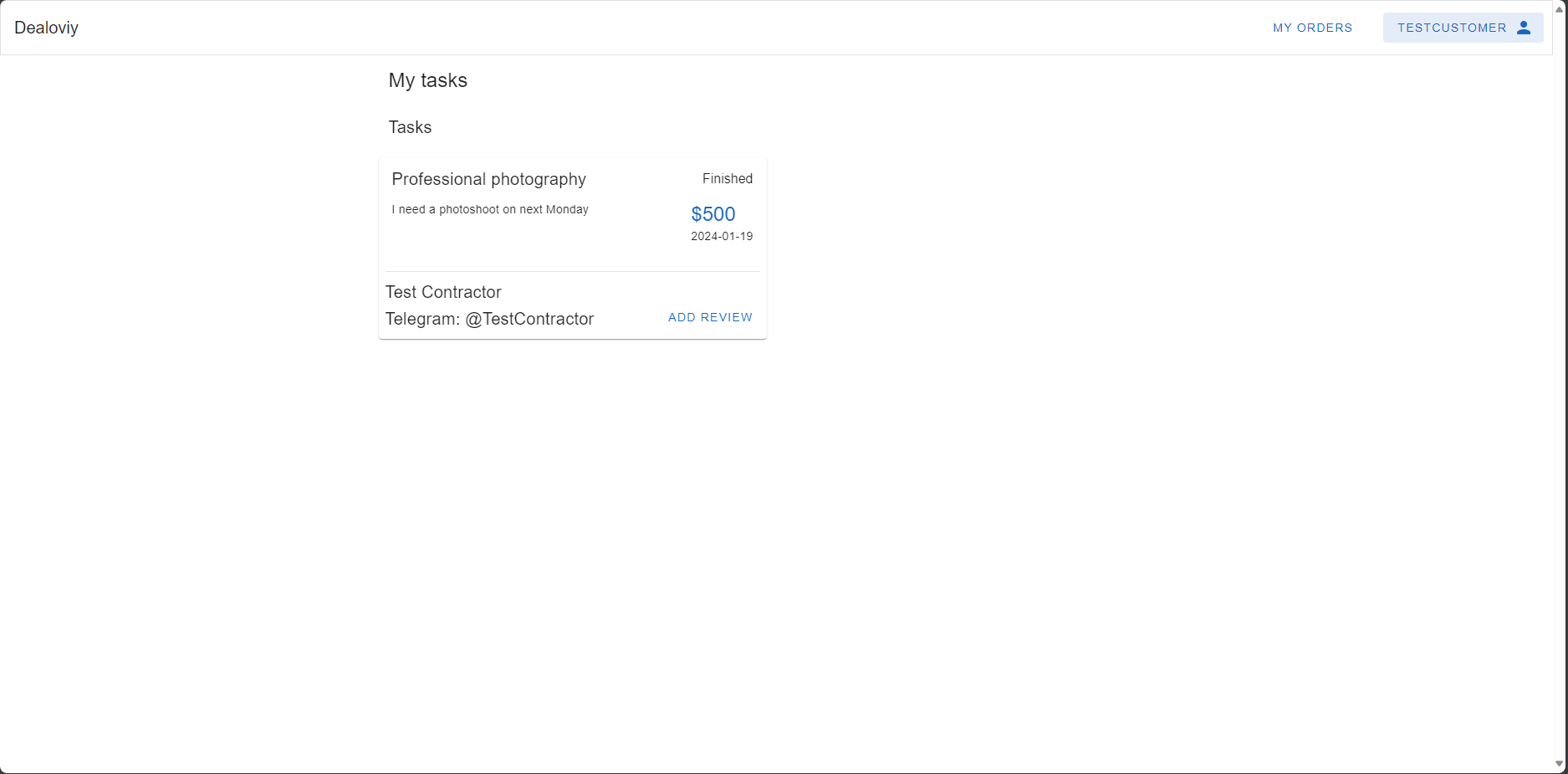


Рисунок 3.24 - Сторінка заявок замовника, замовлення завершено

Бачимо, що замовлення завершено, і бачимо кнопку Add review. Натиснемо, щоб додати огляд - відкривається форма:

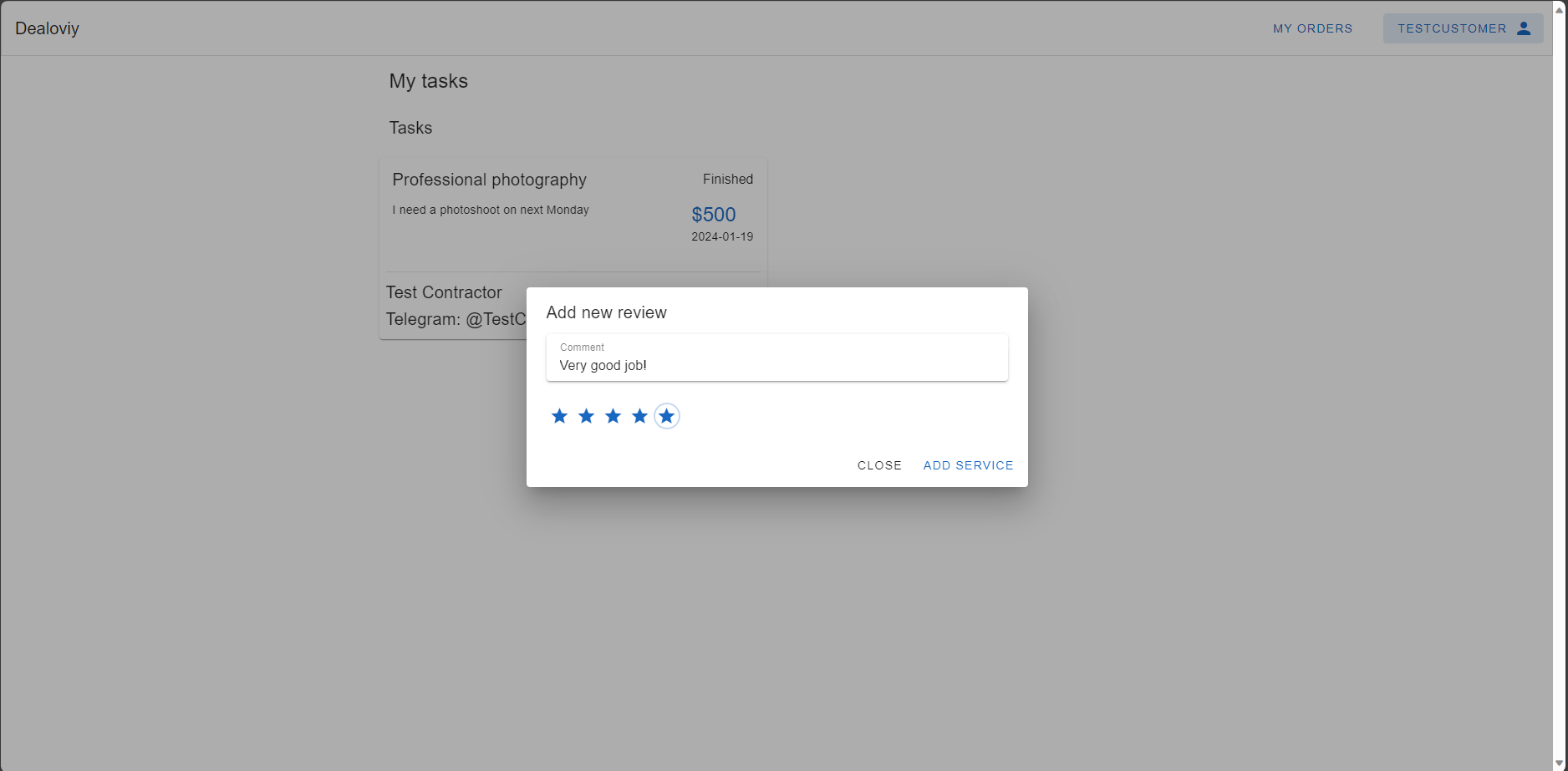


Рисунок 3.25 - Форма створення огляду

Ввели дані, натискаємо кнопку для створення огляду, форма закривається.

Перейдемо на акаунт виконавця, на сторінку My services:

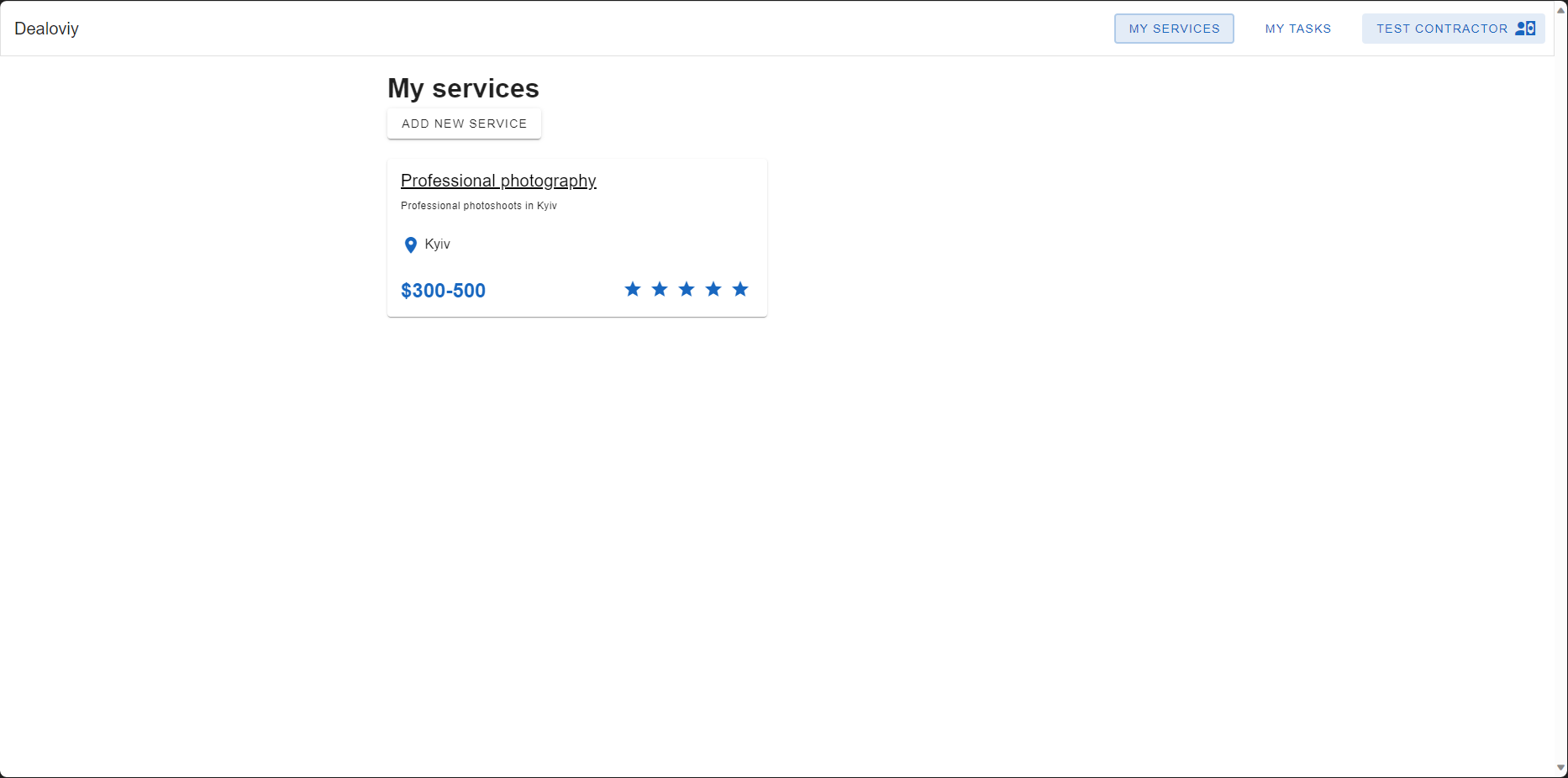


Рисунок 3.26 - Сторінка власних послуг виконавця

Бачимо, що на послузі новий рейтинг, перейдемо на сторінку послуги:

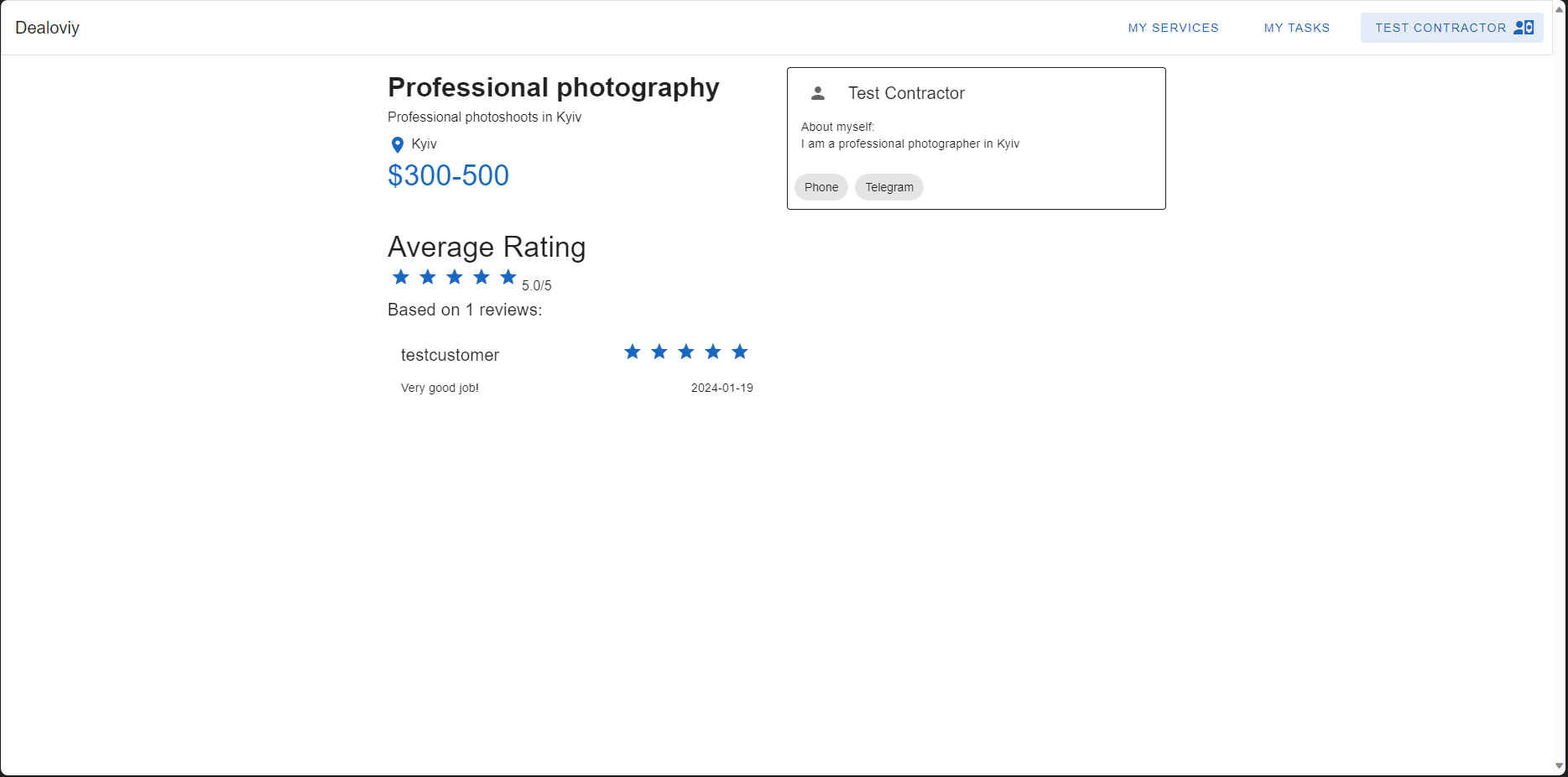


Рисунок 3.27 - Сторінка послуги з новим оглядом на ній

Бачимо новий огляд від замовника.

## Висновки до розділу

У даному розділі було проведено детальне тестування функціоналу веб-застосунку. Проводилось шляхом мануального тестування, яке відтворювало використання веб-застосунку в умовах реального світу, з визначениим вхідними даними та порівнянням очікуваних та фактичних результатів.

Було побудовано контрольний приклад, що охоплює усі функціональні можливості програмного забезпечення: реєстрація, створення профіля виконавця, створення послуги, пошук послуг, створення заявки, відхилення та прийом заявки, зміна статусу замовлення та залишення огляду на надану послугу.

Дані заходи переконали, що розроблений веб-застосунок працює коректно, відповідає очікуванням в роботі та загальній стабільній роботі.

# ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Розгортання програмного забезпечення

Веб-застосунок розгортається локально. Для цього обов’язково на пристрій має бути встановлено .NET 6, .NET CLI [15], Vue CLI [16]. На пристрої повинен бути локальний сервер MS SQL Server з назвою MSSQLSERVER01. Опціонально може бути встановлений git для клонування репозиторію замість завантаження вихідного коду.

Щоб завантажити код веб-застосунку на свій пристрій, потрібно клонувати Github-репозиторій, або ж скачати вміст репозиторію архівом та розпакувати в необхідну директорію.

Після цього переходимо в директорію клонованого репозиторію, і переходимо в директорію backend/Dealoviy.WebApi/, запускаємо термінал в ній та вводимо команду dotnet build – це підключить необхідні залежності та збудує проект. Далі вводимо команду dotnet ef database update, щоб створити базу даних для системи. Останній крок у бекенді – ввести команду dotnet run – наш серверний додаток запущений.

Після цього повертаємось на рівень директорії репозиторію,

переходимо в директорію frontend. Перед початком роботи потрібно створити файл з розширенням .env файл, і внести в нього змінну: VITE\_API\_URL=https://localhost:7107/api. Після цього відкриваємо термінал в директорії і вводимо команду npm install для попереднього встановлення усіх залежностей, а потім команду npm run dev для запуску фронтенду. Після запуску фронтенд-застосунок буде доступний під адресою <http://localhost:>5173.

## Підтримка програмного забезпечення

Щоб оновити застосунок, спочатку завершимо роботу фронтенду через термінал, в якому працює фронтенд, комбінацією Ctrl+C. Тією ж комбінацією завершуємо роботу бекенду.

Тепер необхідно отримати нову версію застосунку. Це можна зробити через git, зробивши пул гілки main, або скачати вміст репозиторію архівом та замінити файли в необхідній директорії. Після цього повторюємо процедуру запуску додатків.

## Висновки до розділу

В даному розділі було розглянуто такі аспекти, як розгортання та підтримка програмного забезпечення.

Підрозділ “Розгортання програмного забезпечення” детально описує вимоги та кроки, які необхідно здійснити для першого розгортання програмного забезпечення на пристрої користувача. Розглянуто можливості клонування репозиторію, та завантаження вихідного коду релізу архівом. Надано інструкції для збірки, створення та оновлення бази даних, запуску бекенду, і збірки та запуску фронтенду.

Підрозділ “Підтримка програмного забезпечення” акцентує увагу на отриманні оновленого коду з git або завантаження оновленого коду архівом, та запуск додатку нової версії.

В підсумку, розглянуто етапи, що забезпечать роботу додатку з подальшим оновленням функціоналу.

# ВИСНОВКИ

У результаті виконання курсової роботи було проаналізовано предметну область, спроектовано та змодельовано функціональні та системні вимоги, бізнес-правила, та реалізовано веб-застосунок системи пошуку виконавців та замовників послуг.

Було визначено мови програмування, платформи, інструменти, програмні застосунки, що будуть використовуватись у розробці. Розробка була проведена на мовах C#, Typescript, з застосуванням фреймворків .NET, ASP.NET Core для серверної частини, та Vue для клієнтської. Використано базу даних MS SQL Server. Серед інструментів: інтегровані середовища розробки: JetBrains Rider для бекенду, Visual Studio Code для фронтенду. В якості програмних пакетів було застосовано Nuget-пакет CsvHelper для роботи з файлами в форматі .csv.

Розроблений застосунок було протестовано мануально, та проаналізовано якість коду вбудованими в IDE методами.

Досягнено такі цілі розробки, як анонімність реєстрації та конфіденційність даних користувачів, а також алгоритм укладення угоди між замовником та виконавцем став простішим, ніж даний в аналозі, при тому виконує кінецву задачу:

1. Замість 4 розгалужень в аналозі алгоритм в розробленому застосунку не налічує жодних.
2. Замість 4 варіантів подання заявки на виконання, таких як публічно доступне завдання, публічно доступне оголошення, завдання, надіслане виконавцю з рейтингу найкращих в категорії, завдання, надіслане виконавцю зі списку послуг в категорії – маємо заявку, яку користувач подає на конкретну послугу, в якій бачить і дані про послугу, і дані про виконавця.
3. Замість варіантів виконання алгоритму, від початку до кінця, що містять від 8 кроків, маємо єдиний варіант виконання алгоритму, що налічує 6 кроків.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Putting Kropotkin Back into the Concept of Mutual Aid” [Електронний ресурс] // Science for the People – 2023 – Режим доступу до ресурсу: <https://magazine.scienceforthepeople.org/vol24-3-cooperation/putting-kropotkin-back-into-the-concept-of-mutual-aid/>
2. “The gig is up” – [Електронний ресурс] // The Harvard Gazette – 2023 – Режим доступу до ресурсу: <https://news.harvard.edu/gazette/story/2020/11/during-covid-19-remote-freelance-work-is-on-the-rise/>
3. “ВООЗ оголосила про закінчення пандемії COVID-19” – [Електронний ресурс] // Слово і Діло – 2023 – Режим доступу до ресурсу: <https://www.slovoidilo.ua/2023/05/05/novyna/svit/vooz-oholosyla-pro-zakinchennya-pandemiyi-covid-19>
4. “The Top 5 Freelancing Trends in 2024: The Future of New Work Unveiled” – [Електронний ресурс] // freelancermap – 2023 – Режим доступу до ресурсу: <https://www.freelancermap.com/blog/freelancing-trends-2024/>
5. “ЯК ЦЕ ПРАЦЮЄ?” – [Електронний ресурс] ManGoWorks – 2023 – Режим доступу до ресурсу: <https://mangoworks.com.ua/how-it-works>
6. “How can freelancers help your business?” – [Електронний ресурс] freelancer.com – 2023 – Режим доступу до ресурсу: <https://www.freelancer.com/info/how-it-works>
7. “Українські фріланс-біржі: ТОП-5 2022 року” – [Електронний ресурс] – sochka.com – 2022 – Режим доступу до ресурсу: <https://sochka.com/ukrainski-frilans-birzhi-top-5-2022-roku.html>
8. “Які види комерційних торгів доступні замовникам SmartTender.biz” – [Електронний ресурс] – SmartTender – 2024 – Режим доступу до ресурсу: <https://smarttender.biz/blog/view/yaki-vidi-komertsiynih-torgiv-dostupni-zamovnikam-smarttender-biz/>
9. “Create .NET apps faster with NuGet” – [Електронний ресурс] – NuGet – 2024 – Режим доступу до ресурсу: <https://www.nuget.org/>
10. “Ukraine Cities Database” – [Електронний ресурс] – SimpleMaps – 2024 – Режим доступу до ресурсу: <https://simplemaps.com/data/ua-cities>
11. “CsvHelper” – [Електронний ресурс] – NuGet – 2024 – Режим доступу до ресурсу: <https://www.nuget.org/packages/CsvHelper>
12. “Visual Studio vs. Jetbrains Rider Performance” – [Електронний ресурс] – ndepend – 2024 – Режим доступу до ресурсу: <https://blog.ndepend.com/visual-studio-vs-jetbrains-rider-performance/>
13. “DEFINITION of client-server” – [Електронний ресурс] – TechTarget – 2024 – Режим доступу до ресурсу: <https://www.techtarget.com/searchnetworking/definition/client-server>
14. “Monolithic Architecture” – [Електронний ресурс] – GeeksForGeeks – 2024 – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/monolithic-architecture/>
15. “.NET CLI overview” – [Електронний ресурс] – Microsoft Learn – 2024 – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/core/tools/>
16. “Vue CLI” – [Електронний ресурс] – Vue.js – 2024 – Режим доступу до ресурсу: <https://cli.vuejs.org/>