МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

**ЗЕЛІНСЬКИЙ МАКСИМ ОЛЕГОВИЧ**

Допускається до захисту:

в.о. завідувача кафедри
інформаційних технологій

канд. техн. наук, доцент

_____ О. В. Зелінська

«____»_____ 2024 р.

# ВЕБДОДАТОК ПОШУКУ ВИКОНАВЦІВ ТА ЗАМОВНИКІВ ПОСЛУГ

Спеціальність 122 «Комп’ютерні науки»

**Кваліфікаційна (бакалаврська) робота**

Керівник:
Ю. В. Поремський, ст. викладач
кафедри інформаційних технологій,
к. т. н., ст. викладач

_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____

Вінниця – 2024

## АНОТАЦІЯ

**Зелінський М.О. Вебдодаток пошуку виконавців та замовників послуг.** Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

Основна мета цієї кваліфікаційної роботи - розробка онлайн платформи для ефективного пошуку виконавців та замовників послуг у різних галузях. Зі зростанням кількості фахівців та підприємців виникає потреба в централізованому ресурсі, який спростить процес взаємодії між замовниками та виконавцями, підвищуючи ефективність співпраці.

У вступі обґрунтовано актуальність розробки веб-додатку для пошуку виконавців та замовників послуг. Перший розділ присвячено аналізу існуючих конкурентів та дослідженню цільової області. Другий розділ описує інструменти та методи, використані при створенні додатку. У третьому розділі представлено інтерфейс користувача, а також особливості реалізації клієнтської та серверної частин веб-додатку.

**Ключові слова**: C#, ASP NET, React, Web App

54 с., 28 рис., 40 джерел.



## ABSTRACT

**Zelinskyi M.O. Web Application for Searching Performers and Customers of Services.** Specialty 122 «Computer Science» educational program «Computer Science». Vasyl Stus Donetsk National University, Vinnytsia, 2024.

The main objective of this qualification work is to develop an online platform for the efficient search of service providers and clients across various industries. With the increasing number of specialists and entrepreneurs, there is a growing need for a centralized resource that simplifies the interaction process between clients and service providers, thereby enhancing the efficiency of collaboration.

The introduction substantiates the relevance of developing a web application for the search of service providers and clients. The first chapter is dedicated to the analysis of existing competitors and the exploration of the target area. The second chapter describes the tools and methods used in the creation of the application. The third chapter presents the user interface, as well as the implementation features of the client-side and server-side of the web application.

**Keywords:** C#, ASP NET, React, Web App

54 p., 28 figures, 40 sources.

# ЗМІСТ

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Історія створення веб-сайтів

Поява Інтернету стала першою важливою подією, що дала можливість створювати сайти. Це відбулося в 1990 році, і спочатку Інтернет без інформаційного наповнення був непотрібним винаходом. Для того щоб він став корисним, необхідно було наповнити його змістом. Цим зайнявся вчений з Женеви Тімоті Бернерс-Лі, який вирішив цю проблему. Перед створенням першого веб-сайту в історії, він розробив багато важливих інструментів для підтримки найвідоміших додатків. Завдяки йому ми маємо Всесвітню павутину, URL-адреси, HTTP, а також мову HTML. [1]

Розробники Інтернету вважали, що їхні винаходи не лише спрощують процес збору та збереження інформації, але й допоможуть отримати до неї доступ. З огляду на сучасні тенденції використання Всесвітньої павутини, можна впевнено сказати, що всі ідеї вчених майже повністю втілилися в життя.

Спочатку проект був задуманий як геніальна і унікальна система "папок" для зберігання документів в Інтернеті, пов'язаних між собою особистими доменними іменами. Веб-сайти та веб-проекти, як старі, так і сучасні, зберігають різні документи та сторінки, фактично являючи собою "шматки" інформації, що належать одній юридичній особі, але доступні багатьом користувачам за визначеним доменним ім'ям.

Перший веб-сайт, створений одним вченим, мав доволі простий вигляд. Він складався з однієї білої домашньої сторінки з інформацією про Всесвітню павутину, яка на той час була найінноваційнішою технологією. Тім Бернерс-Лі детально описав, як встановити сервер браузера на персональний комп'ютер, і цю технологію World Wide Web можна вважати початком сучасної інтернет-мережі.

Система включала такі технічні засоби, як протокол передачі даних HTTP, спеціальну мову HTML для розмітки тексту і систему веб-адрес, відомих як URL.

Використовуючи розроблені ним принципи, включаючи роботу і встановлення браузерів і серверів, Тім Бернерс-Лі створив перший у світі веб-сайт. [2]

Згодом цей перший веб-сайт також став першим каталогом, і на нього почали посилатися численні нові сайти, створені іншими вченими. Завдяки експериментам і розробкам, винахідник Всесвітньої павутини створив такі унікальні речі, як перший браузер для WorldWideWeb, який включав віртуальний редактор і підтримку гіпертексту. Він також створив перший віртуальний сервер і запустив перші веб-сторінки.

Творець Інтернету був переконаний, що гіпертекст може стати основою для мережевого обміну даними, і наполегливо працював над тим, щоб впровадити цю концепцію в повсякденне життя. Врешті-решт йому це вдалося. Тім Бернерс-Лі приписують створення основних технологій Інтернету, таких як HTML, HTTP і URL, навіть якщо інші вчені мали схожі ідеї.

Ще в 40-х роках 20 століття американець Ванневар Буш розмірковував над проблемою використання технологічних ресурсів для розширення людської пам'яті та був зацікавлений у систематизації і організації загального доступу до знань та інформації, накопичених людством протягом століть.

Американські вчені Даг Енгельбарт і Теодор Нельсон розробили технологію, яку назвали гіпертекстом. Ця технологія дозволяла текстам "розгалужуватися", пропонуючи читачам різні варіанти їхнього читання. Нельсон прагнув створити систему перехресних посилань на всі тексти, написані людьми. [3]

Перший веб-сайт також став першим у світі онлайн-каталогом. Згодом розробники почали створювати великі каталоги своїх проектів, представляючи інформацію на інших сайтах і перенаправляючи користувачів на свої сторінки через посилання в цих каталогах.

Сьогодні масштаби того, що Тім Бернерс-Лі започаткував, вражають: його винаходи спричинили інноваційну революцію, забезпечили процвітання сучасних інтернет-технологій та онлайн-бізнесу. Кількість веб-сторінок нині налічує мільярди.

## 1.2 Визначення поняття веб-сайтів та принцип роботи

Для початку розглянемо термін веб-сторінка. Це електронний документ, який може містити текст, зображення, відео та посилання, і є основним елементом веб-сайту, який відображається у веб-браузері користувача. [4]

Веб-сторінки створюються за допомогою мов розмітки, таких як HTML і XHTML, що визначають структуру та формат вмісту сторінки. Медіа-елементи, зображення та відео можна вбудовувати за допомогою відповідних тегів і посилань на зовнішні ресурси. Окрім текстового контенту, веб-сторінки можуть містити посилання, що дозволяють користувачам переходити на інші сторінки та ресурси, або інтерактивні елементи, які забезпечують взаємодію зі сторінкою. Дизайн, стиль і функціональність веб-сторінок можуть різнитися залежно від потреб веб-сайту.

Веб-сайт – це сукупність пов’язаних між собою веб-сторінок, які мають одне доменне ім'я. Веб-сайти складаються з файлів, які містять текст, зображення, відео, аудіо та інші елементи, що відображаються у веб-браузерах користувачів. Разом усі загальнодоступні веб-сайти утворюють Всесвітню павутину.

Принцип роботи веб-сайтів базується на клієнт-серверній моделі. Користувачі використовують браузер для звернення до сервера, який зберігає веб-сайт. Коли користувач вводить URL-адресу, запит надсилається до сервера. Сервер, який є комп'ютером, що зберігає веб-сторінки, сайти або додатки, відповідає на запит. Коли клієнт хоче отримати доступ до веб-сторінки, відбувається копіювання сторінки із сервера на пристрій клієнта, де вона відображається у браузері користувача.

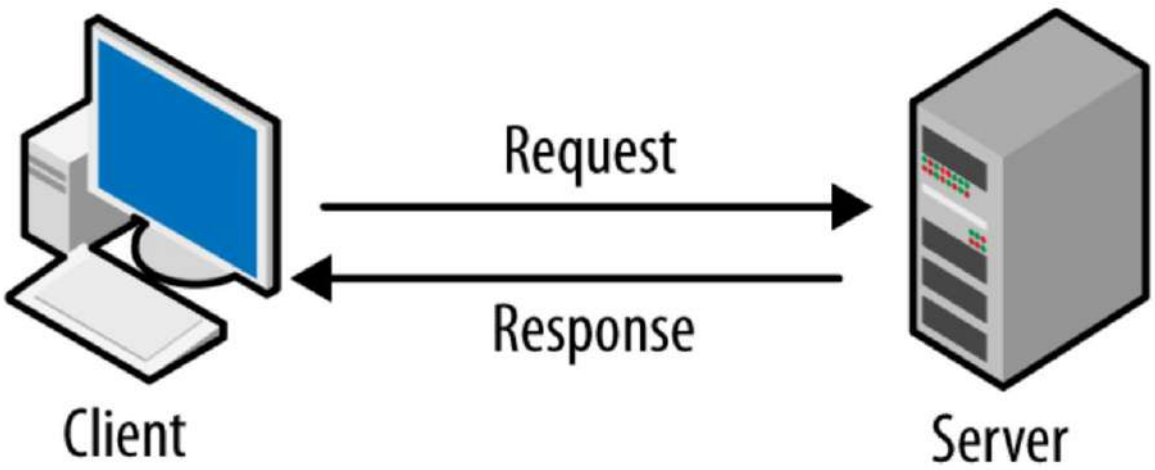


Рисунок 1.1 – Принцип роботи

### 1.3 Інформація про фріланс веб-сайти

Історія фріланс-сайтів починається з виникнення Інтернету та зростанням онлайн-економіки. Спочатку фрілансери шукали роботу через оголошення на форумах і веб-сайтах, але з часом з'явилися спеціалізовані платформи, спрямовані саме на пошук та виконання фріланс-проектів. Розвиток цих сайтів відбувався паралельно з розвитком інтернет-технологій та змінами в робочому стилі людей.

Перші фріланс-сайти, які з'явилися в кінці 1990-х - початку 2000-х років, були спрощеними платформами для публікації проектів та пошуку фрілансерів. Вони дозволяли зареєстрованим користувачам розміщувати оголошення про потребу в послугах та відгуки про роботу. Одним з перших таких сайтів був Elance, який згодом об'єднався з сайтом oDesk, утворивши платформу Upwork.

З часом фріланс-сайти стали більш розвиненими та функціональними. Вони надали користувачам можливість виконувати платежі через платформу, захищаючи обидві сторони від шахрайства. Також з'явилися рейтинги та відгуки, які допомагають оцінити надійність та професіоналізм фрілансерів. [5]

Сьогодні фріланс-сайти стали необхідним інструментом для тисячів людей по всьому світу. Вони дозволяють фрілансерам працювати з будь-якої точки земної кулі і знаходити роботу в різних сферах, від веб-розробки та графічного дизайну до копірайтингу та маркетингу. Клієнти з свого боку мають можливість

знаходити висококваліфікованих фахівців для своїх проектів та ефективно керувати роботою через платформу.

Фріланс-сайти не лише сприяють розвитку глобальної економіки та ринку праці, але й відкривають нові можливості для людей, які шукають гнучкі та цікаві форми заробітку. Ці платформи стали необхідним інструментом для багатьох сучасних професіоналів і підтримують тенденцію до зростання числа фрілансерів у різних галузях. [6]

### 1.4 Існуючі аналоги фріланс веб-сайтів

Upwork - це онлайн-платформа, яка забезпечує можливість знаходити та надавати послуги фрілансу. Вона використовується для різних видів робіт, від копірайтингу та програмування до дизайну та маркетингу. Основна ідея полягає у тому, щоб збирати замовників і фрілансерів з усього світу на одному веб-сайті та упрощувати процес співпраці між ними.

Історія Upwork сягає 2003 року, коли була заснована платформа Elance. Пізніше, у 2013 році, Elance об'єдналася з іншою популярною платформою oDesk, і так виник Upwork, який став однією з найбільших і найпопулярніших майданчиків для фрілансу у світі.

Можливості Upwork включають:

- Пошук роботи або замовників у більшості галузей.
- Створення профілю для представлення своїх навичок та досвіду.
- Участь у тендерах і конкурсах на проекти.
- Організація робочого процесу, включаючи комунікацію, платежі та відстеження часу роботи.

Серед переваг Upwork можна виділити:

- Глобальний доступ до ринку роботи та можливість працювати з клієнтами з усього світу.
- Широкий спектр послуг, що пропонуються на платформі.
- Зручність управління робочим процесом, включаючи платежі та взаємодію з клієнтами.

Недоліки Upwork можуть включати:

- Конкуренція з боку інших фрілансерів, що може ускладнювати пошук роботи.
- Високі комісійні ставки, які стягуються з доходу фрілансерів на платформі.
- Великий обсяг адміністративних процедур, пов'язаних з платформою, наприклад, верифікація акаунту.

Усі ці фактори слід розглядати, обираючи Upwork як майданчик для фрілансу. Ця платформа може бути корисною для тих, хто шукає гнучкість у роботі та можливість співпраці з різними клієнтами.

Далі поговоримо про Freelancer. Це онлайн-платформа для фрілансу, що дозволяє знаходити роботу та надавати послуги в різних галузях, таких як програмування, дизайн, письменництво, маркетинг та інші. Основна мета платформи - з'єднувати замовників і фрілансерів з усього світу для ефективного співробітництва. [7]

Історія Freelancer почалася у 2009 році, коли було засновано платформу з назвою "GetAFreelancer.com". Згодом вона стала відома як Freelancer.com та стала однією з провідних майданчиків для фрілансу в світі.

Можливості Freelancer включають:

- Пошук роботи в широкому спектрі галузей і напрямків.
- Створення профілю з вказанням навичок, досвіду та портфоліо.
- Участь у тендерах і конкурсах на проекти.
- Організація платежів та комунікація з клієнтами через вбудовані інструменти.

Серед переваг Freelancer можна виділити:

- Гнучкість у виборі роботи та робочого графіку.
- Глобальний доступ до ринку роботи та можливість співпраці з клієнтами з різних країн.
- Широкий вибір проектів та можливість розвитку навичок у різних сферах.

Недоліки Freelancer можуть включати:

- Висока конкуренція серед фрілансерів за проекти.
- Платні пакети та комісійні відсотки з кожного успішного проекту.
- Можливість зустрічі з недобросовісними замовниками або фрілансерами.

При виборі Freelancer як платформи для фрілансу, варто уважно оцінювати всі переваги та недоліки, щоб забезпечити успішне та ефективне співробітництво з клієнтами та колегами.

Ще одною відомою платформою є Fiverr. Це онлайн-платформа, яка спрямована на надання послуг у форматі "мікро-робіт" або "гіг". На Fiverr фрілансери пропонують свої послуги за вартість від $5 (відси і назва Fiverr) до значно вищих сум, в залежності від складності та обсягу роботи.

Історія Fiverr сягає 2010 року, коли платформа була запущена як майданчик для невеликих онлайн-послуг і послуг "у форматі $5". Згодом Fiverr розвинувся в одну з найбільших та найпопулярніших платформ для фрілансу.

Можливості Fiverr включають:

- Продаж різноманітних послуг від веб-дизайну та копірайтингу до анімації та музичних композицій.
- Створення профілю з вказанням власних навичок, портфоліо та вартості послуг.
- Участь у промоційних акціях та пропозиціях для залучення клієнтів.
- Розширені можливості спілкування з клієнтами через вбудовані інструменти.

Серед переваг Fiverr можна виділити:

- Широкий асортимент послуг, які можна продавати.
- Можливість отримання замовлень з усього світу.
- Простий інтерфейс та удосконалені функції для забезпечення ефективної роботи.

Недоліки Fiverr можуть включати:

- Висока конкуренція серед фрілансерів за замовлення.

- Платні опції для підвищення видимості профілю або отримання додаткових функцій.
- Можливість стикнутися з недобросовісними клієнтами або фрілансерами.

При використанні Fiverr важливо розглядати всі переваги та недоліки, а також ефективно використовувати стратегії просування та комунікації для успішної роботи на платформі.

## ВИСНОВОК ДО РОЗДІЛУ 1

У цьому розділі розглянуто основні поняття, що стосуються веб-сайтів. Описано історію створення Всесвітньої павутини та першого веб-сайту. Надано інформацію про принцип роботи веб-сайтів та пояснено такі терміни, як веб-сторінка, TCP/IP, DNS та HTTP. Також пояснено принцип роботи клієнт-сервер і представлено відповідну діаграму.

Було наведено історію фріланс сервісів. Фріланс-сайти стали справжнім каталізатором глобальної економіки, дозволяючи фахівцям з різних куточків світу знаходити роботу та розвиватися в своїх сферах. Вони забезпечують безпеку та надійність у взаємодії між сторонами, сприяючи зростанню числа фрілансерів і розширенню можливостей для бізнесу та талантів. Фріланс-сайти стають не тільки платформою для роботи, але й новим способом спілкування та співпраці, що формує сучасний ландшафт професійних відносин.

Були опрацьовані аналоги, цих додатків чи мало, але було виділено і розглянуто найбільш популярні, такі як Upwork, Frelancer та Fiverr ці сайти мають як свої переваги так і недоліки, в свої роботі я хочу вдосконалити вже існуючі аналоги, зробивши веб-сайт, що спросить життя багатьом користувачам, оскільки непотрібно буде шукати, встановлювати або навіть купувати платні додатки.

# РОЗДІЛ 2
# ВИБІР ТЕХНОЛОГІЇ РОЗРОБКИ

## 2.1 Мова програмування C#

Для своєї дипломної роботи я вирішив використовувати C# як основну технологію для розробки програмного забезпечення. Цей вибір обґрунтований кількома важливими факторами. [7]

C# є однією з найпопулярніших мов програмування для розробки веб-додатків, десктопних додатків, мобільних додатків та ігор. Завдяки великій кількості вбудованих бібліотек та фреймворків, таких як ASP.NET для веб-розробки та Xamarin для мобільних додатків, C# забезпечує гнучкість та універсальність у розробці різних типів програмного забезпечення.

C# активно підтримується корпорацією Microsoft, що забезпечує постійне оновлення та вдосконалення мови. Крім того, завдяки великій спільноті розробників, завжди можна знайти підтримку та відповіді на питання, що виникають під час розробки. [8]

Також, C# є об'єктно-орієнтованою мовою програмування, що дозволяє ефективно організовувати код та забезпечувати високу підтримуваність та розширюваність програмного забезпечення. Це особливо важливо для масштабних проектів, де важлива можливість легко додавати нові функції та виправляти помилки.

Крім того, завдяки трирічному досвіду роботи з C#, я добре знайомий з особливостями цієї мови, що дозволяє мені ефективно використовувати її переваги в своєму проекті. Мій досвід роботи з ASP.NET Core, React, Google Drive API та SignalR дає мені впевненість у здатності успішно реалізувати дипломний проект, використовуючи ці технології. [9]

Таким чином, вибір C# як основної технології для моєї дипломної роботи є оптимальним рішенням, що забезпечує високу якість та ефективність розробки програмного забезпечення.

Крім вибору C# як основної мови програмування, важливим аспектом мого проекту є використання .NET платформи. .NET — це потужний фреймворк, розроблений компанією Microsoft, який забезпечує інтегроване середовище для розробки, виконання та розгортання різних типів додатків.

Однією з основних причин вибору .NET є його широка підтримка різних типів додатків, включаючи веб-додатки, мобільні додатки, десктопні додатки та хмарні сервіси. Завдяки таким технологіям, як ASP.NET Core, .NET надає можливості для створення високопродуктивних та масштабованих веб-додатків. У моєму дипломному проекті, який включає розробку фріланс сервісу, ASP.NET Core є ідеальним вибором для створення надійної бекенд-системи.

.NET Core, одна з найновіших реалізацій .NET, є кросплатформенною, що дозволяє розробляти та запускати додатки на різних операційних системах, включаючи Windows, macOS та Linux. Це забезпечує додаткову гнучкість та зменшує залежність від конкретної платформи, що є важливим для сучасних розробників. [10]

Крім того, .NET пропонує багатий набір вбудованих бібліотек та інструментів, що спрощують розробку додатків. Наприклад, Entity Framework Core дозволяє легко працювати з базами даних, а SignalR забезпечує можливість реалізації реального часу в веб-додатках, що є важливим для чат-функціоналу в моєму проекті.

Ще одним вагомим аргументом на користь .NET є висока продуктивність та оптимізація виконання коду. Завдяки JIT-компіляції (Just-In-Time), додатки на .NET виконуються з високою швидкістю, що забезпечує кращу користувацьку взаємодію та задоволення користувачів. [11]

Не менш важливою є наявність великої та активної спільноти розробників, яка підтримує .NET. Це забезпечує легкий доступ до навчальних ресурсів, прикладів коду, а також можливість отримати допомогу від досвідчених колег.

Отже, вибір .NET платформи для моєї дипломної роботи є логічним та обґрунтованим рішенням, що забезпечує високу якість, продуктивність та надійність розробленого програмного забезпечення.

## 2.2 ASP.NET Core

Вибір технології ASP.NET Core для розробки мого дипломного проекту ґрунтується на кількох ключових перевагах, які роблять її оптимальним рішенням для створення сучасних веб-додатків. [12]

Однією з головних переваг ASP.NET Core є його кросплатформеність. На відміну від попередніх версій ASP.NET, ASP.NET Core дозволяє розробляти та запускати додатки на різних операційних системах, включаючи Windows, macOS та Linux. Це забезпечує більшу гнучкість у виборі середовища розробки та розгортання додатків, а також знижує витрати на інфраструктуру.

ASP.NET Core демонструє високу продуктивність завдяки ефективній обробці запитів та використанню асинхронного програмування. Під час розробки веб-додатків важливо забезпечити швидку відповідь сервера на користувацькі запити, особливо при високих навантаженнях. ASP.NET Core досягає цього завдяки оптимізації ядра та використанню Kestrel, високопродуктивного веб-сервера. [13]

ASP.NET Core має модульну архітектуру, що дозволяє підключати тільки необхідні компоненти та бібліотеки. Це зменшує розмір додатка і покращує його продуктивність. Модульна архітектура також спрощує розширення та підтримку додатка, оскільки нові функції можна додавати окремими модулями, не порушуючи основну структуру.

Безпека є критично важливою для будь-якого веб-додатка. ASP.NET Core надає розширені можливості для забезпечення безпеки, включаючи підтримку автентифікації та авторизації, захист від атак типу CSRF (Cross-Site Request Forgery) та XSS (Cross-Site Scripting). Вбудовані засоби захисту та шифрування даних допомагають забезпечити безпеку користувацьких даних та надійність додатка. [14]

ASP.NET Core легко інтегрується з іншими сучасними технологіями та сервісами, що розширює можливості додатка. Наприклад, використання SignalR дозволяє реалізувати функції реального часу, такі як чат, а інтеграція з різними

сторонніми API (наприклад, Google Drive API та Asana API) дозволяє додавати нові функції та сервіси.

Завдяки підтримці хмарних платформ (таких як Microsoft Azure та AWS) та можливості використання контейнеризації (Docker), ASP.NET Core забезпечує легку масштабованість додатків. Це важливо для проектів з потенційно високим навантаженням, оскільки дозволяє швидко розширювати інфраструктуру відповідно до потреб. [15]

ASP.NET Core має велику та активну спільноту розробників, що забезпечує доступ до великої кількості навчальних ресурсів, прикладів коду та готових рішень. Це значно спрощує процес навчання та розробки, а також дозволяє швидко знаходити відповіді на питання та вирішувати проблеми, що виникають під час роботи.

У моєму дипломному проекті, який включає розробку фріланс сервісу, ASP.NET Core буде використовуватися для створення бекенд-системи. Завдяки використанню MVC (Model-View-Controller) архітектури, додаток забезпечить чітке розділення бізнес-логіки, представлення та даних. Це спростить розробку, тестування та підтримку проекту. Крім того, використання SignalR дозволить реалізувати функціонал реального часу для чату, що є важливим для забезпечення миттєвого зв'язку між користувачами платформи. Інтеграція з Google Drive API дозволить користувачам зберігати та обмінюватися файлами безпосередньо через веб-додаток, що підвищить його зручність та функціональність. Таким чином, вибір ASP.NET Core для розробки мого дипломного проекту є оптимальним рішенням, що забезпечує високу якість, надійність та продуктивність створеного фріланс сервісу. [16]

## 2.3 MVC

Однією з ключових технологій, яку я використовую у своєму дипломному проекті, є архітектурний патерн MVC (Model-View-Controller). MVC є широко визнаним підходом до розробки програмного забезпечення, який забезпечує чітке розділення різних аспектів додатка. Використання MVC у моєму проекті

забезпечує декілька важливих переваг, які сприяють створенню ефективного, легко підтримуваного та масштабованого фріланс сервісу. [17]

Що таке MVC? MVC розділяє додаток на три основні компоненти:

1. Model (Модель): Модель відповідає за управління даними додатка. Вона взаємодіє з базою даних та обробляє бізнес-логіку. Модель також може перевіряти вхідні дані та застосовувати необхідні бізнес-правила.
2. View (Представлення): Представлення відповідає за відображення даних користувачеві. Воно отримує дані від контролера і форматує їх для відображення. Представлення не повинно містити логіки бізнес-правил, а лише логіку відображення.
3. Controller (Контролер): Контролер керує взаємодією між моделлю та представленням. Він обробляє вхідні запити від користувачів, взаємодіє з моделлю для отримання необхідних даних і вибирає відповідне представлення для відображення даних. [18]

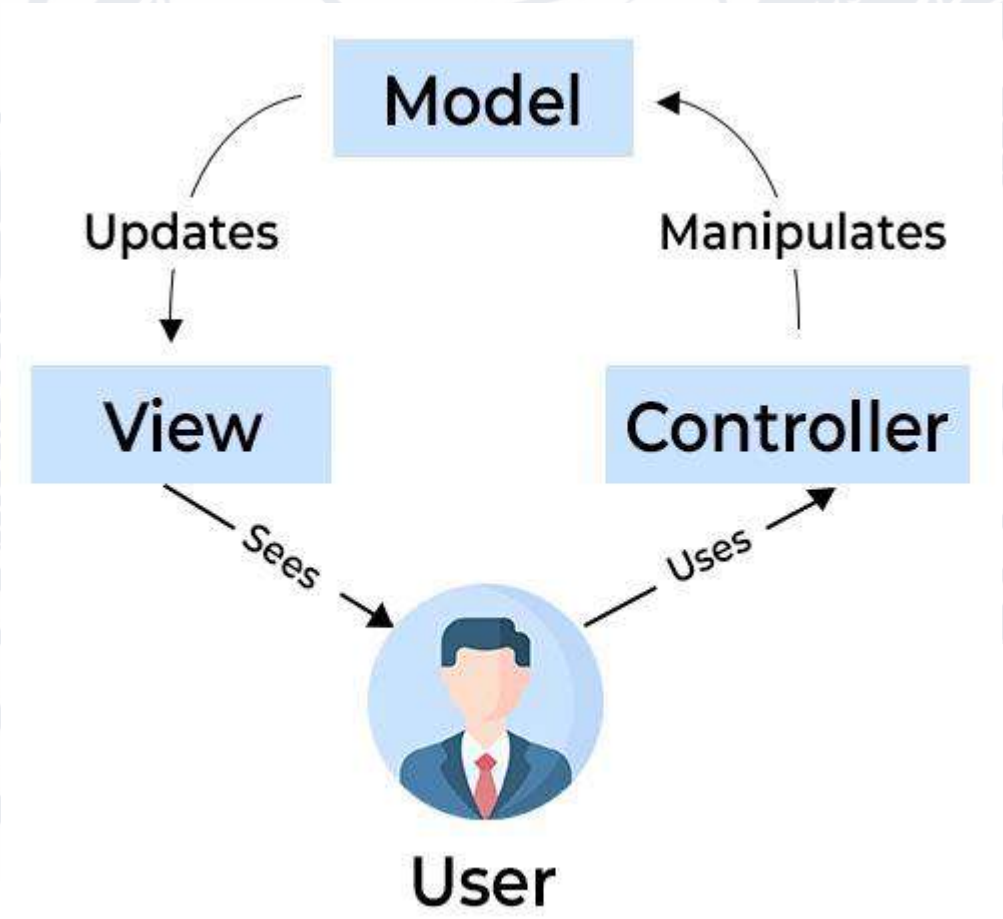


Рисунок 2.1 – Представлення MVC

Завдяки розділенню на модель, представлення та контролер, код стає більш організованим та легким для розуміння. Це спрощує процес розробки, тестування та підтримки додатка, оскільки кожен компонент має чітко визначені обов'язки.

Розділення логіки додатка на окремі компоненти дозволяє легко тестувати кожну частину окремо. Це сприяє розробці більш надійного коду та швидшому виявленню помилок. [19]

MVC забезпечує легкість у розширенні додатка. Додавання нових функцій або зміна існуючих компонентів не впливатиме на інші частини додатка, що дозволяє зберігати стабільність та надійність системи.

Завдяки чіткій структурі MVC, багато компонентів можна використовувати повторно. Наприклад, модель може бути використана для різних представлень, що зменшує дублювання коду та підвищує його ефективність. [20]

MVC сприяє командній роботі, оскільки розробники можуть працювати над різними частинами додатка незалежно один від одного. Наприклад, один розробник може працювати над моделлю, тоді як інший займається розробкою представлення.

У моєму дипломному проекті, який включає розробку фріланс сервісу, MVC використовується для забезпечення структурованого підходу до розробки веб-додатка. Це дозволяє створити надійну та масштабовану систему, яка легко підтримується та розширюється. [21]

## 2.4 Entity Framework Core

Одним із ключових компонентів технологічного стека, який я використовую в своєму дипломному проекті, є Entity Framework Core (EF Core). Це сучасний об’єктно-реляційний засіб відображення (ORM), який спрощує взаємодію між додатком та базою даних. Використання EF Core забезпечує декілька важливих переваг для розробки та підтримки фріланс сервісу. [22]

На наступному малюнку показано, як Entity Framework вписується у додаток.

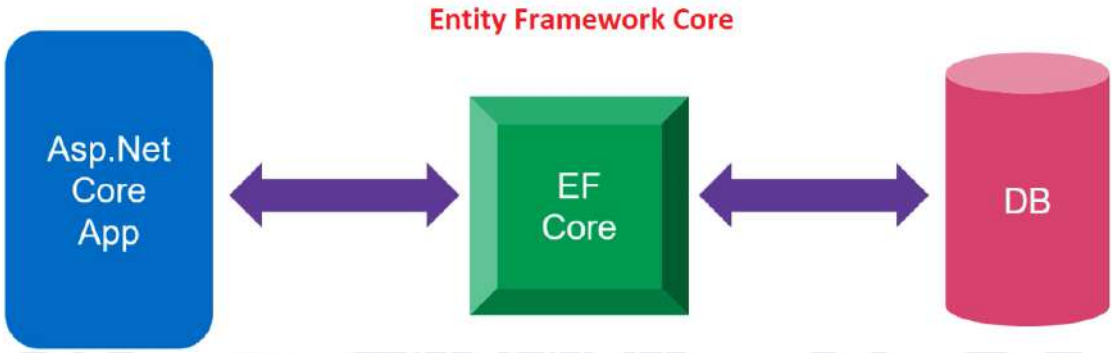


Рисунок 2.2 – Entity Framework у додатку

Entity Framework Core - це ORM для .NET, який дозволяє розробникам працювати з базою даних, використовуючи об'єктно-орієнтовані принципи. EF Core автоматично транслює об'єктно-орієнтовані операції в SQL-запити, що значно спрощує роботу з базою даних та знижує кількість коду, необхідного для управління даними. [23]

EF Core забезпечує високий рівень абстракції, що спрощує роботу з базою даних. Замість написання складних SQL-запитів, розробники можуть використовувати знайомі об'єктно-орієнтовані концепції для взаємодії з даними.

EF Core підтримує роботу з різними типами баз даних, включаючи SQL Server, SQLite, MySQL, PostgreSQL та інші. Це забезпечує гнучкість у виборі бази даних для проекту.

EF Core забезпечує зручний механізм міграцій, який дозволяє автоматично оновлювати структуру бази даних відповідно до змін у моделях. Це спрощує управління версіями бази даних та забезпечує узгодженість даних під час розвитку проекту. [24]

EF Core підтримує LINQ, що дозволяє використовувати потужний синтаксис запитів для отримання даних з бази. LINQ забезпечує інтуїтивно зрозумілий та зручний спосіб написання запитів, що робить код більш читабельним та легким у підтримці.

EF Core підтримує асинхронні операції, що дозволяє збільшити продуктивність додатка, зменшуючи час очікування під час взаємодії з базою даних. Це особливо важливо для веб-додатків, які обробляють велику кількість запитів одночасно.

EF Core автоматично відстежує зміни у даних і забезпечує синхронізацію цих змін з базою даних під час збереження. Це зменшує кількість ручної роботи, необхідної для управління станом даних, та знижує ризик помилок. [25]

У моєму дипломному проекті, який включає розробку фріланс сервісу, EF Core використовується для управління даними користувачів, замовлень, послуг та іншими сутностями. Використання EF Core забезпечує ефективну та зручну роботу з базою даних, спрощує процес розробки та підтримки проекту.

### 2.5 MS SQL Server

У моєму дипломному проекті, який включає розробку фріланс сервісу, я обрав Microsoft SQL Server (MS SQL Server) як основну систему управління базами даних (СУБД). Цей вибір ґрунтується на кількох важливих перевагах, які забезпечують надійну та ефективну роботу з даними. [26]

MS SQL Server - це реляційна СУБД, розроблена компанією Microsoft. Вона забезпечує збереження, управління та обробку даних для різноманітних додатків. MS SQL Server підтримує широкий спектр функцій, включаючи збережені процедури, тригери, транзакції та реплікацію.

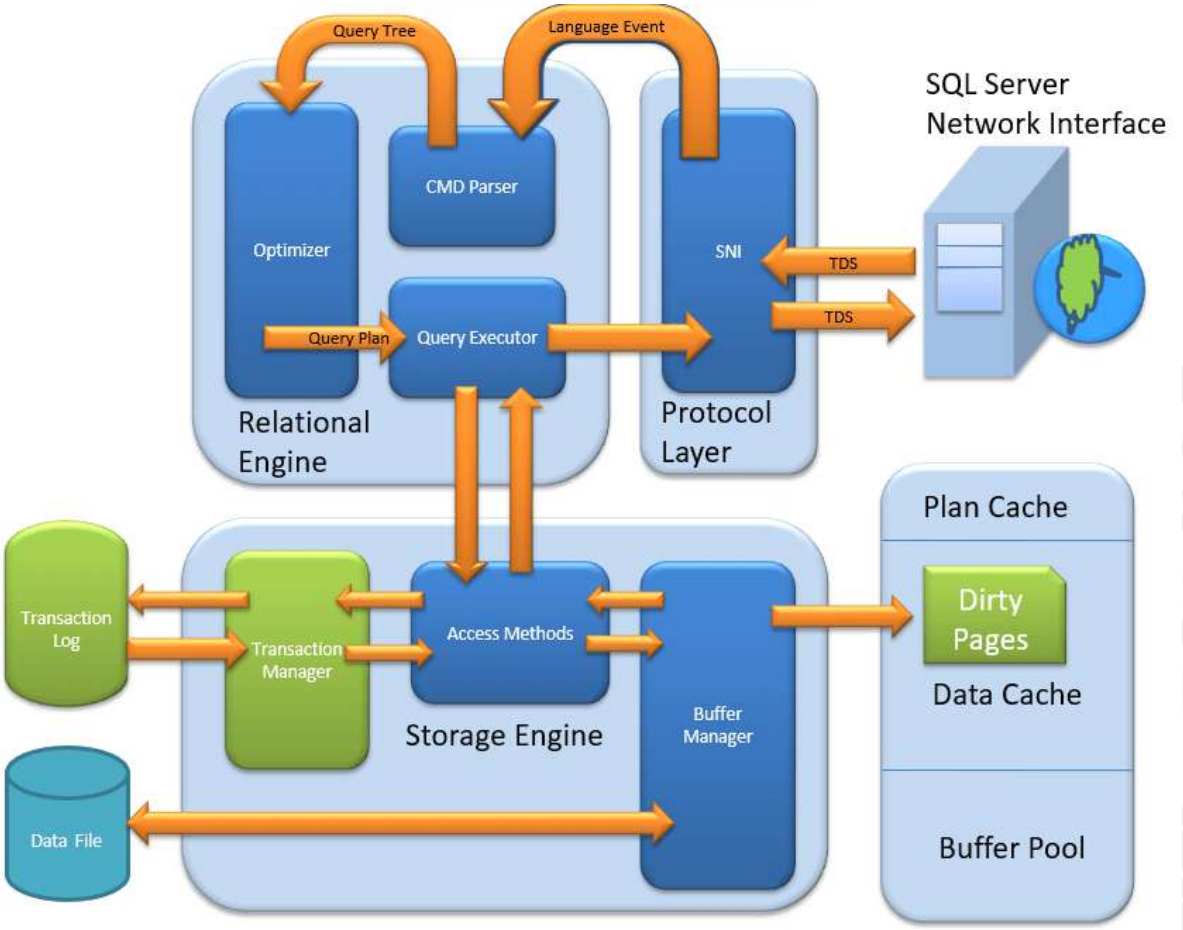


Рисунок 2.3 – Архітектура SQL Server

MS SQL Server відома своєю надійністю та високою продуктивністю. Вона забезпечує швидкий доступ до даних і ефективну обробку великих обсягів інформації, що є критично важливим для масштабованих веб-додатків. [27]

MS SQL Server надає розширені засоби безпеки, включаючи шифрування даних, аудит, контроль доступу на рівні ролей і користувачів. Це забезпечує захист конфіденційної інформації та відповідає сучасним вимогам безпеки.

MS SQL Server легко інтегрується з іншими продуктами та сервісами Microsoft, такими як Azure, Power BI, Microsoft 365 та інші. Це забезпечує додаткові можливості для аналізу даних, звітності та розгортання додатків у хмарі.

MS SQL Server підтримує горизонтальне та вертикальне масштабування, що дозволяє збільшувати ресурси бази даних відповідно до потреб додатка. Це важливо для забезпечення стабільної роботи додатка при збільшенні кількості користувачів та обсягу даних. [28]

MS SQL Server надає потужні інструменти для розробки та адміністрування баз даних, такі як SQL Server Management Studio (SSMS) та Visual Studio. Ці інструменти спрощують процес створення, налагодження та підтримки баз даних.

MS SQL Server забезпечує повну підтримку транзакцій, що дозволяє зберігати цілісність даних навіть у випадку збоїв. Транзакції забезпечують атомарність, консистентність, ізоляцію та довговічність (ACID).

У моєму дипломному проекті, який включає розробку фріланс сервісу, MS SQL Server використовується для зберігання та управління даними користувачів, замовлень, послуг та іншими сутностями. Використання MS SQL Server забезпечує надійну та ефективну роботу з даними, спрощує процес розробки та підтримки проекту. [29]

## 2.6 JWT Token

JWT (JSON Web Token) є стандартом для створення токенів доступу в мережі, які можна використовувати для автентифікації та авторизації користувачів в веб-додатках. Основні переваги JWT полягають у його простоті, безпеці та легкості використання. Цей механізм забезпечує передачу даних між клієнтом і сервером у форматі JSON, які можна перевірити та підтвердити за допомогою ключа.

JWT Token складається з трьох основних частин: заголовок (header), тіло (payload) та підпис (signature), які розділяються крапками (.):

Заголовок містить метадані про тип токену та алгоритм, використовуваний для його підпису. [30]

Тіло містить корисну інформацію, таку як ідентифікатор користувача, термін дії токену та інші важливі дані.

Підпис використовується для перевірки цілісності токену та підтвердження, що він був створений сервером, який має право його створювати. Підпис генерується на основі заголовка та тіла токену за допомогою певного алгоритму, наприклад, HMAC-SHA256.

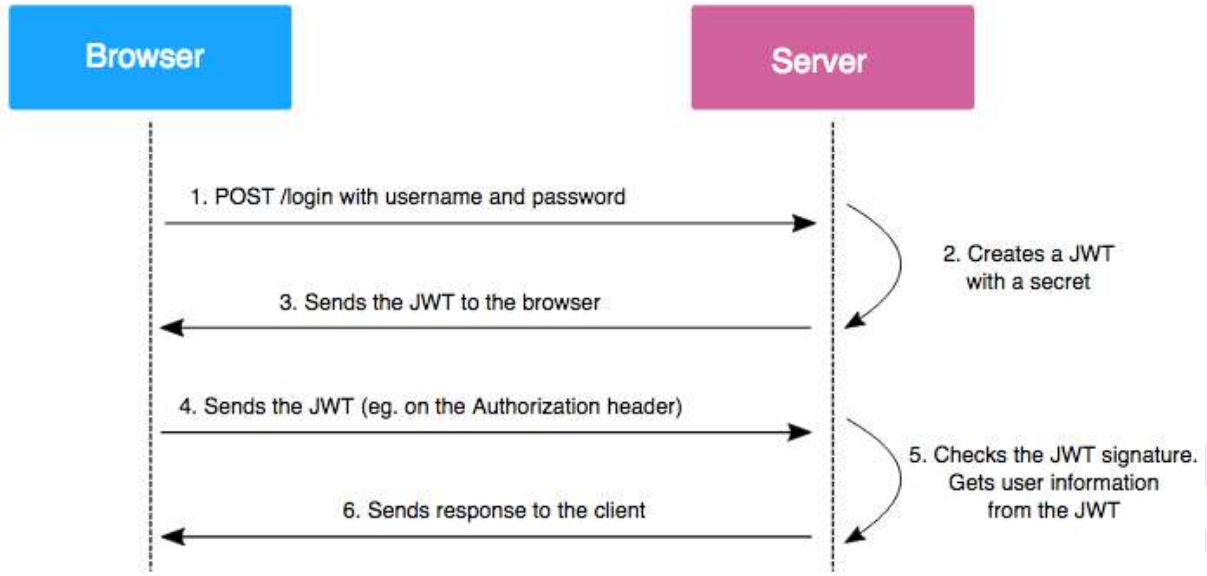


Рисунок 2.4 – Принцип роботи токену

Коли користувач успішно аутентифікується на сервері (наприклад, вводячи логін та пароль), сервер генерує JWT Token і відправляє його клієнту. Клієнт зберігає токен (наприклад, у локальному сховищі), із якого він потім передається у заголовку кожного запиту до сервера.

При отриманні запиту від клієнта з JWT Token, сервер перевіряє токен, дешифрує та перевіряє підпис. Якщо токен валідний і не закінчився термін дії, сервер визначає, які дії користувач має право виконувати, основуючись на інформації в тілі токену.

JWT Token захищає дані, що передаються між клієнтом і сервером, оскільки вони шифруються і підписуються. Це дозволяє уникнути маніпуляцій або підробки даних під час передачі. [31]

JWT Token дозволяє підтримувати автентифікацію та авторизацію без необхідності зберігання стану сесії на сервері, що сприяє масштабуванню додатка.

Отже, JWT Token - це потужний і популярний механізм для забезпечення безпеки та автентифікації в веб-додатках. Він дозволяє ефективно передавати та перевіряти дані між клієнтом і сервером, забезпечуючи захист даних та простоту у використанні.

## 2.7 React

React - це бібліотека для створення користувацьких інтерфейсів (UI) веб-додатків. Вона розроблена компанією Facebook і використовується для побудови відзивчивих, масштабованих та ефективних інтерфейсів. Основні переваги React полягають у його компонентній архітектурі, відзивчивості та можливостях перевикористання коду.

React базується на концепції компонентів, які дозволяють розділити інтерфейс на невеликі незалежні частини. Компоненти можуть бути класовими (Class Components) або функціональними (Functional Components) і містять логіку та представлення для відображення даних на екрані.

React дозволяє створювати відзивчиві інтерфейси, оновлюючи тільки ті частини сторінки, які змінилися (за допомогою Virtual DOM). Це забезпечує швидку відповідь на дії користувача та ефективне використання ресурсів браузера.

React дозволяє створювати односторінкові додатки, де весь інтерфейс завантажується один раз, а подальші переходи між сторінками відбуваються без перезавантаження сторінки. [32]

Virtual DOM - це концепція, яка дозволяє React ефективно відображати зміни на екрані. React створює віртуальне дерево DOM, яке порівнює з реальним DOM та оновлює тільки ті елементи, які змінилися, що забезпечує швидку відображення змін.

JSX - це розширення JavaScript, яке дозволяє писати код React у вигляді XML-подібних структур. Це зробило React більш зрозумілим та зручним для розробників, оскільки код стає більш декларативним.

React дозволяє розділити інтерфейс на малий набір незалежних компонентів, які можна перевикористовувати та легко підтримувати.

React дозволяє створювати швидкі та відзивчиві інтерфейси за рахунок використання Virtual DOM та ефективного оновлення частин сторінки.

## 2.8 Axios

Axios - це бібліотека для виконання HTTP-запитів з JavaScript, яка дозволяє зручно взаємодіяти з веб-серверами та отримувати або відправляти дані через мережу. Вона широко використовується у фронтенд-розробці, зокрема у веб-додатках, що використовують React, Vue.js та інші сучасні фреймворки.

Простота використання: Axios має зрозуміле та легко зроблене API, що дозволяє легко виконувати HTTP-запити, встановлювати заголовки, передавати параметри та отримувати відповіді.

Підтримка Promise: Axios повертає об'єкт Promise, що дозволяє використовувати асинхронний код і обробляти результати запиту зручним способом. [33]

Автоматична серіалізація та десеріалізація JSON: Axios автоматично серіалізує дані в формат JSON при відправці запитів та десеріалізує JSON-відповіді у JavaScript-об'єкти.

Підтримка перехоплення помилок: Axios дозволяє легко обробляти помилки, які виникають під час виконання запитів, включаючи помилки мережі, статуси HTTP-відповідей тощо.

Можливість встановлення заголовків: Axios дозволяє встановлювати заголовки для кожного запиту, що дозволяє контролювати параметри запиту, такі як Content-Type, Authorization тощо. [34]

## 2.9 HTML & CSS

HTML (HyperText Markup Language) і CSS (Cascading Style Sheets) - це основні мови для розробки веб-сайтів і створення їх зовнішнього вигляду. HTML відповідає за структуру і вміст веб-сторінок, а CSS - за їх стиль та візуальне оформлення. Основні концепції цих мов допомагають розуміти їх функціональність та використання в розробці веб-інтерфейсів.

HTML - це мова розмітки, яка використовується для створення структури веб-сторінок. Вона складається з тегів (tags), які вказують браузеру, як відображати вміст сторінки. [35]

CSS - це мова стилів, яка використовується для задання зовнішнього вигляду елементів HTML. Вона дозволяє встановлювати кольори, шрифти, розміри, відступи та інші стилізації.

HTML визначає структуру сторінки, а CSS - її стиль і вигляд, що дозволяє розділити логіку та представлення.

Використання CSS дозволяє створювати відзивчиві дизайни, які пристосовуються до різних розмірів екранів.

Можливість використовувати класи та ідентифікатори дозволяє перевикористовувати стилі для різних елементів.

HTML і CSS - це прості мови, які легко вивчити та використовувати для створення веб-сторінок і інтерфейсів. [36]

## 2.10 Stripe

Stripe є однією з найпопулярніших платформ для обробки онлайн-платежів, що пропонує широкий спектр API для інтеграції платіжних можливостей у веб-додатки. У контексті фріланс сервісу інтеграція зі Stripe дозволяє забезпечити зручну та безпечну оплату за проекти, обробку транзакцій та виплати виконавцям. Цей розділ детально описує кроки інтеграції Stripe з використанням ASP.NET Core для бекенду та React для фронтенду, а також конкретні приклади реалізації. [37]

Для кожного клієнта створюється окремий об’єкт Customer, за допомогою якого замовник може зберігати свої методи оплати для подальшого використання. Коли замовник і виконавець домовилися про всі деталі справи, замовник може оплатити угоду через Stripe Checkout Session, де зберігаються його попередні методи оплати.

Для кожного виконавця створюється Connected Account у Stripe. Спочатку цей аккаунт стає неактивним поки виконавець не пройде верифікацію, яка складається з декількох етапів. Також поки аккаунт не верифікований він не може брати участь у будь-яких угодах. Верифікувати аккаунт виконавець може на своїй сторінці. Після верифікації він може переглянути свою сторінку транзакції та

баланс як статистику. Усі гроші надходять на Connected Stripe Account, та раз у день всі гроші виводяться на його карту якою він може також керувати.

### 2.11 Google Drive API

Google Drive API використовується для завантаження та зберігання фотографій користувачів на клауд диску. Цей сервіс дозволяє кожному виконавцю додавати фотографії до свого профілю, забезпечуючи зручний спосіб управління медіафайлами. Всі фотографії зберігаються в окремій папці, призначеній для даного користувача. [38]

Для доступу до диску Google Drive використовується ClientId, ClientSecret та Refresh Token. З їх допомогою отримується Access Token, який надає доступ до ресурсів Google Drive. Це забезпечує безпечне та надійне завантаження файлів.

Після автентифікації користувачів, система дозволяє завантажувати фотографії у відповідні папки. Це спрощує управління медіаконтентом і дозволяє зберігати файли централізовано та безпечно.

### 2.12 SignalR

SignalR використовується для реалізації реального часу спілкування у чаті між користувачами платформи. Це забезпечує миттєвий обмін повідомленнями та підвищує зручність та ефективність взаємодії між фрілансерами та замовниками.

SignalR використовує WebSocket або інші протоколи для встановлення постійного з'єднання між клієнтами та сервером. Це дозволяє відправляти та отримувати повідомлення в режимі реального часу без необхідності оновлювати сторінку. [39]

Реалізація чату за допомогою SignalR дозволяє користувачам обмінюватись повідомленнями миттєво. Це значно покращує користувацький досвід та сприяє швидшому та ефективнішому спілкуванню.

## ВИСНОВОК ДО РОЗДІЛУ 2

У даному розділі розглянуто технології, що використовуються у практичній частині цієї роботи. Детально розглянута мова програмування C# та платформа .Net, зокрема її можливості та приклади інших схожих мов програмування.

Продемонстровано фреймворк для back end — ASP.Net Core, описано його можливості та порівняно з ASP.Net Framework. Обрано ASP.Net Core через його кросплатформенність, що дозволяє створювати застосунки для різних операційних систем.

Розглянуто принцип роботи патерна MVC та наведено ілюстрацію з поясненнями. Надано інформацію про Entity Framework, його можливості та порівняно EF 6 та EF Core.

Також описано базу даних MS SQL Server, порівняно її з іншими БД та виділено переваги обраної. Наведено архітектурну схему.

Детально розглянуто JSON Web Token, його принцип роботи та призначення, з наведенням схеми роботи. Описано бібліотеку React та причини її популярності.

# РОЗДІЛ 3
# ДЕТАЛІ РОЗРОБКИ СИСТЕМИ

## 3.1 Архітектура проекта - Backend частина

В цьому розділі буде детально описано архітектуру розробленої системи фріланс сервісу, що ґрунтується на сучасних технологіях ASP.NET Core та React. Основна мета даної архітектури полягає в забезпеченні ефективної взаємодії між клієнтською та серверною частинами, а також у розподілі відповідальності між різними компонентами системи для покращення її масштабованості та підтримуваності. [40]

Архітектура проекту структурована таким чином, що серверна частина (backend) складається з трьох основних шарів:

1. Data Access Layer (Шар доступу до даних): Цей шар забезпечує взаємодію з базою даних. Його головна задача полягає в реалізації механізмів збереження, читання, оновлення та видалення даних. Використовуючи Entity Framework Core ORM (Object-Relational Mapping), шар доступу до даних забезпечує зручність роботи з базою даних, абстрагуючись від деталей реалізації бізнес-логіки. Даний шар складається з наступних компонентів:

- Models: папка, що містить моделі. Кожна модель репрезентує таблицю в базі даних та мапиться через ORM систему.
- Enums: папка, що містить перерахування для моделей.
- Migrations: папка, що містить міграції які утворюють таблиці та структуру бази даних за принципом Code-First.
- Repositories: папка, що містить патерн репозиторії для доступу до таблиць у базі даних та маніпулювання цими даними.
- AppDbContext: клас, що містить конфігурацію та контекст бази даних.

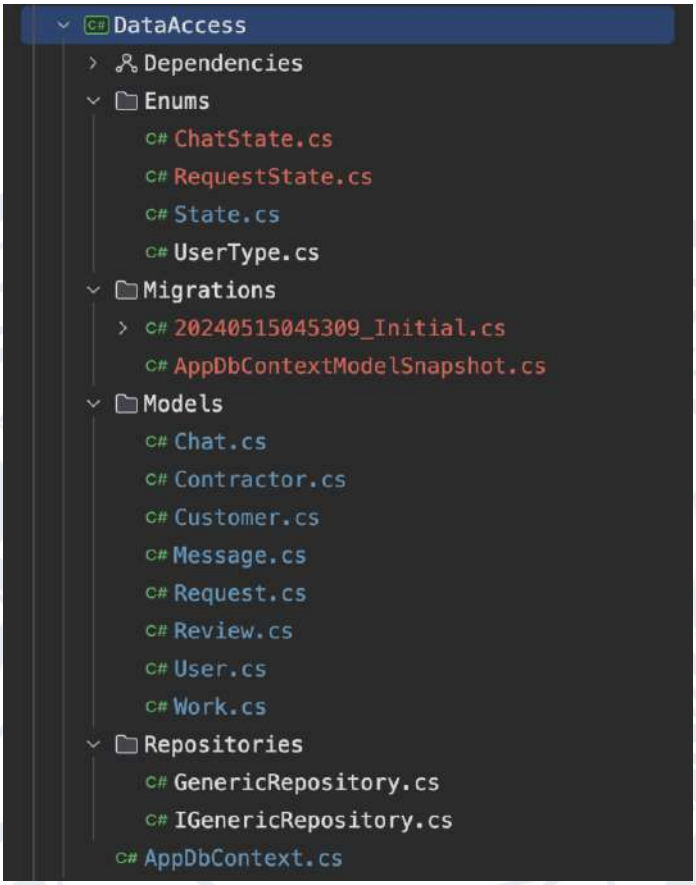


Рисунок 3.1 - Структура Data Access проекту.

2. Business Logic Layer (Шар бізнес-логіки): Цей шар відповідає за обробку основних логічних операцій системи, таких як обробка замовлень, управління користувачами, обробка платежів та інші важливі функції. Він є ключовим компонентом, що реалізує правила і процеси бізнесу, забезпечуючи коректність і цілісність даних та логіки роботи системи. Для роботи платіжної системи було використано платформу Stripe, для роботи із зображеннями та файлами було використано Google Drive API, також для real-time спілкування було використано бібліотеку SignalR, яка працює на основі WebSockets. Більш детально по кожній інтеграції ми пройшлись у попередніх розділах. Даний шар складається з наступних компонентів:

- DTOs: папка, що містить DTO (Data Transfer Object), об’єкти які використовуються для передачі даних між шарами Presentation Layer до Data Access Layer.
- Hubs: папка, що містить хаб, який обробляє запити SignalR від React’у, обробляє запит та з’єднує його з DAL.

- Interfaces: папка, що містить інтерфейси різних сервісів які потрібні для обрахунку бізнес логіки.
- Service: папка, що містить реалізацію інтерфейсів та використовує Third-Parties, такі як Stripe, Drive, Entity Framework Identity Core.
- WorkModel: папка, що містить реалізацію патерну UOW (Unit Of Work), яка допомагає бізнес лозіці створювати запити до бази даних з використанням LINQ.

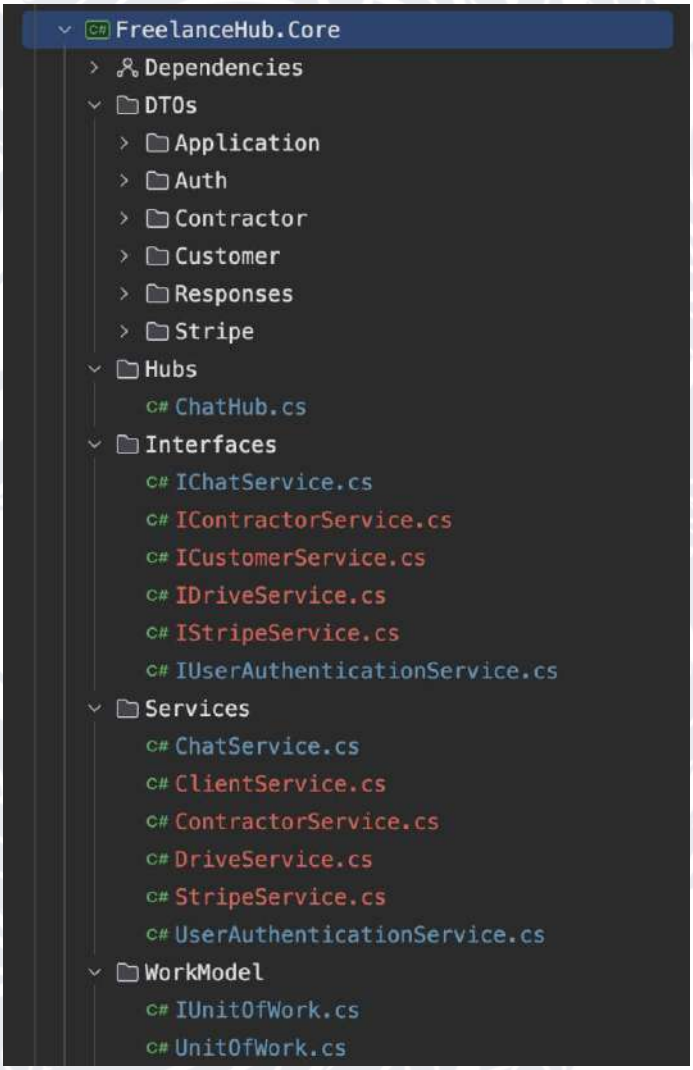


Рисунок 3.2 - Структура Business Logic проекту.

3. Presentation Layer (Шар презентації): Цей шар відповідає за взаємодію з користувачами системи, надаючи їм зручний інтерфейс для виконання необхідних дій. Використання React на фронтенді дозволяє створювати динамічні та реактивні веб-додатки, що забезпечують високу швидкодію та зручність у користуванні. Даний шар складається з наступних компонентів:

- Program.cs: клас, вхідна точка до проекту, тут конфігурується веб-додаток.
- Extensions: папка, що містить розширення для зручного додавання сервісів через DI (Dependence Injection).
- appsetings.json: файл, що містить конфігурації та ключі доступу для деяких сервісів.
- Controllers: папка, що містить контролери, які є ендпоїнтами для спілкування між фронт та бек частиною веб-додатку. Ця папка містить в собі контролери для аунтентифікації, для чату, для замовника, для виконавця, для роботи з файлами та зображеннями, та контролер який проймає оновлення від Stripe за технологією Webhooks.
- ClientApp: є папками для фронт частини проекту яку ми розглянемо у наступному розділі.

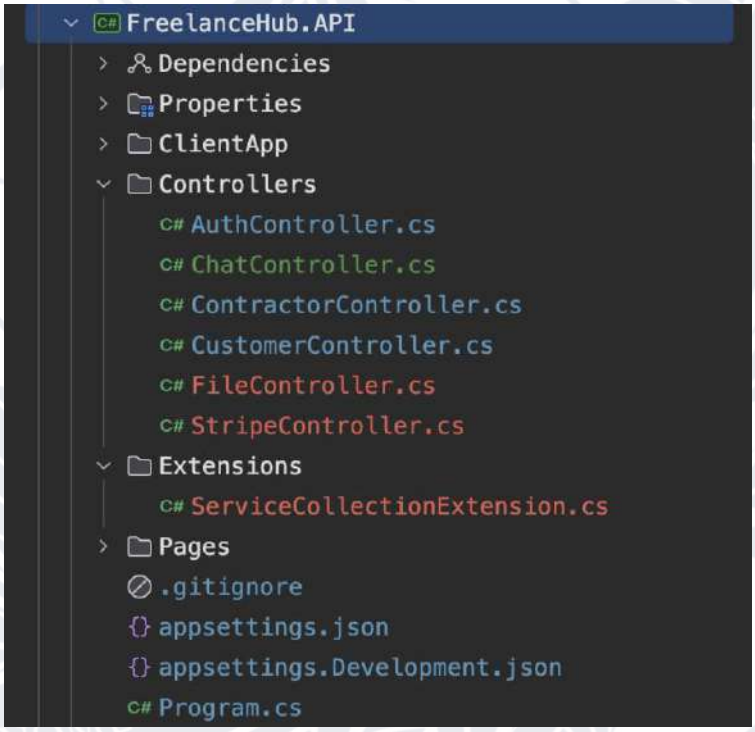


Рисунок 3.3 - Структура Presentation Layer проекту.

Загалом, така багатошарова архітектура дозволяє досягти високої продуктивності, гнучкості та підтримуваності системи, що є надзвичайно важливим для успішного функціонування фріланс сервісу. У наступних підрозділах буде докладніше розглянуто кожен з цих шарів, а також специфічні аспекти їх реалізації та взаємодії.

Структура бази даних (ER діаграма):

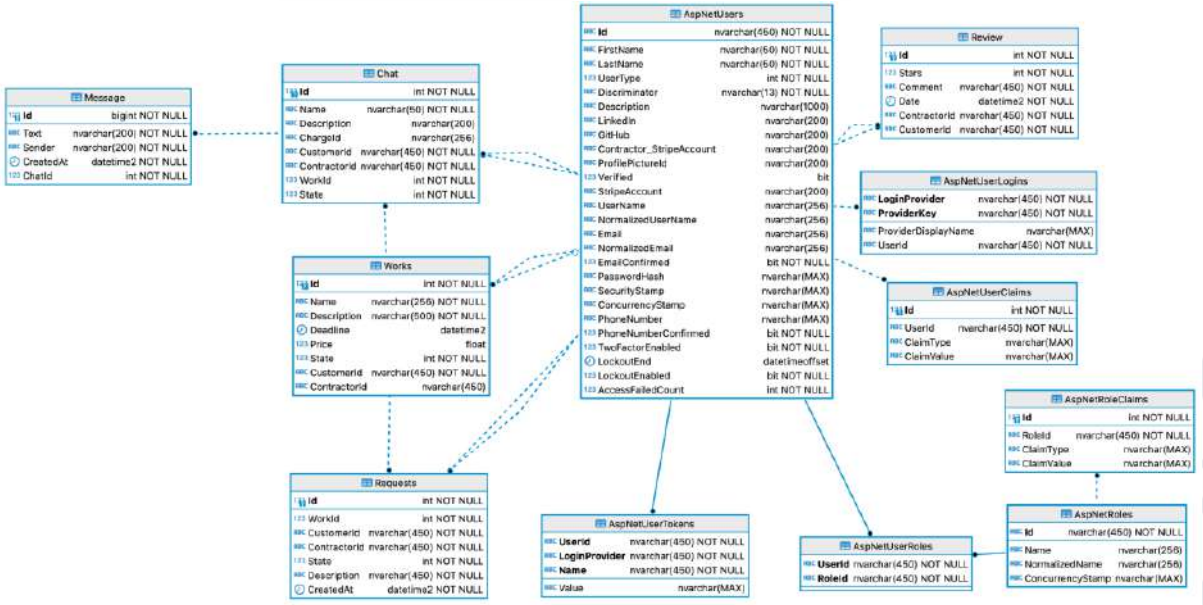


Рисунок 3.4 - ER діаграма бази даних.

## 3.2 Архітектура проекта - Frontend частина

Фронтенд частина проекту створена за допомогою React і організована у вигляді папки ClientApp, що містить різні компоненти і налаштування. Далі наведено опис кожного файлу та папки, що входять до складу проекту.

Папки та файли:

- node_modules - папка, яка містить всі встановлені пакети та залежності для проекту. Вона автоматично створюється при встановленні залежностей за допомогою npm або yarn.
- public - папка, що містить статичні файли, які не проходять через процес зборки Webpack. Це можуть бути HTML файли, зображення, шрифти та інші статичні ресурси.
- src - основна папка, де знаходиться весь вихідний код додатку. “api" - тут знаходяться файли, які відповідають за взаємодію з серверною частиною, зокрема за виконання HTTP-запитів до backend. “components” - папка містить окремі компоненти React, які використовуються для створення інтерфейсу користувача. Кожен компонент може включати власні файли CSS, тестові файли та інші залежності. “pages” - папка містить компоненти,

що відповідають за окремі сторінки додатку. Кожен файл у цій папці представляє собою окрему сторінку або маршрут.

- App.js - головний компонент додатку, який є точкою входу в додаток. Він визначає основну структуру додатку та включає в себе інші компоненти.
- AppRoutes.js - файл, що відповідає за визначення маршрутів у додатку. Він містить налаштування маршрутизації, які визначають, які компоненти повинні бути відображені для кожного маршруту.
- custom.css - файл стилів, що містить кастомні CSS правила, які застосовуються до всього додатку.
- index.js - основний файл входу JavaScript, який рендерить компонент App у DOM. Це точка входу для Webpack і React.
- setupProxy.js - файл налаштувань проксі, який використовується для перенаправлення запитів API до backend серверу під час розробки. Це дозволяє обійти проблеми з CORS.
- aspnetcore-https.js - файл налаштувань для використання HTTPS в ASP.NET Core додатках. Він містить налаштування, які забезпечують безпечне з'єднання.
- aspnetcore-react.js - файл налаштувань для інтеграції ASP.NET Core з React. Він містить налаштування, які забезпечують коректну роботу обох технологій разом.

Це базовий опис архітектури фронтенд частини проекту. Далі можна перейти до детального опису кожної папки та файлів, що в них знаходяться.

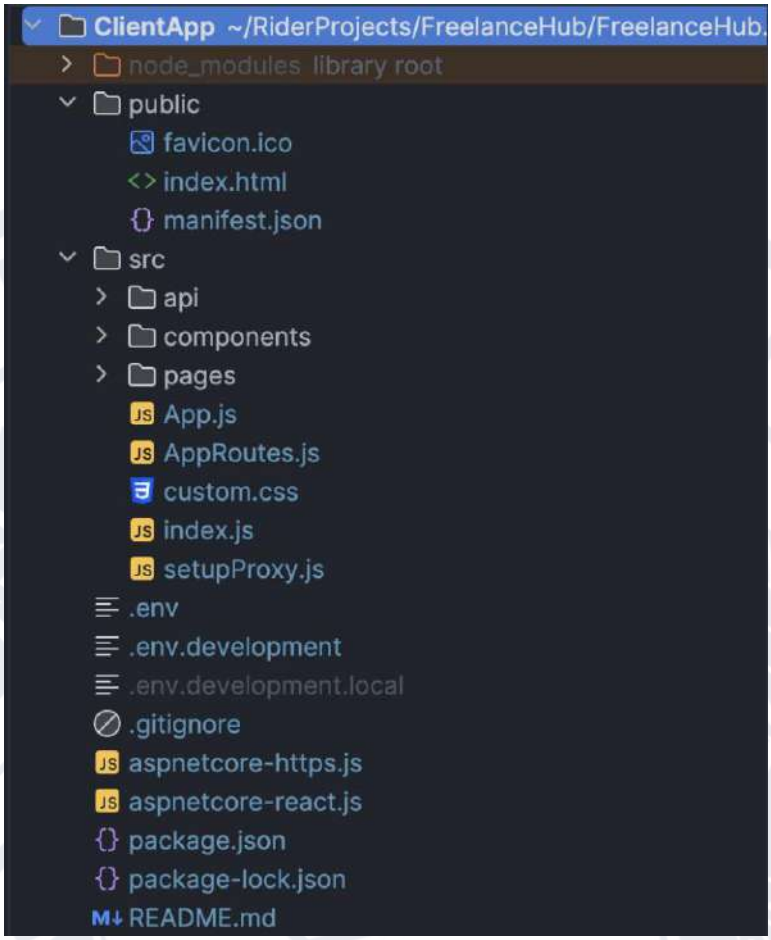


Рисунок 3.5 - Загальна структура фронтенд частини

Папка src:

“api”:

- index.js: Основний файл для управління запитами до API. Він містить налаштування для виконання HTTP-запитів до серверної частини за допомогою **axios**.
- signalR.js: Файл для налаштування і використання бібліотеки SignalR, що забезпечує реально-часову комунікацію між клієнтом і сервером.

“components/layout”:

- AppFooter.jsx: Компонент, що відповідає за відображення нижнього колонтитула (footer) додатку.
- AppNavbar.jsx: Компонент, що відповідає за відображення навігаційної панелі (navbar) додатку.

- Layout.jsx: Основний компонент макету, який визначає загальну структуру сторінок додатку, включаючи header, footer та основний контент.

“pages”:

1. auth:
   - Login.jsx: Компонент сторінки входу до системи. Містить форму для введення облікових даних користувача (логін та пароль).
   - Register.jsx: Компонент сторінки реєстрації нового користувача. Містить форму для введення реєстраційних даних.
2. chats:
   - Chats.jsx: Компонент для відображення та управління чатами. Забезпечує функціональність обміну повідомленнями між користувачами.
3. forms:
   - AddWorkForm.jsx: Форма для додавання нової роботи або проекту. Використовується користувачами для створення нових завдань.
   - CommentPopup.jsx: Спливаюче вікно для додавання коментарів. Використовується для введення і відображення коментарів до завдань або проектів.
4. home:
   - ClientPage.jsx: Сторінка клієнта, що містить інформацію та функціональність, доступну клієнтам сервісу.
   - ContractorPage.jsx: Сторінка підрядника (виконавця), що містить інформацію та функціональність, доступну підрядникам сервісу.
   - Home.jsx: Головна сторінка додатку. Відображає загальну інформацію та основну навігацію.
   - ReviewPage.jsx: Сторінка для перегляду відгуків про підрядників або проекти. Містить інформацію про рейтинги та відгуки.
5. requests:

- RequestModal.jsx: Модальне вікно для створення або редагування запитів. Використовується для взаємодії з користувачем у контексті запитів.
- RequestPage.jsx: Сторінка для перегляду і управління запитами. Відображає список запитів та їх деталі.

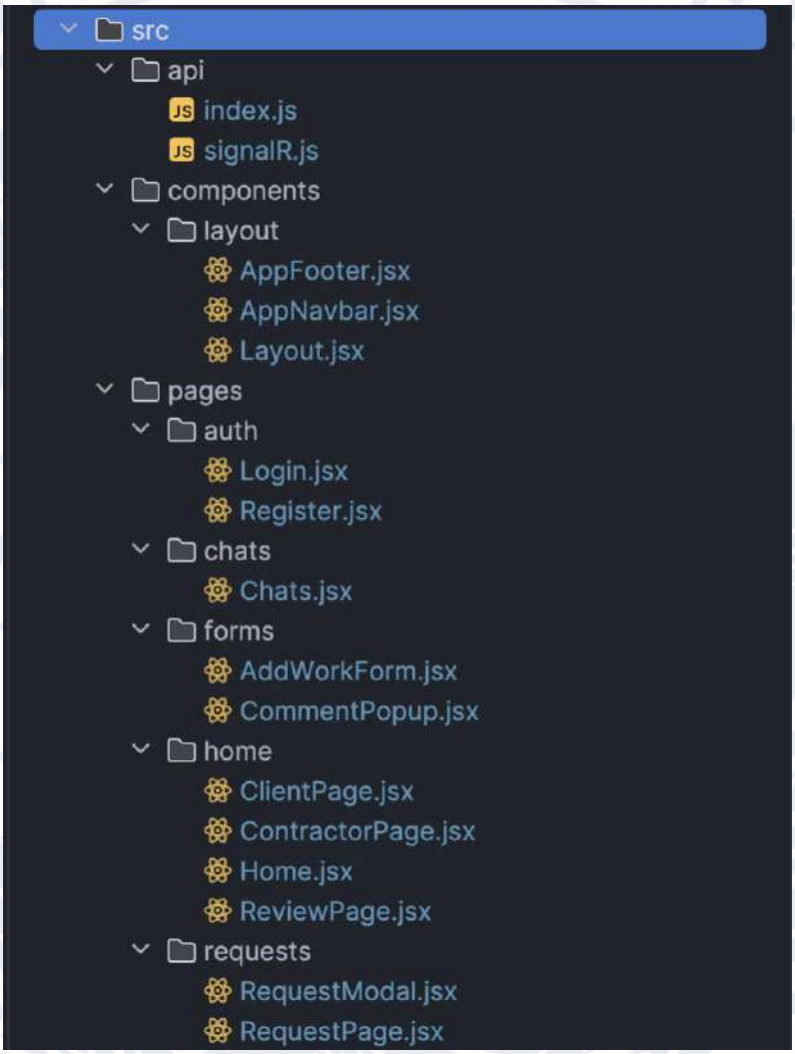


Рисунок 3.6 - Структура та компоненти src папки

### 3.3 Робота проекту

Нижче розписаний алгоритм роботи проекту, та всі функції вебсайту. Розглянемо функції які доступні для замовника та виконавця, алгоритм проведення угоди та як взаємодіють всі інтеграційні системи разом.

Почнемо з реєстрації та входу в аккаунт. Основна сторінка реєстрації є першою що бачить користувач при заході на сайт (рис. 3.7).

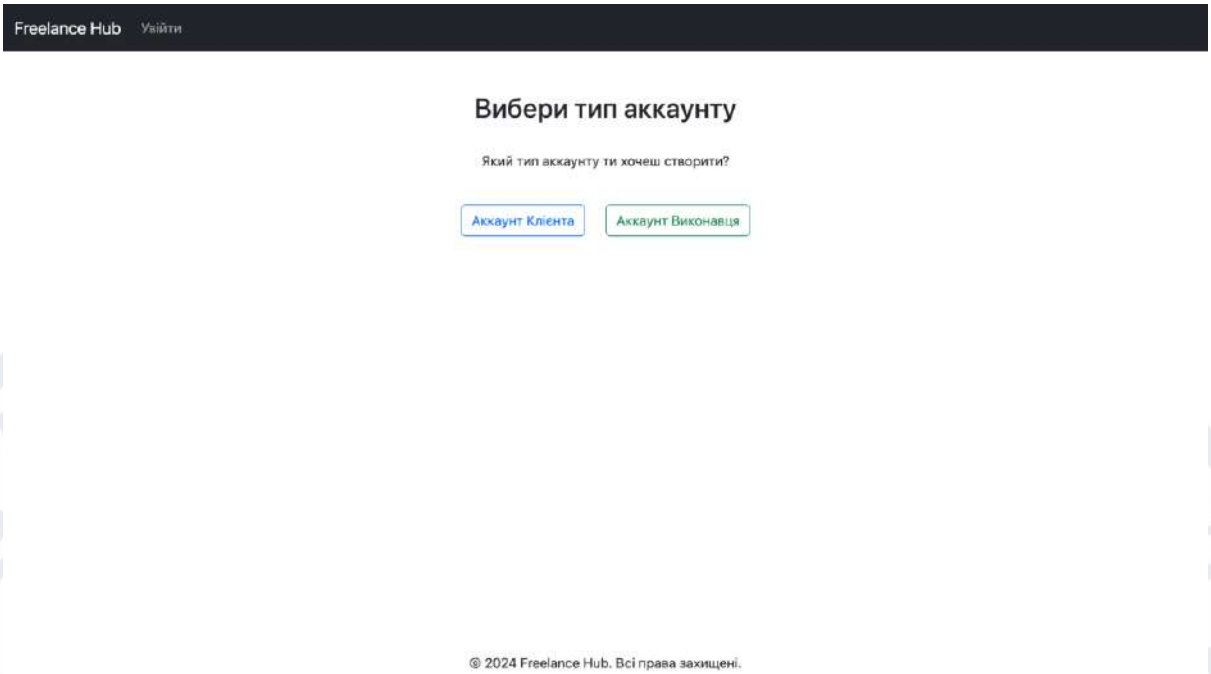


Рисунок 3.7 - Сторінка вибору аккаунту.

При виборі типу аккаунту ми маємо однакову форму як для замовника так і для виконавця з типовими полями (рис. 3.8).

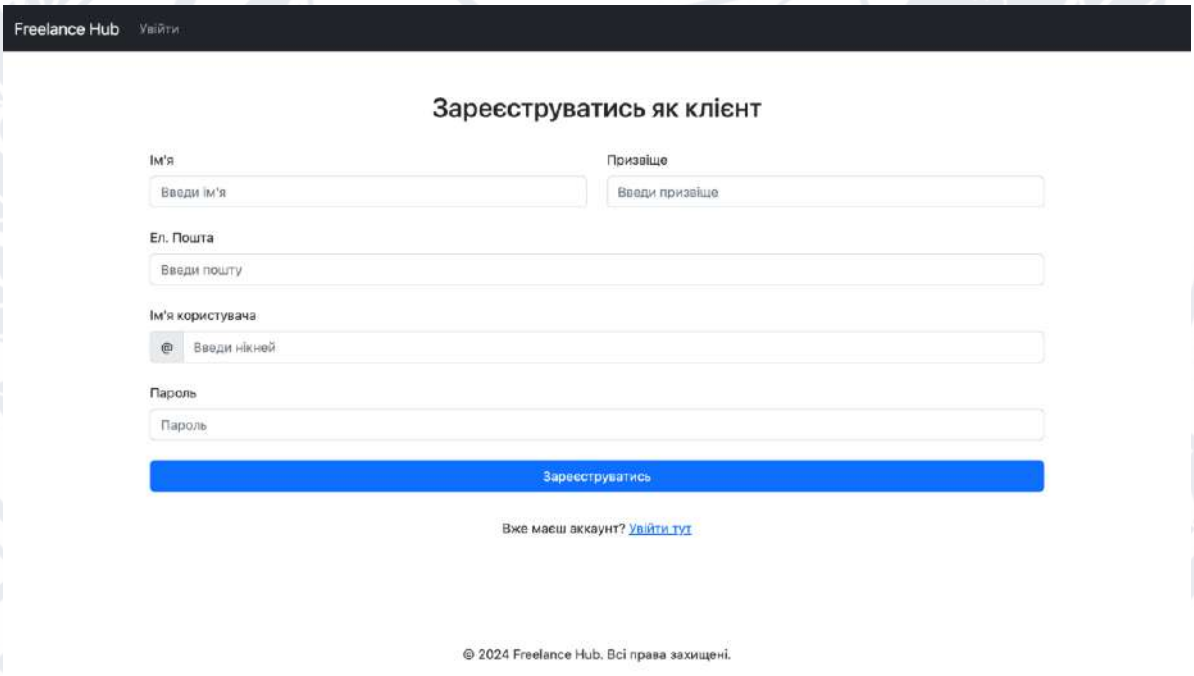


Рисунок 3.8 - Форма реєстрації.

Після реєстрації також можливо увійти в аккаунт через форму входу (рис. 3.9).

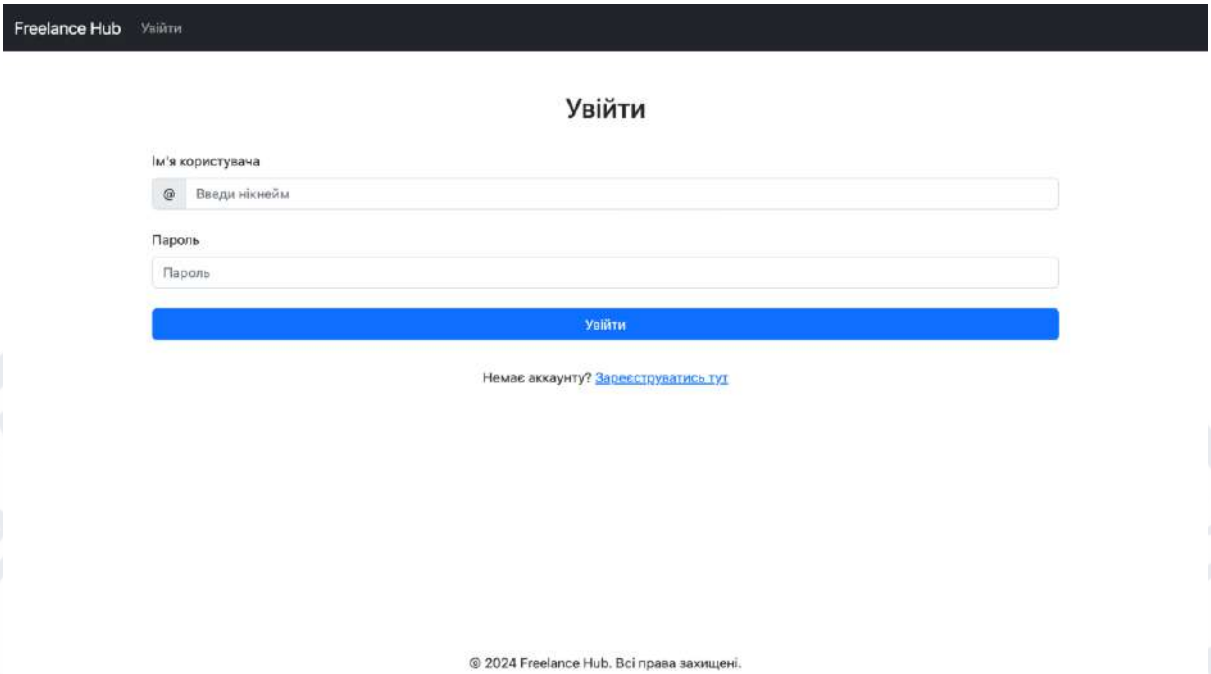


Рисунок 3.9 - форма входу в акаунт.

Далі поговоримо про функції замовника. Нас зустрічає головна сторінка яка містить інформацію про опубліковані завдання замовника, вони діляться на категорії: Активні, В процесі, та Завершені (рис. 3.10).

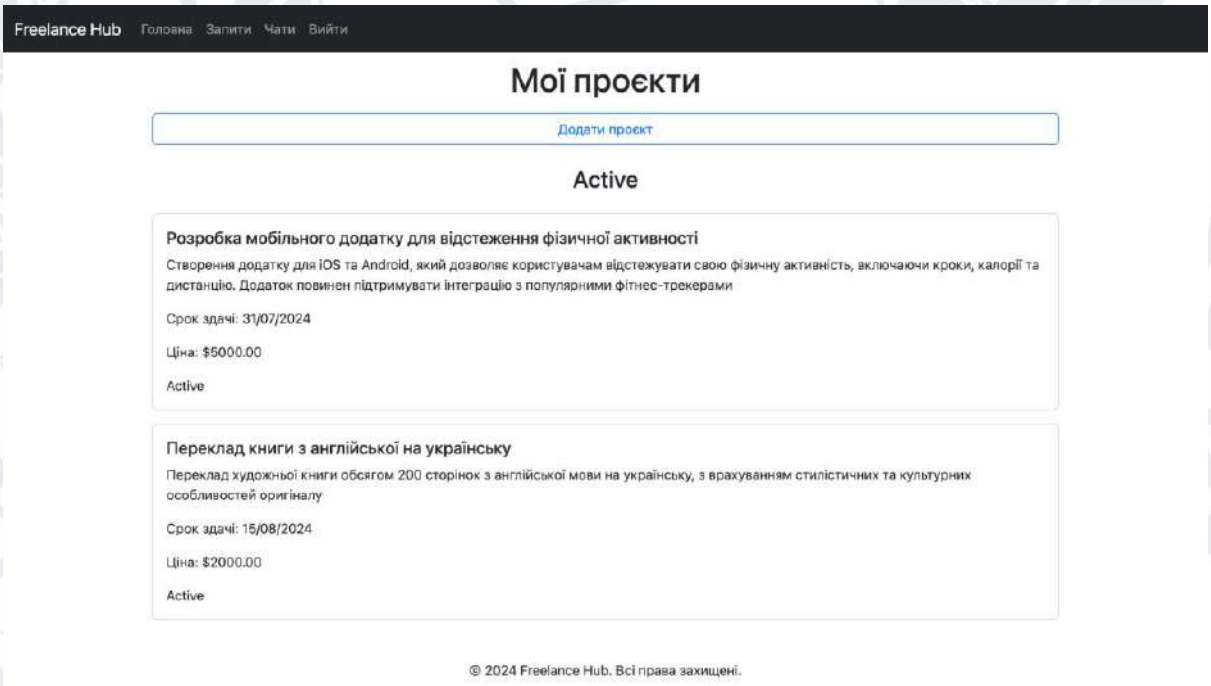


Рисунок 3.10 - Головна сторінка замовника

Щоб додати нове завдання достатньо натиснути кнопку та заповнити форму, і всі виконавці зможуть побачити та відправити запит на виконання даної роботи рис (3.11).

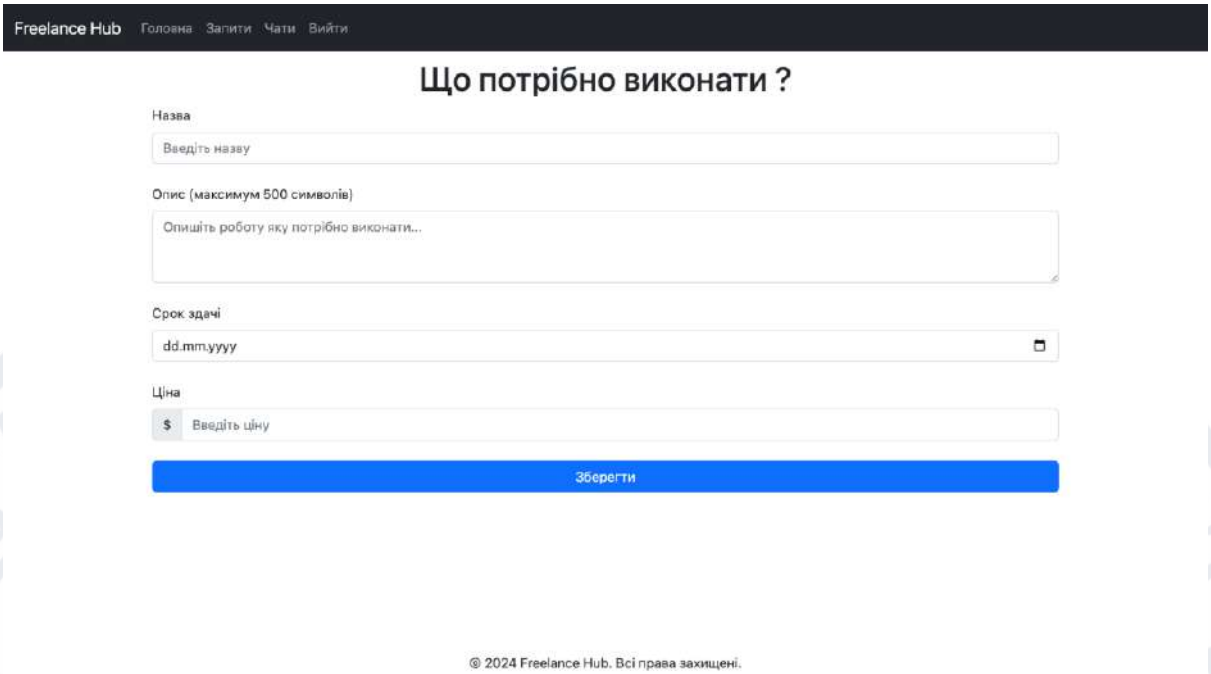


Рисунок 3.11 - Форма додавання нової роботи.

Далі, коли виконавці побачили ваше замовлення вони можуть відправити запит на виконання. У розділі Запити можна побачити всі запити (рис. 3.12), які розділені на 3 категорії: В очікування, Прийняті, та Відхилені. Тут можна прийняти чи відхилити запити від виконавців, також можна перейти на сторінку виконавця щоб переглянути його профіль та відгуки (рис. 3.13).

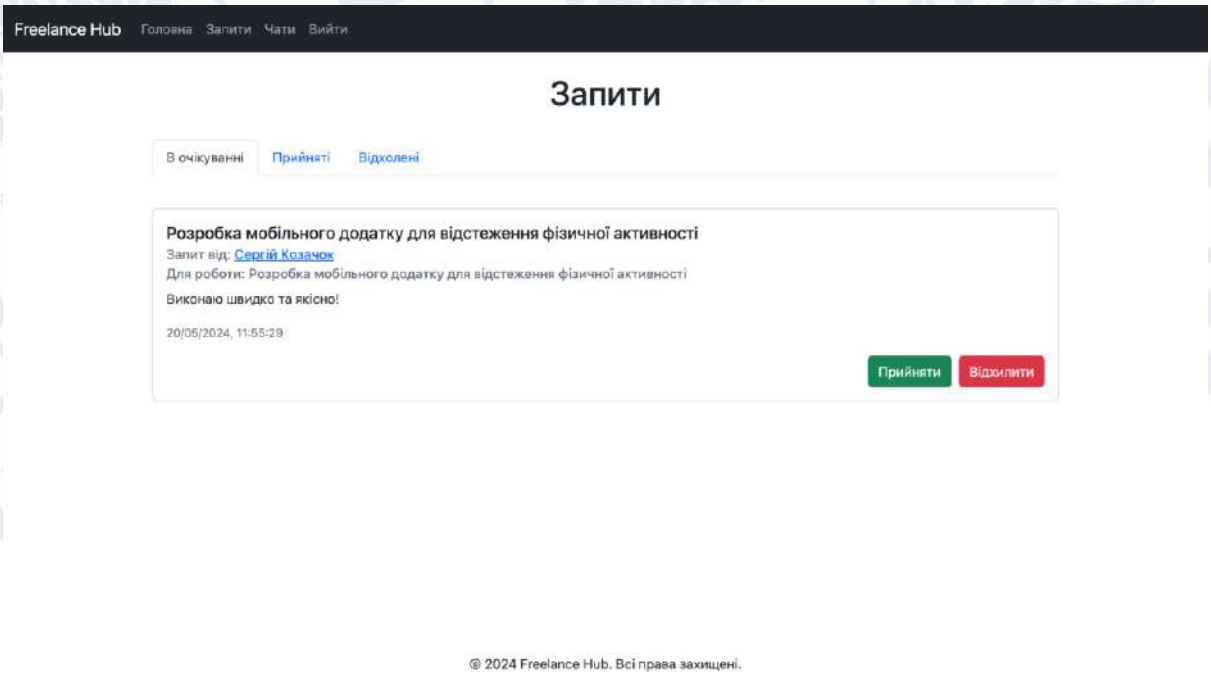


Рисунок 3.12 - Сторінка Запитів від виконавців.

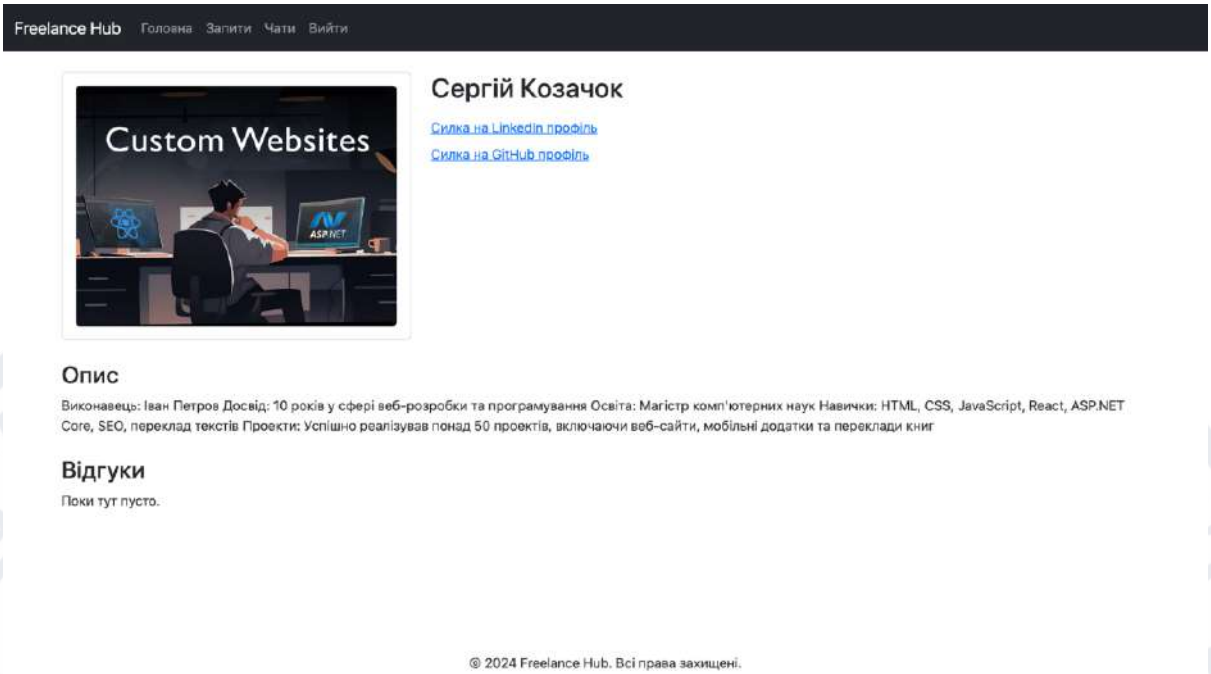


Рисунок 3.13 - Сторінка виконавця яку бачить замовник.

Якщо прийняти заявку, буде створений чат де дві сторони можуть обговорити всі деталі та питання у режимі реального спілкування за допомогою технології SignalR. Далі ми переходимо у новостворений чат (рис. 3.14), і після цього Замовник може оплатити угоду (рис. 3.15), у замовника відкривається сторінка для оплати за допомогою Stripe.

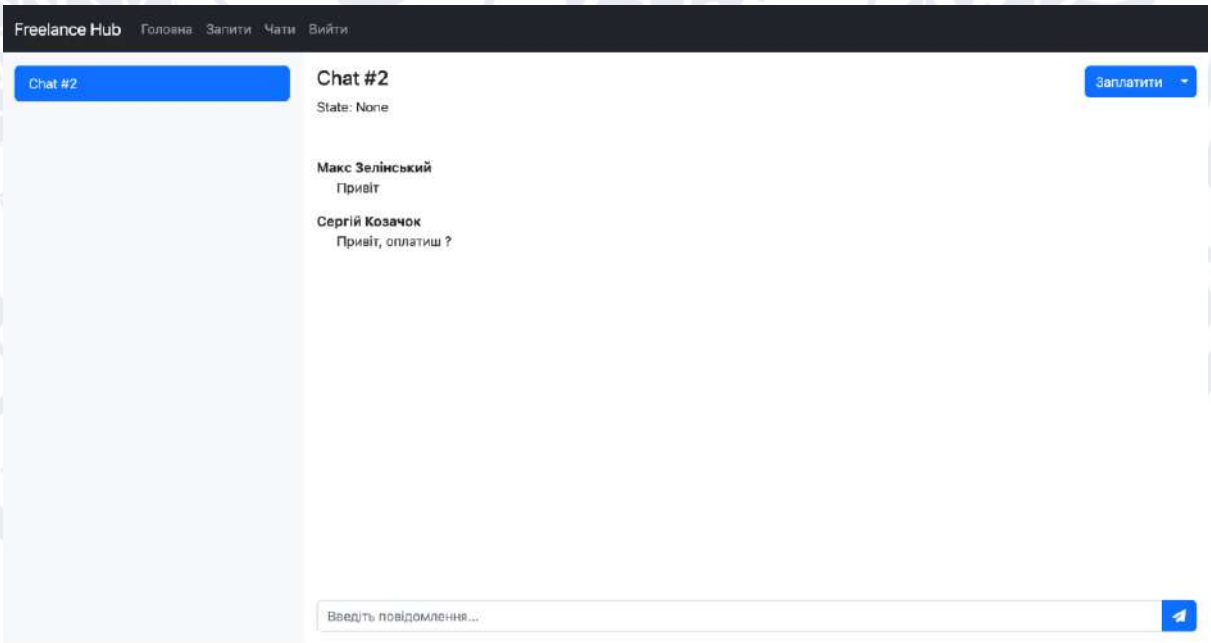


Рисунок 3.14 - Сторінка чату.

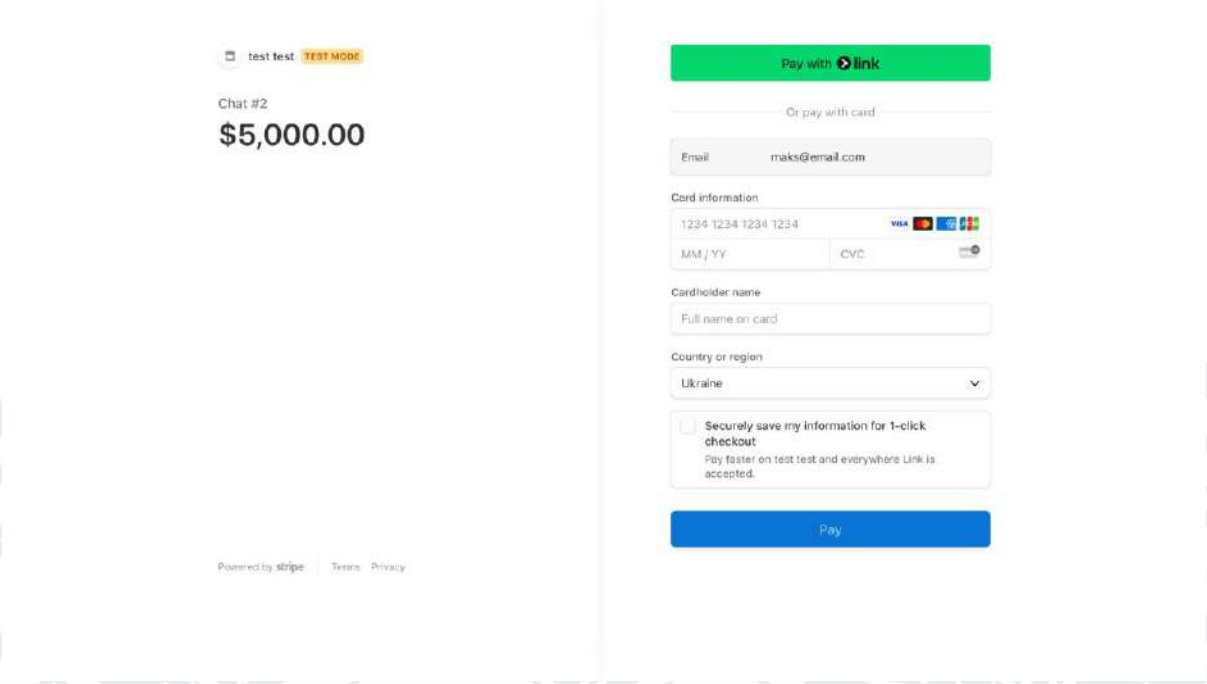


Рисунок 3.15 - Сторінка оплати.

Після оплати прийде сповіщення що угода було успішно проплачена (рис. 3.16), гроші не йдуть безпосередньо на аккаунт виконавця, а заморожуються.

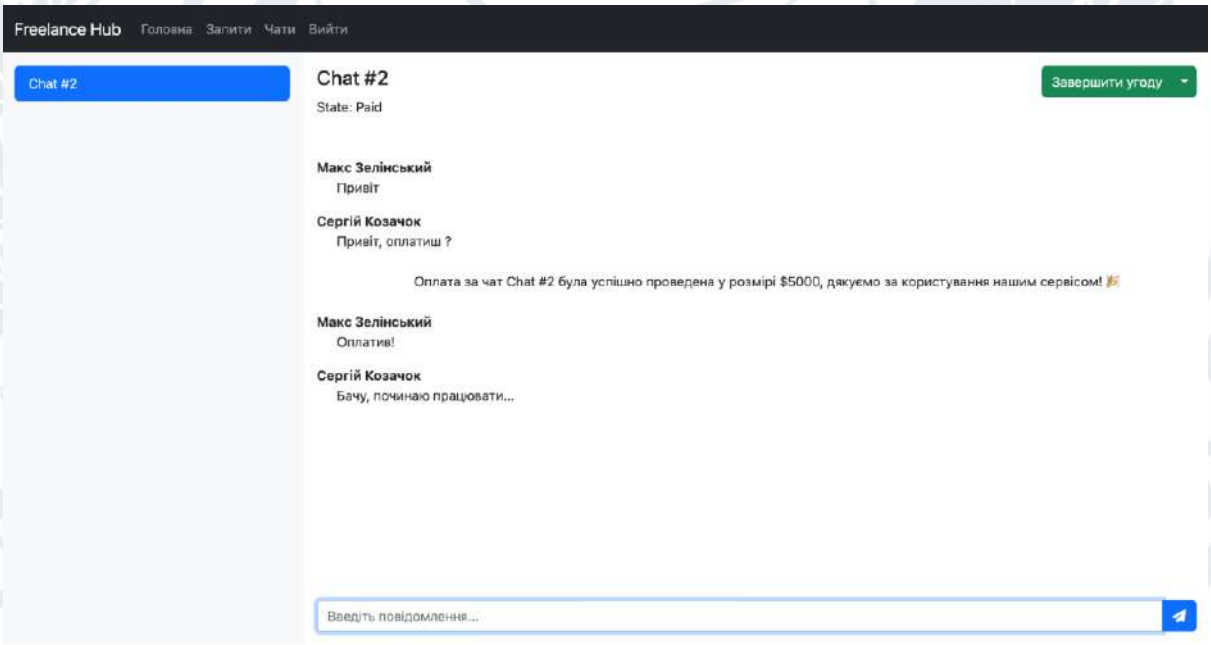


Рисунок 3.16 - Сповіщення про оплату.

Далі виконавець може виконувати завдання, та після завершення і згоди з замовником, замовник може завершити угоду, залишити коментар (рис. 3.17) і тоді гроші будуть зачислені на аккаунт виконавця (рис. 3.18).

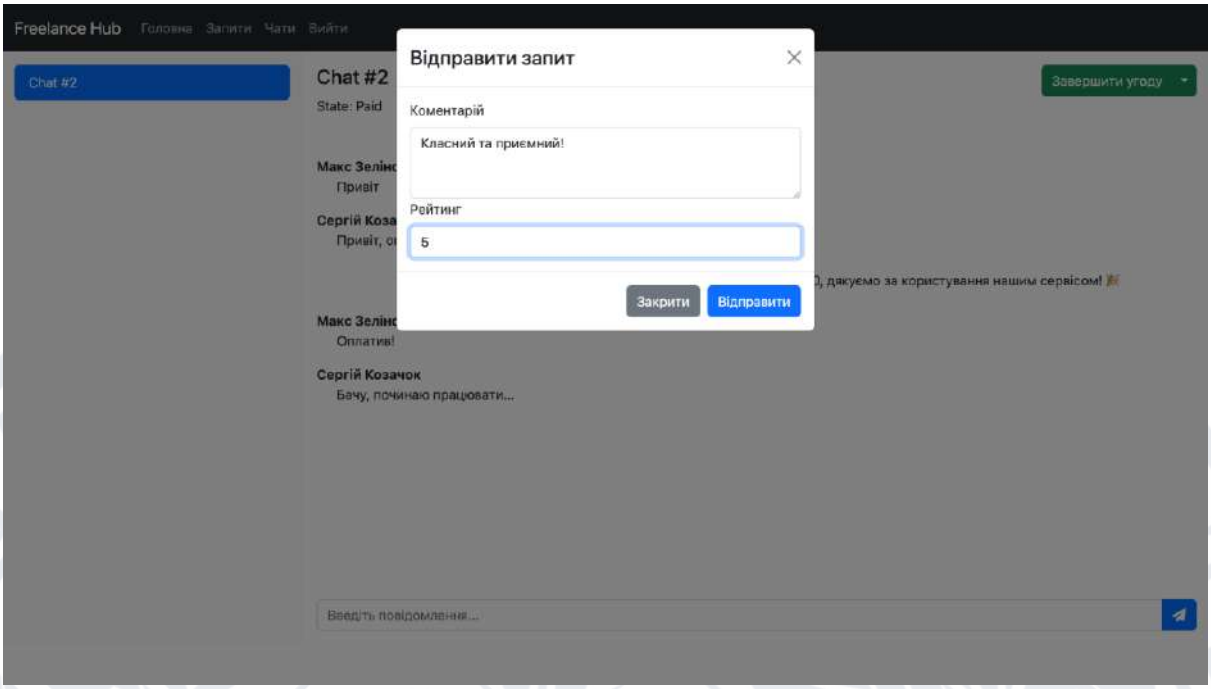


Рисунок 3.17 - Залишення відгуку.

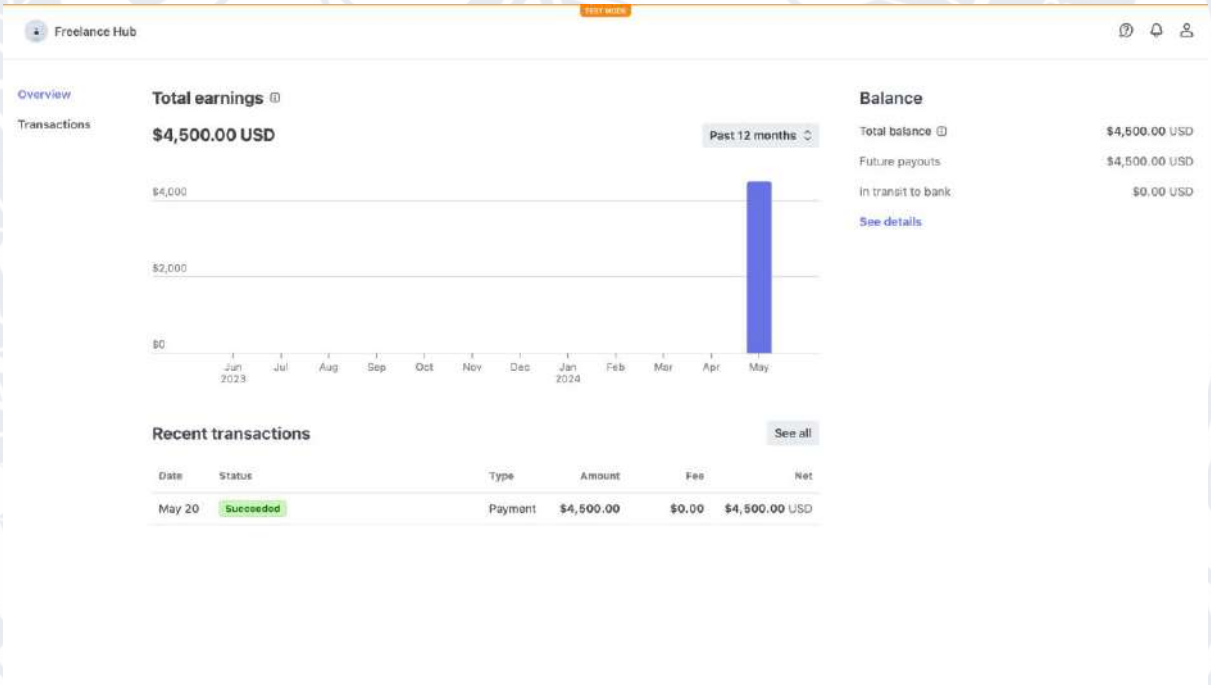


Рисунок 3.18 - Гроші були зачислені на аккаунт виконавця.

Якщо замовник оплатив, але виконавець не справився із поставленим завданням гроші повертаються замовнику на карту з якої він оплатив.

І останньою складовою проекту є функції виконавця. Виконавець має верифікувати свій профіль перш ніж почати працювати, в противному випадку він не зможе відправляти запити. Верифікацію можна пройти на вкладці Моя Сторінка (рис. 3.19).

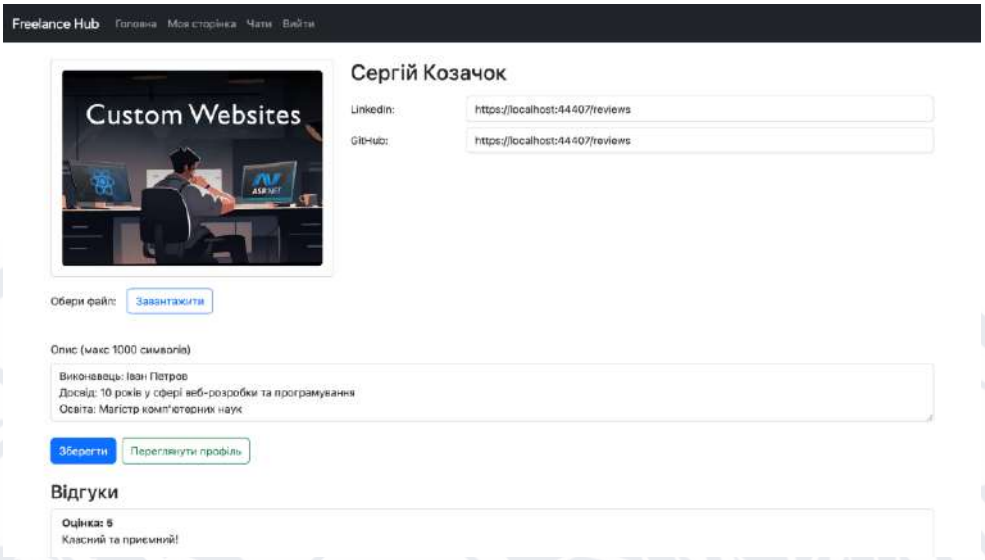


Рисунок 3.19 - Вкладка **Моя Сторінка.**

Верифікація складається з декількох етапів, всі ми не будемо розглядати але сама сторінка виглядає наступним чином (рис. 3.20).

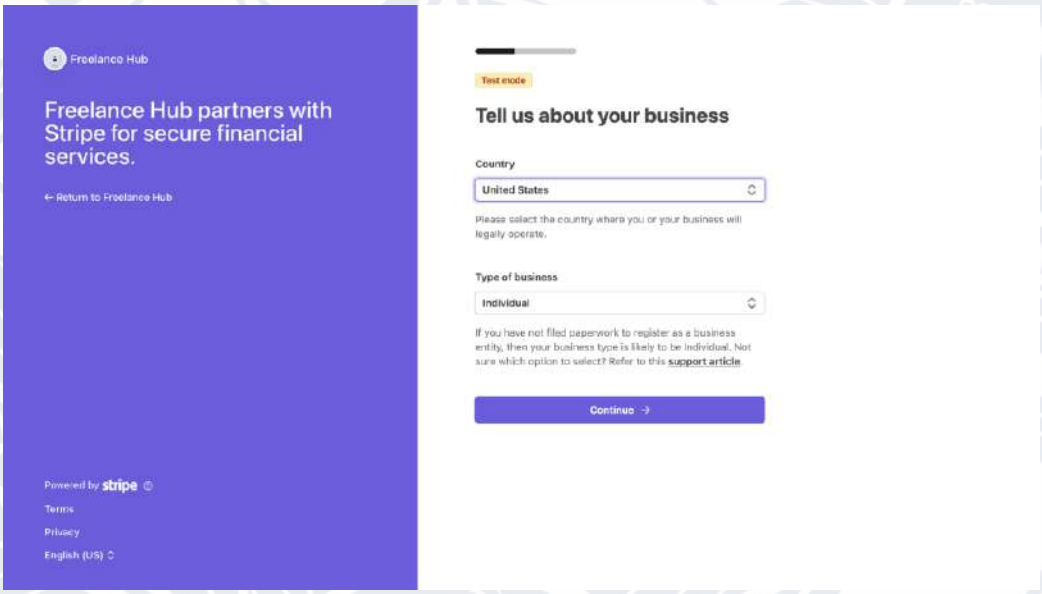


Рисунок 3.20 - Верифікація.

Після проходження верифікації виконавець може надсилати запити, та потім отримувати кошти за виконання робіт. Перейшовши на головну сторінку виконавця, він бачить усі доступні роботи та може шукати за назвою (рис. 3.21).

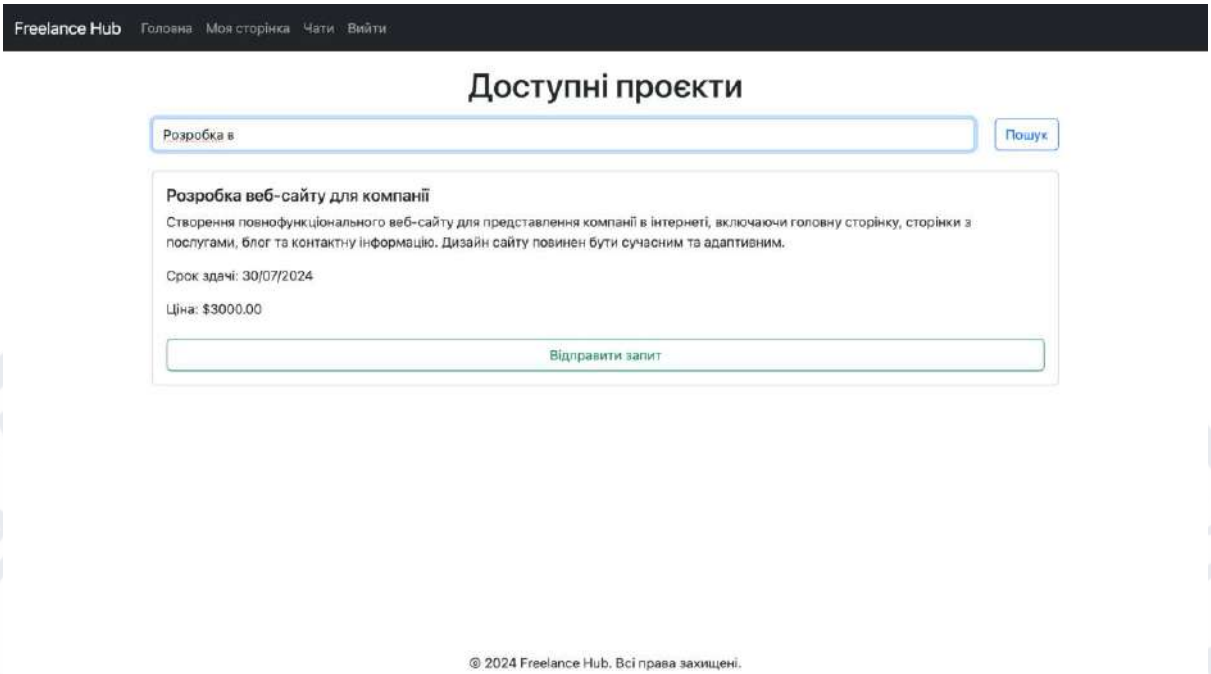


Рисунок 3.21 - Головна сторінка Виконавця з прикладом пошуку.

Надсилання запиту може відбуватись з коментарем (рис. 3.22). Далі запит йде на перевірку замовника і якщо останній хоче співпрацювати буде створений новий чат для обговорення всіх деталей.

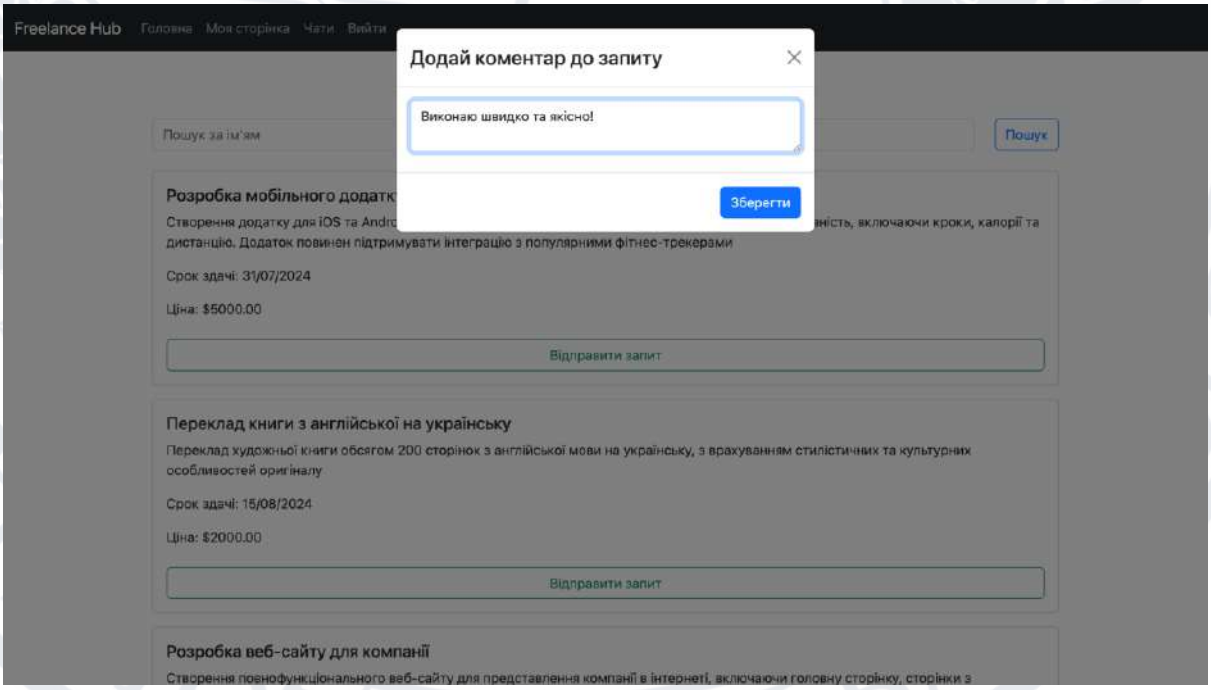


Рисунок 3.22 - Додавання коментаря до запиту

Також виконавець може переглядати свою статистику замовлень та оплат (рис. 3.23). Усі можливі питання та суперечки будуть регулюватись

адміністрацією вебсайту та може призвести до тимчасового чи до повного блокування аккаунту.

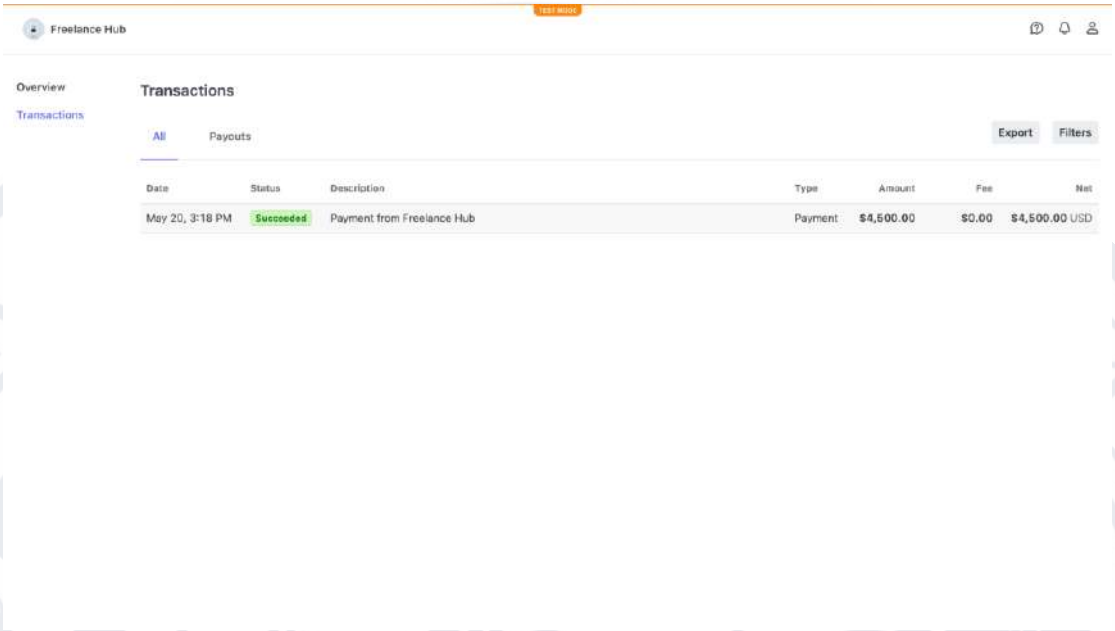


Рисунок 3.23 - Історія транзакцій.

## ВИСНОВОК ДО РОЗДІЛУ 3

У третьому розділі було детально розглянуто архітектуру проекту фріланс сервісу, який складається з трьох основних шарів: бізнес логіки, шару доступу до даних та презентаційного шару. Завдяки використанню сучасних технологій, таких як ASP.NET Core, React, та інших, вдалося створити надійну, масштабовану та зручну платформу для користувачів.

Також було розглянуто процес розробки проекту, включаючи етапи аналізу вимог, проектування, розробки, тестування та впровадження. Особлива увага приділялася інтеграціям з зовнішніми сервісами, такими як Google Drive API для завантаження фотографій та SignalR для реалізації реального часу спілкування у чаті. Ці інтеграції значно підвищують функціональність та зручність використання платформи.

В цілому, розробка даного проекту дозволила отримати цінний досвід у створенні складних веб-застосунків та інтеграції з різними сервісами, що є важливим кроком у професійному розвитку та підвищенні кваліфікації у сфері розробки програмного забезпечення.

# ВИСНОВОКИ

Розробка фріланс сервісу на основі сучасних веб-технологій стала важливим етапом у моєму професійному становленні як розробника. В рамках дипломної роботи було успішно створено платформу, яка забезпечує зручний і ефективний спосіб взаємодії між фрілансерами та замовниками.

Проект включав в себе ретельний аналіз вимог, проектування архітектури системи, реалізацію бізнес-логіки та інтерфейсу користувача, а також інтеграцію з зовнішніми сервісами для забезпечення додаткових функціональних можливостей. Зокрема, використання Google Drive API для завантаження медіафайлів та SignalR для реалізації реального часу спілкування в чаті зробило платформу більш зручною та ефективною для користувачів.

Основні технології, використані в проекті, включають ASP.NET Core для серверної частини, React для клієнтської частини, Entity Framework для роботи з базою даних та різноманітні інші інструменти для забезпечення безпеки та продуктивності системи. Завдяки цьому вдалося створити надійний, масштабований та зручний для користувачів сервіс.

Під час роботи над проектом я здобув цінні навички та знання, які значно покращили мою компетенцію як розробника програмного забезпечення. Цей досвід стане міцною основою для моєї майбутньої кар'єри та подальшого професійного зростання.

Успішна реалізація проекту підтвердила важливість детального планування, ефективного управління ресурсами та використання сучасних технологій у розробці програмного забезпечення. Цей проект є вагомим внеском у мій професійний портфель та значним кроком у напрямку досягнення моїх кар'єрних цілей.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Adam Freeman. "Pro ASP.NET Core MVC 2". Apress, 2017. 1017 p.
2. Andrew Troelsen, Philip Japikse. "Pro ASP.NET Core MVC 2". Apress, 2017. 1017 p.
3. Mark J. Price. "C# 9 and .NET 5 – Modern Cross-Platform Development". Packt Publishing, 2021. 786 p.
4. Dino Esposito. "Programming ASP.NET Core". Microsoft Press, 2018. 528 p.
5. Andrew Lock. "ASP.NET Core in Action". Manning Publications, 2018. 616 p.
6. Eric Lippert. "C# in Depth". Manning Publications, 2019. 528 p.
7. Jon Galloway, Damian Edwards, Scott Hanselman. "ASP.NET Core Application Development: Building an application in four sprints". Microsoft Press, 2019. 432 p.
8. Jesse Liberty. "Programming C# 8.0: Build Windows, Web, and Desktop Applications". O'Reilly Media, 2020. 878 p.
9. Bill Wagner. "C# 9.0 in a Nutshell: The Definitive Reference". O'Reilly Media, 2021. 1068 p.
10. John Sharp. "Microsoft Visual C# Step by Step". Microsoft Press, 2018. 816 p.
11. Jason De Oliveira, Michel Bruchet. "Building Single Page App With ASP.NET Core and Angular". Apress, 2019. 544 p.
12. Adam Freeman. "Pro ASP.NET Core Identity". Apress, 2019. 723 p.
13. Ricardo Peres. "Entity Framework Core in Action". Manning Publications, 2018. 520 p.
14. Robert C. Martin. "Clean Code: A Handbook of Agile Software Craftsmanship". Prentice Hall, 2008. 464 p.
15. Steve McConnell. "Code Complete: A Practical Handbook of Software Construction". Microsoft Press, 2004. 960 p.
16. Scott Millett, Brett Lonsdale. "Patterns, Principles, and Practices of Domain-Driven Design". Wiley, 2015. 720 p.

17. Martin Fowler. "Patterns of Enterprise Application Architecture". Addison-Wesley Professional, 2002. 560 p.
18. Eric Evans. "Domain-Driven Design: Tackling Complexity in the Heart of Software". Addison-Wesley Professional, 2003. 560 p.
19. Addison-Wesley Professional. "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley Professional, 1994. 416 p.
20. Freeman, Adam. "Pro React 16". Apress, 2019. 697 p.
21. Володимир Кравець. "ASP.NET Core 2.0 MVC & Razor Pages для професійних розробників". Львів: Видавництво Самміт-Книга, 2019. 544 с.
22. Владислав Галка. "Веб-програмування на C# і JavaScript". Київ: Діафра-К, 2020. 368 с.
23. Андрій Буда. "Професійна веб-розробка з ASP.NET Core 3.1". Київ: Наш Формат, 2020. 512 с.
24. Віталій Орешков. "ASP.NET Core: Сучасна веб-розробка на C# для професіоналів". Київ: БВС, 2019. 672 с.
25. Ігор Фесенко, Євген Башир. "ASP.NET Core. Професійна веб-розробка на C# і .NET Core для професіоналів". Київ: Видавництво "Світ книги", 2020. 768 с.
26. Юрій Федишин. "ASP.NET Core 2.0 розробка веб-додатків на C# для професіоналів". Київ: Діафра-К, 2019. 616 с.
27. Євген Ковалів. "ASP.NET Core 3.1 MVC з використанням стандартних шаблонів". Львів: Видавництво Самміт-Книга, 2020. 360 с.
28. Дмитро Павлюк. "Професійний React". Київ: КИЇВСЬКА ШКОЛА ЕКОНОМІКИ, 2020. 384 с.
29. Юрій Мінаков. "Реактивне програмування на прикладах". Київ: Факт, 2020. 400 с.
30. Володимир Ільїн. "Мобільна розробка з Xamarin.Forms для професіоналів". Київ: БХВ-Петербург, 2020. 416 с.
31. Олексій Літвіненко. "Професійний Vue.js". Київ: Видавництво "Будинок книги", 2020. 392 с.

32. Андрій Кучерявий. "Програмування на мові C# для початківців". Львів: Видавництво Самміт-Книга, 2020. 368 с.
33. Володимир Поліщук. "Українська програмістка. Стартап від А до Я". Київ: Видавництво "Видавництво Світ", 2019. 336 с.
34. Андрій Дубінський. "Школа програмування". Київ: БХВ-Петербург, 2020. 528 с.
35. Ігор Потурайло. "Програмування на C# з використанням .NET Framework". Львів: Видавництво Самміт-Книга, 2020. 368 с.
36. Євген Набок. "Програмування на мові JavaScript". Київ: БХВ-Петербург, 2020. 576 с.
37. Олександр Кульчицький. "Vue.js: Повний курс". Київ: Діафра-К, 2020. 368 с.
38. Ігор Кущ. "Інтернет-магазин на базі ASP.NET Core та Angular". Київ: Факт, 2020. 432 с.
39. Олександр Данилов. "Програмування на мові Python для початківців". Київ: Факт, 2020. 480 с.
40. Сергій Гончаренко. "HTML5 та CSS3. Веб-програмування для початківців". Київ: Факт, 2020. 464 с.

# ДОДАТКИ

## ДЕКЛАРАЦІЯ
про дотримання академічної доброчесності

Я, _____
_____
__

*Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)*

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

**ЗОБОВ'ЯЗУЮСЬ:**

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

**ПІДТВЕРДЖУЮ:**

що мені відомі положення статті 42 Закону України «Про освіту»; що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації; що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

**УСВІДОМЛЮЮ:**

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів; що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики

Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)