

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Сайт фрілансу»

Виконав студент групи ІІЗ-19-3
ЖУРБЕЙ Артем Валентинович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЖУРБЕЙ Артем Валентинович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Сайт Фрілансу»**,

затверджена Наказом закладу вищої освіти від «14» лютого 2023 р. №70с

Вихідні дані для роботи: **вебдодаток для пошуку виконавців проєктів з системою рейтингу та оплати.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р.С.	16.02.2023р.	16.02.2023р.
2	Савіцький Р.С.	04.04.2023р.	04.04.2023р.
3	Савіцький Р.С.	02.05.2023р.	02.05.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі.	17 лютого 2023 – 27 лютого 2023	Виконано
2	Пошук, огляд та аналіз аналогічних розробок.	10 березня 2023 – 15 березня 2023	Виконано
3	Формулювання технічного завдання.	16 березня 2023 – 20 березня 2023	Виконано
4	Опрацювання літературних джерел.	13 квітня 2023 – 18 квітня 2023	Виконано
5	Проектування структури системи	25 квітня 2023 – 7 травня 2023	Виконано
6	Написання програмного коду	12 травня 2023 – 12 червня 2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається безпосередньо з вебдодатку фріланс та пояснювальної записки. Пояснювальна записка до випускної роботи містить 64 сторінок, 46 ілюстрацій та 16 таблиць.

Метою роботи є створення сайту пошуку роботи з рейтинговою системою, оплатою праці та функцією арбітражу.

В роботі було визначено основні цілі для розробки вебдодатку, проаналізовано різні методи вирішення поставленої задачі, аргументовано обрану архітектуру для сайту та проведено аналіз методів обробки, аналізу даних. Окрім цього, був наведений детальний опис алгоритму роботи додатку, а також розглянуто архітектуру та взаємодію його основних компонентів.

КЛЮЧОВІ СЛОВА: ВЕБДОДАТОК, ФРІЛАНС, ГЕНЕРАЦІЯ QR КОДУ, MYSQL, TYPEORM.

					ІПЗ.КР.Б-121-23-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Сайт фрілансу Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Журбей А.В.					6	64
Перевір.						ФІКТ Гр. ІПЗ-19-3[2]		
Керівник		Савіцький Р.С.						
Н. контр.								
Зав. каф.		Морозов А.В.						

ABSTRACT

The bachelor thesis consists of a software complex of the attendance control system and an explanatory note. The explanatory note to the final thesis contains 64 pages, 46 illustrations and 16 tables.

The goal of the work is to create a freelance website.

In the work, the main goals for the development of the web application were defined, various methods of solving the task were analyzed, the chosen architecture for the site was argued, and the methods of data processing and analysis were analyzed. A detailed description of the algorithm of the application was also given, as well as the architecture and interaction of its main components were considered.

KEY WORDS: WEB APPLICATION, FREELANCE, QR CODE GENERATION, MYSQL, TYPEORM.

					ІПЗ.КР.Б-121-23-ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	9
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ФРІЛАНСУ	5
1.1 Постановка задачі	5
1.2 Аналіз аналогів програмного продукту	6
1.3 Вибір архітектури системи контролю відвідуваності	11
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	15
Висновки до першого розділу.	16
РОЗДІЛ 2. ПРОЄКТУВАННЯ САЙТУ ФРІЛАНСУ.....	18
2.1 Визначення варіантів використання	18
2.2 Розробка бази даних системи.....	31
2.3 Проєктування та реалізація алгоритмів роботи системи	34
Висновки до другого розділу	36
3.1 Структура інтерфейсу вебдодатку	37
3.2 Тестування роботи програмного комплексу	48
Висновки до третього розділу	50
ВИСНОВКИ.....	52
ДОДАТКИ.....	57
ДОДАТОК А.....	58

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Web API – прикладний програмний інтерфейс

SOLID – набір принципів об'єктно-орієнтованого програмування

БД – база даних

ПЗ – програмне забезпечення

ORM – об'єктно-реляційна проєкція

MySQL – система управління реляційними базами

UI – користувацький інтерфейс

MHz – одиниця виміру частоти

FRP – функціонально-реактивне програмування

OOP – об'єктно-орієнтоване програмування

FP – функціонального програмування

					ІПЗ.КР.Б-121-23-ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. В останні роки спостерігається значне зростання кількості людей, які обирають роботу на фрілансі. У Freelancehunt фіксують стрімке зростання реєстрацій фрілансерів на платформі, а початок року відзначився новими піками, навіть вищими за зростання у перші місяці війни. За словами адміністрації сервісу, зовсім трохи лишилося до позначки у 20 000 зареєстрованих нових спеціалістів за місяць. Це пов'язано з розвитком інтернету, цифрових технологій та можливості віддаленої роботи. Багато людей воліють гнучкий графік роботи, свободу вибору проєктів та можливість працювати з будь-якої точки світу.

З розвитком технологій з'явилися різні цифрові платформи та ринки фрілансу, які полегшують пошук роботи та знаходження клієнтів. Це включає платформи, такі як Upwork, Freelancer, Fiverr та інші. Такі платформи створюють можливості для фрілансерів пропонувати свої послуги та знаходити замовників, що робить фріланс доступним для широкого кола людей.

Сучасні технології зв'язку та комунікації роблять віддалену роботу більш доступною та ефективною. Замовники можуть наймати фрілансерів із різних країн, що розширює можливості обох сторін. Фрілансери мають можливість працювати на міжнародному рівні, отримуючи доступ до клієнтів з усього світу.

Фріланс надає можливість працювати в різних галузях, таких як вебробота, дизайн, письменство, маркетинг, переклади та багато іншого. Це дозволяє людям вибирати область, яка відповідає їхнім інтересам.

Враховуючи всі ці чинники, можна дійти до висновку, що тема фрілансу є актуальною. Вона надає можливості для особистого та професійного зростання.

Метою випускної роботи є розробка вебсайту фрілансу, кінцевий результат якого, повинен спростити процес пошуку роботи для фрілансерів та знаходження клієнтів для виконавців.

Встановлена мета обумовлює наступні завдання:

					ІПЗ.КР.Б-121-22-ПЗ	
						3

1. проведення аналізу та вимог до фрілансу;
2. аналіз аналогів програмного продукту;
3. обґрунтування та вибір засобів реалізації;
4. розробка вебресурсу та реалізація всіх визначених вимог до програмного забезпечення;
5. розробка бази даних;
6. опис структури інтерфейсу сайту;
7. тестування сайту.

Об'єктом дослідження є сайт фрілансу.

Предметом дослідження є розгляд платформи фріланс яка допомагає знайти роботу, а також зробити проєкт за найкращу вартість і якість.

В роботі було використано методи об'єктно-орієнтованого програмування і проєктування на бекенді, а також принципи розробки дизайну інтерфейсів на фронтенді.

На даному етапі розроблений сайт буде використовуватись для особистих потреб. При необхідності, вебдодаток може бути вдосконалений шляхом додавання нових функцій та модернізації вже наявного функціонала. Додаток буде оновлюватись в зв'язку з розвитком технологій та потреб користувачів.

Якщо розглянути сайт як джерело прибутку, то в подальшому його можна використовувати для різних цілей і задач. Наприклад, сайт відповідає необхідним критеріям для ведення бізнесу, залучення клієнтів або партнерів, надання послуг за визначену ціну.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ФРІЛАНСУ

1.1 Постановка задачі

Розробка сайту фрілансу повинна виконувати наступні функції: автентифікація, розміщення та пошук проєктів, управління замовленнями та платіжками, рейтингову систему користувачів.

Автентифікація повинна реалізовувати наступні можливості:

1. авторизація;
2. реєстрацію.

Вебсервіс має представляти собою додаток, за допомогою якого можна буде отримувати інформацію про проєкти. У цьому додатку повинні бути дві ролі користувачів: замовник та фрілансер.

Замовники повинні мати змогу:

1. створювати нові замовлення;
2. оцінювати якісь виконаної роботи;
3. редагувати особистий профіль;
4. обирати найкращу пропозицію серед пропозицій фрілансерів;
5. отримувати рейтинг фрілансерів.

Основними етапами розробки описаного сайту є:

1. Розробка серверної частини вебдодатку;
 - 1.1. вибір технологій;
 - 1.2. розробка API, яке буде визначати набір ендпоінтів, до яких клієнтська частина може звертатись для отримання даних або виконання операцій;
 - 1.2.1. написання програмних модулів для:
 - 1.2.1.1. аутентифікації та аутеризації користувачів;
 - 1.2.1.2. створення заявок робіт;
 - 1.2.1.3. створення тегів для робіт;

1.2.1.4. створення відгуків на роботи;

1.3.створення бази даних та імплементація логіки зберігання та взаємодії з даними;

1.3.1. створення моделей та зв'язків між ними для взаємодії серверу з базою даних за допомогою ORM;

1.4.реалізація бізнес-логіки яка буде відповідати за валідацію даних, обробку помилок та інше. Цей етап також включає розробку автентифікації та авторизації;

2. розробка клієнтської частини сервісу;

2.1.вибір технологій;

2.2.розробка інтерфейсу;

2.3.розробка логіки для взаємодії з серверною частиною за допомогою API-запитів;

2.4.перевірка та оптимізація продуктивності клієнтської частини для швидкого завантаження та відгуку;

3. наповнити систему інформацією.

Результатом виконання поставлених завдань є створення сайту, на якому люди зможуть знаходити собі роботу або робітників.

1.2 Аналіз аналогів програмного продукту

Для розробки сучасного, конкурентоспроможного та функціонального програмного забезпечення важливо провести аналіз аналогічних рішень. Під час дослідження вебдодатків було виявлено значну кількість аналогічних продуктів, з якими можна порівняти готовий продукт. Під час проведення аналізу були розглянуті наступні рішення:

Upwork [1], freelance.ua [2], fiverr [3].

Upwork (рис.1.1). Однією з основних переваг цього сайту є велика база фрілансерів з усіх куточків світу, що дозволяє знайти виконавців з різноманітними навичками та досвідом.

Основними перевагами **Upwork** є:

					ІПЗ.КР.Б-121-22-ПЗ	
						6

1. різноманітність, можна знайти проєкти з різних галузей та областей;
2. гнучкість, фрілансери можуть працювати на власний розсуд із будь-якого місця, де є доступ до Інтернету;
3. оплата за захист, Upwork забезпечує систему оплати, яка гарантує безпеку для фрілансерів та клієнтів.

Недоліками є:

1. конкуренція, заради привабливих проєктів на сайті може бути велика конкуренція серед фрілансерів;
2. платні послуги, сайт збирає певний відсоток від оплати кожного проєкту, а також кількість робіт на які може відкликнутись фрілансер обмежена і якщо він захоче збільшити число відкликів, йому прийдеться заплатити за це.

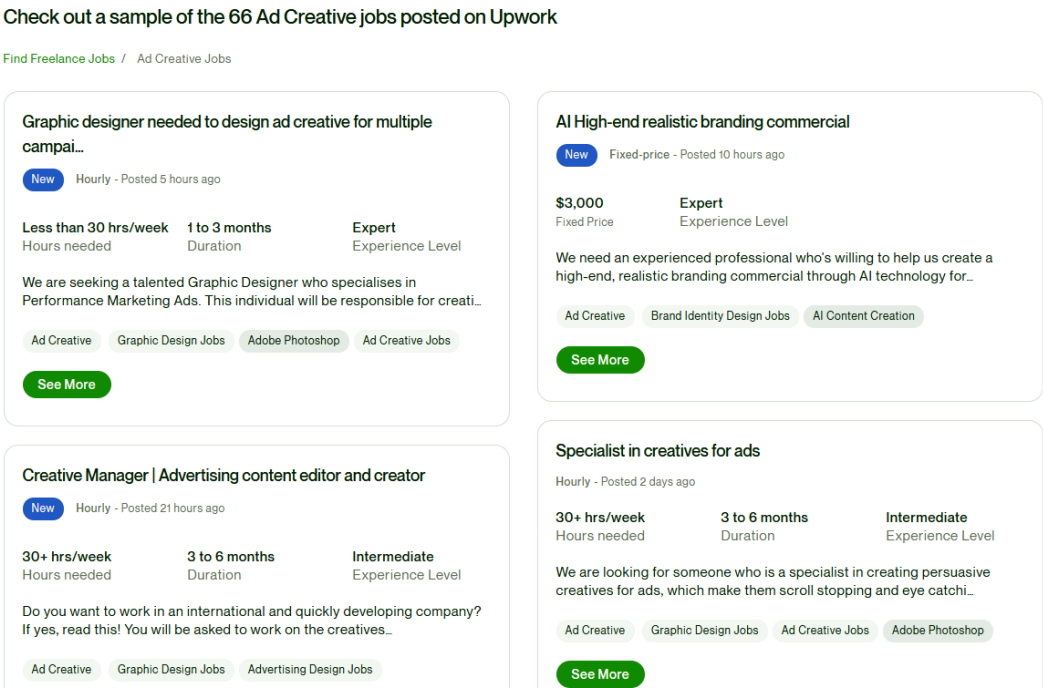


Рис 1.1 Сторінка пошуку роботи сайту Upwork

Freelance.ua (рис.1.2) – це популярна українська платформа для пошуку та здійснення фріланс-проєктів

Основними перевагами Freelance.ua є:

1. українська спеціалізація: Freelance.ua спеціалізується на українському ринку та залучає клієнтів та фрілансерів з України;
2. наявність локальних проєктів: на Freelance.ua можна знайти проєкти, які прямо пов'язані з українськими компаніями або ринком. Це мо-

же бути вигідно для фрілансерів, які хочуть працювати з місцевими клієнтами або отримувати проєкти, специфічні для українського ринку;

3. локалізована підтримка: Freelance.ua надає підтримку на українській мові, що може спростити комунікацію та вирішення проблем для українських користувачів.

Недоліками є:

1. обмежена глобальна експозиція: У порівнянні з глобальними платформами, такими як Upwork або Freelancer.com, Freelance.ua може мати обмежений обсяг проєктів і клієнтів. Це може зменшити кількість доступних можливостей для фрілансерів;
2. обмежений спектр проєктів: Freelance.ua може бути спеціалізованою на певних типах проєктів або індустріях, що може обмежити розмаїтість доступних можливостей для фрілансерів.

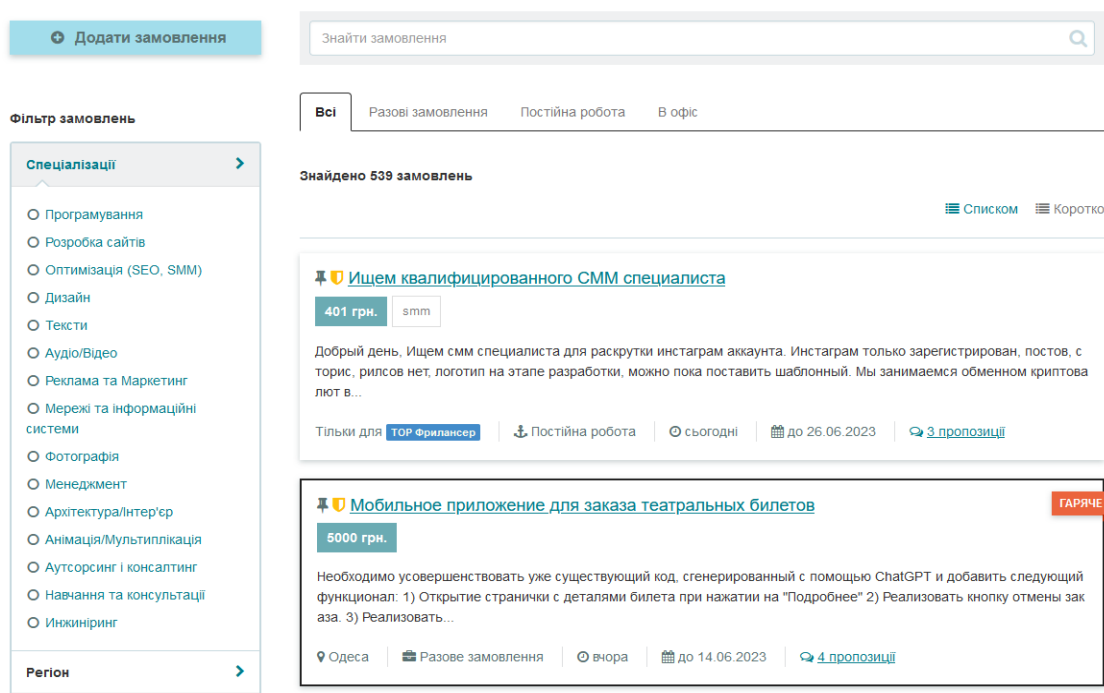


Рис 1.2 – сторінка пошуку роботи сайту Freelance.ua

Fiverr (рис.1.3) – популярна глобальна платформа для фрілансу, де фрілансери можуть пропонувати свої послуги, а клієнти можуть шукати та наймати фрілансерів для виконання різних проєктів.

Основними перевагами Fiverr є:

					ІПЗ.КР.Б-121-22-ПЗ	8

1. широкий вибір фахівців: Fiverr має велику базу фрілансерів з різних областей, що дозволяє клієнтам знайти відповідного фахівця для свого проєкту;
2. конкурентні ціни: За рахунок великої кількості фрілансерів на платформі, на Fiverr можна знайти різні цінові пропозиції для послуг. Це може бути вигідним для клієнтів, оскільки вони можуть вибрати найбільш прийнятну для них ціну;
3. зручний пошук та оцінки: Fiverr має зручну систему пошуку, фільтрів та оцінок, яка допомагає клієнтам знайти відповідних фрілансерів залежно від їх потреб і попередніх відгуків інших клієнтів.

Недоліками є:

1. обмежені можливості спілкування: На Fiverr комунікація між клієнтом та фрілансером обмежена платформою. Це може ускладнити спілкування та обговорення проєктних деталей;
2. висока комісія: Fiverr збирає комісію з кожної угоди між клієнтом та фрілансером, що може призвести до вищих витрат для обох сторін.

Logo Design

Stand out from the crowd with a logo that fits your brand personality. | ● How Fiverr Works

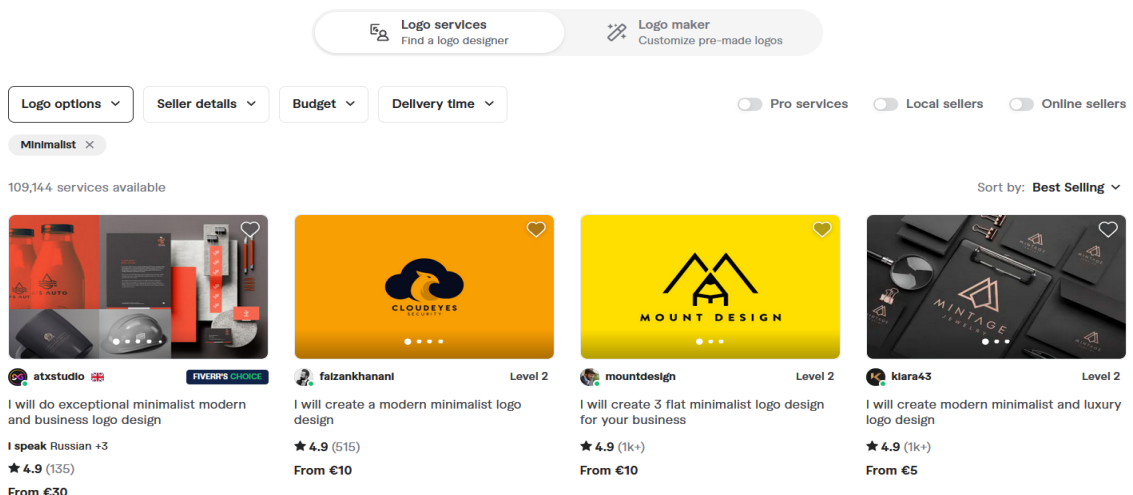


Рис 1.3 – Сторінка пошуку роботи сайту Fiverr

Базуючись на отриманих даних, складемо порівняльну характеристику згаданих платформ та власного вебдодатку(табл. 1.1).

Таблиця 1.1

Характеристика	Upwork	Freelance.ua	Fiverr	Власний вебдодаток
Масштаб	Глобальна платформа	Місцева платформа	Глобальна платформа	Місцева платформа
сайт	Великий	Обмежений	Великий	Обмежений
Комісія	Висока	Залежить від проєкту	Висока	Немає
Конкуренція	Висока	Середня	Висока	Низька
Функціональність	Розширена	Базова	Розширена	Базова
Зручність користування	Висока	Середня	Висока	Висока

Після порівняння характеристик чотирьох платформ фрілінсу - Upwork, Freelance.ua, Fiverr і власний вебдодаток, можна зробити наступні висновки:

- масштаб: Upwork та Fiverr є глобальними платформами, тоді як Freelance.ua та власний вебдодаток є місцевою платформою. Це означає, що Upwork та Fiverr можуть надати доступ до більш широкої бази фрілансерів та клієнтів;
- вибір спеціалістів: Upwork, Fiverr та власний вебдодаток мають великий вибір фахівців з різних галузей, тоді як вибір на Freelance.ua є обмеженим. Це означає, що на Upwork, Fiverr та на власному вебдодатку ви знайдете більше спеціалістів, що відповідають вашим потребам;
- комісія: Upwork та Fiverr вимагають високих комісій з угод, що може вплинути на вартість проєкту. У Freelance.ua комісія залежить від конкретного проєкту. У власного вебдодатку взагалі немає комісії що ніяк не може вплинути на вартість проєкту;

4. конкуренція: на Upwork та Fiverr конкуренція серед фрілансерів може бути високою, що може вплинути на ціни та доступність проєктів. У Freelance.ua конкуренція є середня. Оскільки власний вебдодаток тільки розвивається на ньому все ще доволі мала конкуренція серед фрілансерів;
5. функціональність та зручність користування: Upwork та Fiverr мають розширені функціональні можливості та високий рівень зручності користування. Freelance.ua та власний вебдодаток мають базові функціональні можливості.

У підсумку, можна зробити висновок, що для початківців найкращим вибором буде наш власний вебдодаток. Це обумовлено тим, що наша платформа не стягує комісію з транзакцій, що дозволяє користувачам зекономити кошти. Крім того, конкуренція серед фрілансерів на нашій платформі є мінімальною порівняно з іншими відомими площадками.

1.3 Вибір архітектури вебдодатку фріланс

Вебдодаток складається з компонентів клієнтської та серверної частин. Серверна частина, яка являє собою Web API, була спроектована з використанням принципів SOLID та розділенням відповідальностей [4].

Основні компоненти архітектури:

1. сервер. Це сам сервер, який використовується для обробки запитів та надання відповідей;
2. база даних: включає міграції для керування схемою бази даних та налаштування бази даних;
3. джерело даних: містить сутності, які відображають таблиці бази даних та взаємозв'язки між ними;
4. модулі: Розділяють функціональність програми на логічні модулі, наприклад, модуль "job". Включає контролер для обробки HTTP-запитів [5], інтерфейс для опису даних, модуль для імпорту та на-

лаштування залежностей, сервіс для бізнес-логіки та DTO [6] для передачі даних між компонентами;

5. сервіси: Це перевикористовувані компоненти, які можуть бути використані в різних модулях або частинах програми;
6. головний модуль та головний сервіс: Це основні компоненти програми, які об'єднують різні модулі та сервіси разом.

В цілому, архітектура застосунку використовує незалежні модулі, використання сервісів для бізнес-логіки та інші принципи, що допомагають створити розширювану та легко зрозумілу програму.

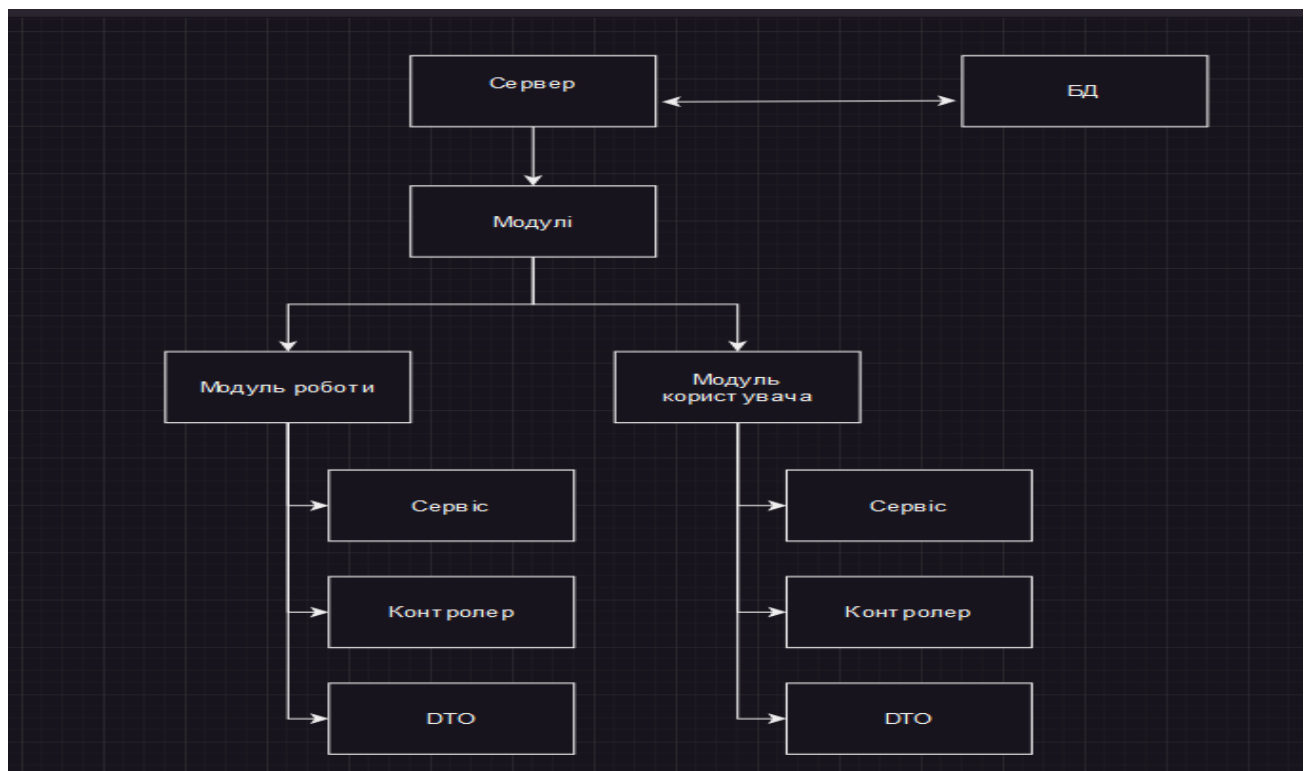


Рис. 1.4 – Спрощена архітектура проєкту

На сьогоднішній день існує розмаїття архітектурних підходів у вебдодатках, кожен з яких має свої переваги та недоліки. Вибір конкретної архітектури залежить від поставлених цілей та вимог. Для розробки фронт-енд частини можна розглянути такі архітектурні рішення:

1. монолітна [7];
2. мікросервісна [8];
3. безсерверна [9].

Monolithic Architecture (Монолітна архітектура):

					ІПЗ.КР.Б-121-22-ПЗ	
						12

1.1.переваги:

- 1.1.1. простота в розробці та випуску;
- 1.1.2. легка масштабованість на невеликому рівні;
- 1.1.3. простота в тестуванні та супроводі.

1.2.недоліки:

- 1.2.1. великі розміри проєкту, що може спричинити проблеми з управлінням та розширенням;
- 1.2.2. потенційна залежність між компонентами, що може впливати на стійкість та надійність системи;
- 1.2.3. складності в оновленні окремих компонентів без впливу на всю систему.

Microservices Architecture (Мікросервісна архітектура):

1.1.переваги:

- 1.1.1. можливість незалежно розгортати та масштабувати окремі мікросервіси;
- 1.1.2. полегшена розробка, оновлення та тестування окремих компонентів;
- 1.1.3. можливість використовувати різні технології та мови програмування для кожного мікросервісу.

1.2.Недоліки:

- 1.2.1. складність в управлінні та координації між мікросервісами;
- 1.2.2. велика кількість мережових викликів, що можуть призвести до збільшення затримок та складнощів у тестуванні;
- 1.2.3. потреба в розробці та підтримці інфраструктури для управління мікросервісами.

Serverless Architecture (Безсерверна архітектура):

1.1.переваги:

- 1.1.1. автоматичне масштабування та керування ресурсами в залежності від навантаження;

- 1.1.2. знижені витрати на інфраструктуру, оскільки платиться лише за виконання функцій;
- 1.1.3. швидкий випуск нових функцій та можливість концентруватися на розробці функціоналу.

1.2.Недоліки:

- 1.2.1. обмежені можливості контролю та налаштування інфраструктури;
- 1.2.2. потреба уважного планування та розробці функцій, щоб уникнути накладання збільшених витрат;
- 1.2.3. складне тестування та локальний розвиток, оскільки функції виконуються в середовищі хмари.

Після проведення порівнянь згаданих варіантів та аналізу вимог додатку було обрано монолітну архітектуру.

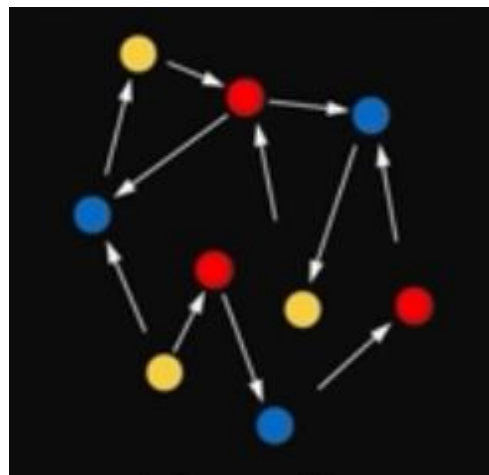


Рис. 1.5 – Результат використання монолітної архітектури

Всі описані компоненти: сервер, клієнтська частина та база даних сполучаються двонаправленим зв'язком між собою (рис. 1.6).



Рис. 1.6 – Взаємодія компонентів системи

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Розробка вебсайту фрілансу вимагає ретельного підходу та уваги до деталей. Нижче буде приведено перелік обраних для розробки технологій.

Розробка клієнтського додатку відбуватиметься в інтегрованому середовищі розробки – Visual Code [10]. Для розробки клієнтської частини були обрані наступні засоби:

1. React – це популярний JavaScript-фреймворк для розробки користувацького інтерфейсу (UI) вебдодатків [11];
2. Redux – бібліотека для управління й використання стану додатку [12];
3. Ant Design – це набір готових компонентів для розробки користувацького інтерфейсу (UI) вебдодатків. Він базується на бібліотеці React та надає готові компоненти, стилізацію й інструменти для швидкої розробки сучасних і привабливих інтерфейсів [13];
4. TypeScript є розширенням JavaScript, яке надає статичний типізацію, об'єктно-орієнтований підхід та додаткові можливості для розробки вебдодатків [14];
5. ESLint є інструментом статичного аналізу коду, який використовується для виявлення потенційних проблем, помилок та недоліків форматування у коді [15].

Завдяки обраним інструментам та технологіям, реалізація клієнтського додатку була легкою та зрозумілою для розробника.

Розробка серверної частини сервісу виконувалася в Visual Code за допомогою інструментів:

1. Node.js – це відкрите середовище виконання JavaScript, побудоване на рушії V8. Воно дозволяє виконувати JavaScript-код на серверній стороні та створювати різноманітні вебдодатки та сервіси [16];
2. NestJs – фреймворк для розробки серверних додатків на мові TypeScript або JavaScript, побудований на основі Node.js [17];

3. TypeORM – бібліотека для роботи з базами даних в середовищі Node.js, яка надає ORM для зручного та ефективного взаємодії з реляційними базами даних [18];
4. MySQL Workbench – це інтегроване середовище розробки та керування базами даних MySQL [19].

Вказані інструментальні засоби забезпечують чудову продуктивність й швидкодію системи при великій кількості запитів та об’єму даних.

Для забезпечення ефективної та надійної роботи системи, максимізації продуктивності та оптимального використання ресурсів апаратного забезпечення, були встановлені конкретні вимоги, а саме:

1. Вимоги до серверного обладнання вебсервісу: [20]
 - 1.1.оперативна пам’ять – 16 GB;
 - 1.2.жорсткий диск - 10 TB;
 - 1.3.операційна система – Windows 10 22H2;
 - 1.4.процесор з частотою від 3.5 GHz.
2. Вимоги до робочої станції користувача:
 - 2.1.гаджет з браузером, який відповідає наступним вимогам:
 - 2.1.1. Chrome 60+;
 - 2.1.2. Safari 10+ / iOS Safari 10+;
 - 2.1.3. Firefox ESR+;
 - 2.1.4. Opera.

Час реакції програми на дії користувача (маніпуляції з пристроями введення даних) не повинен перевищувати 0,5 с.

Час виконання команд меню не більше 1 с.

Відображення масивів даних за запитом не більше 2 секунд.

Доступність БД – 90% цілодобово.

Операції з’єднання з БД не більше 2 секунд.

Програма повинна нормально функціонувати при безперебійній роботі ПК.

Висновки до першого розділу.

					ІПЗ.КР.Б-121-22-ПЗ	
						16

Аналіз предметної області дозволив визначити основні вимоги та функціональні можливості вебдодатку, який спрощує пошук роботи та працівників. У процесі дослідження аналогів програмного продукту було проаналізовано різні системи та платформи, що надають подібні послуги. Проведена порівняльна характеристика за конкуруючих продуктів за визначеними параметрами.

Аналіз архітектурних підходів допоміг нам визначити подальшу архітектуру проєкту, та дав зрозуміти які можуть виникнути проблеми в майбутньому при покращенні додатка.

На основі поставлених вимог, визначених критеріїв та обраних архітектурних рішень був сформований набір технологій та засобів для розробки програмного продукту. Крім того, були визначені апаратні вимоги для кожного елемента системи з метою ефективного використання ресурсів та відповідності програмного забезпечення усім визначеним критеріям та потребам.

РОЗДІЛ 2. ПРОЄКТУВАННЯ САЙТУ ФРІЛАНСУ

2.1 Визначення варіантів використання

За ціллю візуалізації взаємозв'язків між компонентами системи, була створена діаграма варіантів використання (рис. 2.1).

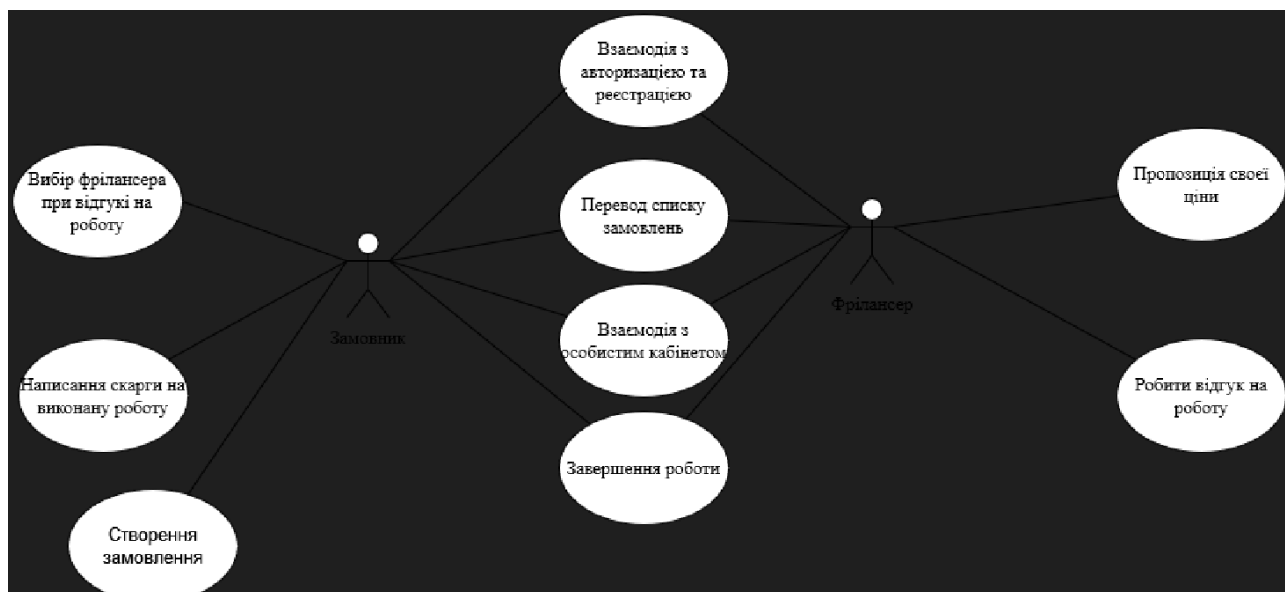


Рис. 2.1 – Діаграма варіантів використання

Опис кожного прецеденту наведено у таблиці (табл. 2.1 – 2.9).

Таблиця 2.1

Назва варіанту використання	Взаємодія з авторизацією та реєстрацією
Основні актори	Замовник, фрілансер
Короткий опис	Замовник та фрілансер мають змогу завести акаунт на сайті або увійти у свій акаунт

Таблиця 2.2

Назва варіанту використання	Взаємодія з особистим кабінетом
Основний актор	Фрілансер, замовник
Короткий опис	Виконавець та замовник мають особистий кабінет в якому можуть змінити свої дані

Таблиця 2.3

Назва варіанту використання	Перегляд списку замовлень
Основний актор	Фрілансер, замовник
Короткий опис	Фрілансер та замовник мають змогу переглядати роботи, які є на сайті

Таблиця 2.4

Назва варіанту використання	Створення замовлень
Основний актор	Замовник
Короткий опис	Замовник може створювати замовлення

Таблиця 2.5

Назва варіанту використання	Вибір фрілансера при відгуку на роботу
Основні актори	Замовник
Короткий опис	Замовник обирає який фрілансер йому найбільше підходить для цієї роботи

Таблиця 2.6

Назва варіанту використання	Запропонувати свою вартість
Основні актори	Фрілансер
Короткий опис	Якщо фрілансера зацікавила робота, але ціна не відповідає очікуванням, він може запросити свою ціну за проєкт

Таблиця 2.7

Назва варіанту використання	Робити відгук на роботу
Основні актори	Фрілансер
Короткий опис	Фрілансер може відгукнутись на роботу, яка його зацікавила

Таблиця 2.8

Назва варіанту використання	Завершення роботи
Основні актори	Замовник, фрілансер
Короткий опис	Замовник та фрілансер мають змогу завершити роботу

Таблиця 2.9

Назва варіанту використання	Написання скарги на виконану роботу
Основні актори	Замовник
Короткий опис	Замовник має змогу написати скаргу на фрілансера

Головна мета створення фріланс-сайту полягає в полегшенні процесу пошуку робітників і роботи для користувачів. Для того, щоб наш сайт задовольняв потреби користувачів, були визначені відповідні вимоги до нашого продукту.

Бізнес-вимоги:

1. Основні цілі: сайт створюється з метою спрощення пошуку роботи або робітників.
2. Представлення проєкту: проєкт буде реалізовано у вигляді вебсайту.
3. Реєстрація користувачів: забезпечення можливості реєстрації для фрілансерів та замовників, включаючи аутентифікацію та управління обліковими записами.

4. Розміщення та пошук проєктів: забезпечення функціонала для замовників розміщувати проєкти, описувати вимоги та критерії, а також для фрілансерів шукати та відгукатися на ці проєкти.

5. Скарги та оцінки: можливість користувачам залишати скарги та оцінки проєктів.

Вимоги користувачів:

1. замовники:

1.1. перегляд інших замовлень задля розуміння орієнтованої ціни за проєкт;

1.2. можливість обирати фрілансера за визначеними критеріями (рейтинг, повідомлення, ціна) ;

1.3. можливість написання скарги на фрілансера за виконану роботу;

1.4. можливість змінити свій пароль;

1.5. можливість змінити дані свого акаунту, окрім електронної пошти;

1.6. можливість авторизуватись у свій акаунт;

1.7. можливість створити акаунт;

1.8. можливість зробити оплату послуги по qr коду.

2. Фрілансери:

2.1. можливість залишити заявку на замовлення;

2.2. можливість надсилати виконане завдання замовнику;

2.3. можливість задавати свою ціну замовлення, якщо не влаштовує запропонована;

2.4. можливість пошуку роботи за тегами які нас цікавлять;

2.5. редагування свого профілю та зміна паролю;

2.6. можливість авторизуватись у свій акаунт;

2.7. можливість створити акаунт.

Нефункціональні вимоги:

1. сприйняття:

1.1. інтерфейс повинен бути зручним та інтуїтивно зрозумілим.

2. Надійність:

2.1. максимальна норма помилок та дефектів в роботі сайту – 1 помилка на 1000 запитів користувача.

3. Продуктивність:

3.1. Система повинна працювати без перебоїв при одночасному використанні понад 200 користувачів.

За основу об'єктно-орієнтованої структури серверу було обрано архітектуру на основі фреймворка NestJS, яка поєднує різні аспекти FRP, OOP і FP та розділення відповідальностей. Основними структурними елементами серверу є: моделі-сутності, API-класи, контролери та сервіси.

Діаграма класів, яка описує моделі наведена на (рис. 2.2).

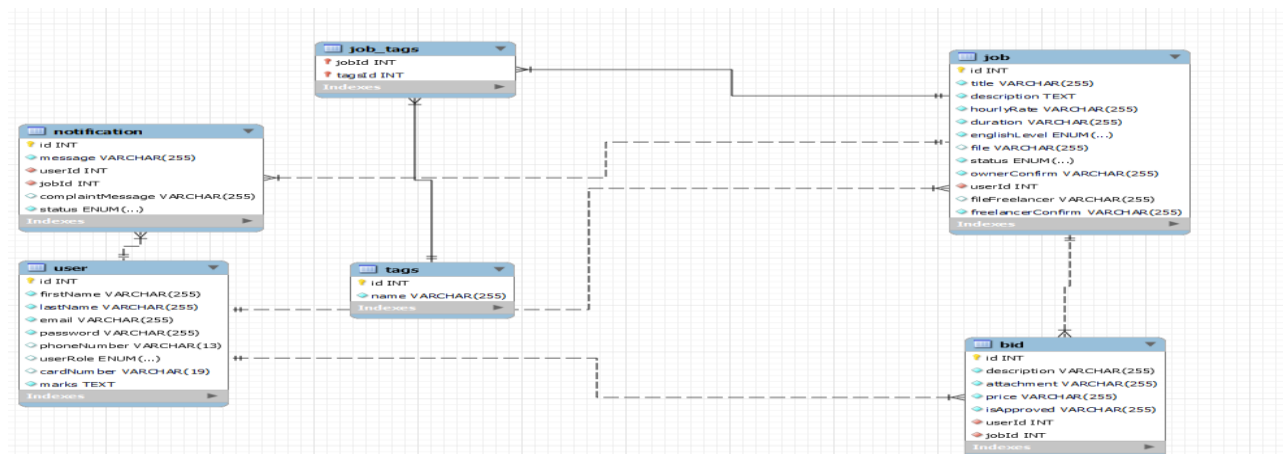


Рис. 2.2 – Діаграма класів

User – модель-сутності, яка являє собою користувача. Містить наступні поля:

1. id (унікальне значення, тип int) – унікальний ідентифікатор користувача;
2. firstName (тип varchar) – ім'я користувача;
3. lastName (тип varchar) – прізвище користувача;
4. email (унікальне значення, тип varchar) – електронна адреса;
5. password (тип varchar) – пароль користувача;
6. userRole (тип enum) – роль користувача. Може мати такі значення: employer – замовник, freelancer – фрілансер;
7. cardNumber (тип varchar) – номер картки користувача;

8. `phoneNumber` (унікальне значення, тип `varchar`) – номер телефону користувача;
9. `marks` (тип `text`) – оцінки які були отримані за виконану роботу.

`Notification` – модель-сутності, яка являє собою сповіщення про виконану роботу. Містить наступні поля:

1. `id` (унікальне значення, тип `int`) – унікальний ідентифікатор сповіщення;
2. `message` (тип `varchar`) – повідомлення;
3. `status` (тип `enum`) – статус сповіщення. Може мати такі значення `approved` – підтверджений, `waiting` – очікує, `rejected` – відхилено, `complaint` – скарга;
4. `complaintMessage` (тип `varchar`) – повідомлення скарги;
5. `jobId` (тип `int`) – ідентифікатор роботи;
6. `userId` (тип `int`) – ідентифікатор користувача який відправив це сповіщення.

`Tags` – модель-сутності, яка являє собою теги для робіт. Містить наступні поля:

1. `id` (унікальне значення, тип `int`) – унікальний ідентифікатор тега;
2. `name` (унікальне значення, тип `varchar`) – назва тега.

`Job` – модель-сутності, яка являє собою роботу. Містить наступні поля:

1. `id` (унікальне значення, тип `int`) – унікальний ідентифікатор роботи;
2. `title` (унікальне значення, тип `varchar`) – заголовок;
3. `description` (тип `varchar`) – опис задачі яку в майбутньому буде виконувати фрілансер;
4. `hourlyRate` (тип `varchar`) – почасова оплата;
5. `duration` (тип `varchar`) – час який потрібний на виконання роботи;
6. `englishLevel` (тип `enum`) – рівень володіння англійського щоб, впоратись з задачею. Може мати такі значення: `advanced` – продвинутий, `intermediate` – середній, `elementary` – низький;
7. `file` (тип `varchar`) – файл замовника;

8. status (тип varchar) – статус. Може мати такі значення: notStarted – не розпочата, atProcess – виконується, finished - завершена;
9. ownerConfirm (тип boolean) – чи підтвердив замовник що робота виконана;
- 10.userId (тип int) – ідентифікатор користувача який створив цю роботу;
- 11.fileFreelancer (тип varchar) – файл фрілансера який виконав цю роботу;
- 12.freelancerConfirm (тип boolean) – чи підтвердив фрілансер що робота виконана.

Job_tags – модель-сутності, яка являє собою допоміжну таблицю для робіт та тегів. Містить наступні поля:

1. jobId (тип int) – ідентифікатор роботи;
2. tagId (тип int) – ідентифікатор тега.

Bid – модель-сутності, яка являє собою таблицю для відгуків на роботи.

Містить наступні поля:

1. jobId (тип int) – ідентифікатор роботи;
2. userId (тип int) – ідентифікатор користувача;
3. id (унікальне значення, тип int) - унікальний ідентифікатор відгука;
4. description (тип varchar) – чому саме вас повинні взяти на роботу;
5. price (тип varchar) – ціна за яку ви готові виконати роботу;
6. isApproved (тип boolean) – чи прийняли вашу кандидатуру.

Діаграма класів, яка описує сервіси наведена на (рис. 2.3).

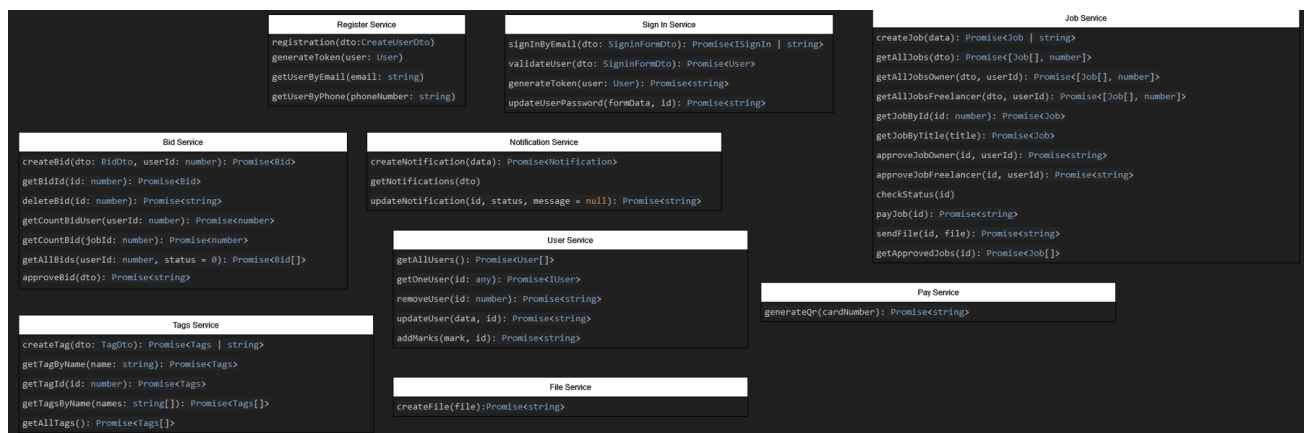


Рис. 2.3 – Діаграма класів провайдерів

User service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю User.

Методи:

1. getAllUsers(): Promise<User[]> - повертає список користувачів;
2. getOneUser(id: any): Promise<IUser> - повертає користувача за ідентифікатором;
3. removeUser(id: number): Promise<string> - видаляє користувача, повертає повідомлення про успішне видалення;
4. updateUser(data, id): Promise<string> - оновлює користувача за вхідними даними та повертає повідомлення про успіх;
5. addMarks(mark, id): Promise<string> - за вхідними даними додає оцінку за роботу.

Tags service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю Tags.

Методи:

1. createTag(dto: TagDto): Promise<Tags | string> - створює тег за вхідними даними;
2. getTagByName(name: string): Promise<Tags> - пошук тегу за назвою;
3. getTagId(id: number): Promise<Tags> - пошук тегу за ідентифікатором;
4. getTagsByName(names: string[]): Promise<Tags[]> – повертає масив тегів за вхідними даними;
5. getAllTags(): Promise<Tags[]> – повертає список тегів.

Notification service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю Notification.

Методи:

1. createNotification(data): Promise<Notification> - створює сповіщення;
2. getNotifications(dto) - повертає список сповіщень за вхідними параметрами з додатковою інформацією для пагінації;
3. updateNotification(id, status, message = null): Promise<string> – оновлює дані сповіщення та повертає відповідне повідомлення;

Job service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю Job.

Методи:

1. createJob(data): Promise<Job | string> - створює роботу та повертає її;
2. getAllJobs(dto): Promise<[Job[], number]> - повертає список робіт за вхідними параметрами з додатковою інформацією для пагінації;
3. getAllJobsOwner(dto, userId): Promise<[Job[], number]> – повертає список робіт які належать замовнику за вхідними параметрами з додатковою інформацією для пагінації
getAllJobsFreelancer(dto, userId): Promise<[Job[], number]> - повертає список робіт для фрілансера за вхідними параметрами з додатковою інформацією для пагінації;
4. getJobById(id: number): Promise<Job> - повертає роботу за ідентифікатором;
5. getJobByTitle(title): Promise<Job> - повертає роботу за заголовком;
6. approveJobOwner(id, userId): Promise<string> - підтвердження того що робота була виконана фрілансером та замовник отримав те що бажав;
7. approveJobFreelancer(id, userId): Promise<string>- підтвердження того що робота була виконана фрілансером;
8. checkStatus(id)- перевірка статусу роботи;
9. payJob(id): Promise<string> - отримання qr коду для оплати;
10. sendFile(id, file): Promise<string> - відправка файлу;
11. getApprovedJobs(id): Promise<Job[]> - отримання підтверджених робіт.

Bid service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю Bid.

Методи:

1. createBid(dto: BidDto, userId: number): Promise<Bid> - створює ставку на роботу та повертає її;
2. getBidId(id: number): Promise<Bid> - повертає ставку за ідентифікатором;

3. deleteBid(id: number): Promise<string> – видаляє ставку за ідентифікатором;
4. getCountBidUser(userId: number): Promise<number> - повертає кількість ставок на роботи замовників;
5. getCountBid(jobId: number): Promise<number> - повертає кількість ставок на роботу;
6. getAllBids(userId: number, status = 0): Promise<Bid[]> - повертає масив ставок;
7. approveBid(dto): Promise<string> - замовник підтверджує фрілансера на роботу.

SignIn service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю User.

Методи:

1. signInByEmail(dto: SigninFormDto): Promise<ISignIn | string> - виконує вхід у систему під певним користувачем;
2. validateUser(dto: SigninFormDto): Promise<User> - перевірка на правильність даних;
3. generateToken(user: User): Promise<string> – генерація jwt токenu;
4. updateUserPassword(formData, id): Promise<string> - зміна паролю.

Registration service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю User.

Методи:

1. registration(dto: CreateUserDto) – реєстрація користувача в системі;
2. generateToken(user: User) – генерація jwt токenu;
3. getUserByEmail(email: string) – пошук користувача по email;
4. getUserByPhone(phoneNumber: string) – пошук користувача по номеру телефону.

Files service – клас-сервіс, який містить операції з базою даних для виконання запису файлу з сутністю Job.

Методи:

					ІПЗ.КР.Б-121-22-ПЗ	
						27

1. `createFile(file):Promise<string>` - записує файл в статистику проєкту.

Pay service – клас-сервіс.

Методи:

2. `generateQr(cardNumber): Promise<string>` - генерує qr код по номеру картки.

Діаграма класів, яка описує API-класи наведена на рис. 2.4.

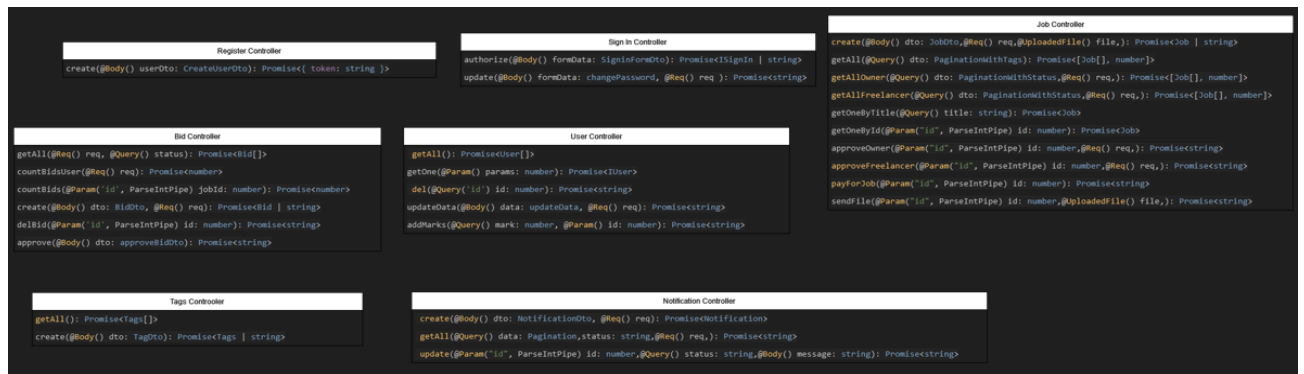


Рис. 2.4 – Діаграма класів API-класів

Register controller – клас, який містить методи операцій обробки даних та роботи з сутністю User.

Методи:

1. `create(@Body() userDto: CreateUserDto): Promise<{ token: string }>` - реєструє користувача в базі даних.

Bid controller – клас, який містить методи операцій обробки даних та роботи з сутністю Bid.

Методи:

1. `getAll(@Req() req, @Query() status): Promise<Bid[]>` - повертає список ставок за вхідними параметрами;
2. `countBidsUser(@Req() req): Promise<number>` – повертає кількість ставок на роботу для конкретного користувача;
3. `countBids(@Param('id', ParseIntPipe) jobId: number): Promise<number>` – повертає кількість ставок на роботу;
4. `create(@Body() dto: BidDto, @Req() req): Promise<Bid | string>` – створює ставку;

5. `delBid(@Param('id', ParseIntPipe) id: number): Promise<string>` – видаляє ставку;
6. `approve(@Body() dto: approveBidDto): Promise<string>` - замовник підтверджує фрілансера на роботу.

Tags controller – клас, який містить методи операцій обробки даних та роботи з сутністю Tag.

Методи:

1. `getAll(): Promise<Tags[]>` - повертає список тегів;
2. `create(@Body() dto: TagDto): Promise<Tags | string>` – створює тег за вхідними даними.

SignIn controller – клас, який містить методи операцій обробки даних та роботи з сутністю User.

Методи:

1. `authorize(@Body() formData: SigninFormDto): Promise<ISignIn | string>` - авторизація користувача в системі;
2. `update(@Body() formData: changePassword, @Req() req): Promise<string>` – зміна пароллю.

User controller – клас, який містить методи операцій обробки даних та роботи з сутністю User.

Методи:

1. `getAll(): Promise<User[]>` - повертає список користувачів;
2. `getOne(@Param() params: number): Promise<IUser>` – повертає користувача за ідентифікатором;
3. `del(@Query('id') id: number): Promise<string>` - видаляє користувача;
4. `updateData(@Body() data: updateData, @Req() req): Promise<string>` - оновлення даних користувача;
5. `addMarks(@Query() mark: number, @Param() id: number): Promise<string>` - додавання оцінок користувачу.

Notification controller – клас, який містить методи операцій обробки даних та роботи з сутністю Tag.

Методи:

1. create(@Body() dto: NotificationDto, @Req() req): Promise<Notification>
- створення сповіщення про готовність роботи;
2. getAll(@Query() data: Pagination, status: string, @Req() req): Promise<string> – отримання всі сповіщень користувача;
3. update(@Param("id", ParseIntPipe) id: number, @Query() status: string, @Body() message: string): Promise<string> - зміна статусу сповіщення.

Job controller – клас, який містить методи операцій обробки даних та роботи з сутністю Job.

Методи:

1. create(@Body() dto: JobDto, @Req() req, @UploadedFile() file): Promise<Job | string> - створення роботи;
2. getAll(@Query() dto: PaginationWithTags): Promise<[Job[], number]> – повертає список робіт з параметрами для пагінації;
3. getAllOwner(@Query() dto: PaginationWithStatus, @Req() req): Promise<[Job[], number]> - отримання робіт які створив користувач;
4. getAllFreelancer(@Query() dto: PaginationWithStatus, @Req() req): Promise<[Job[], number]> - отримання робіт які виконав або виконуються фрілансером;
5. getOneByTitle(@Query() title: string): Promise<Job> - пошук роботи за заголовком;
6. getOneById(@Param("id", ParseIntPipe) id: number): Promise<Job> - пошук роботи за ідентифікатором;
7. approveOwner(@Param("id", ParseIntPipe) id: number, @Req() req): Promise<string> - підтвердження того що робота була виконання зі сторони замовника;
8. approveFreelancer(@Param("id", ParseIntPipe) id: number, @Req() req): Promise<string>- підтвердження того що робота була виконання зі сторони фрілансера;

9. payForJob(@Param("id", ParseIntPipe) id: number): Promise<string> - оплата за роботу;

10. sendFile(@Param("id", ParseIntPipe) id: number,@UploadedFile() file,): Promise<string> - відправка файлу зі сторони замовника.

2.2 Розробка бази даних системи

Для збереження сутностей та взаємодії з даними сайту фрілансу була розроблена база даних, таблиці якої віддзеркалюють моделі-сутності за допомогою ORM. Для нашого вебсервісу було обрано typeORM. Відповідно до описаної архітектури, взаємодія між вебсервером та БД відбувається у контролерах.

База даних сайту складається з 7 таблиць:

1. user – таблиця, яка зберігає дані про користувачів системи, та має зв'язок 1 до багатьох з таблицею job;
2. notification – таблиця, яка зберігає дані про сповіщення, та має зв'язок багато до 1 з таблицею user, а також багато до 1 з таблицею job;
3. job – таблиця, яка зберігає дані про роботи, та має зв'язок багато до з таблицею user, а також багато до багатьох з таблицею tags;
4. bid – таблиця, яка зберігає дані про ставки, та має зв'язок багато до 1 з таблицею job;
5. tags – таблиця, яка зберігає дані про теги, та має зв'язок багато до багатьох з таблицею job;
6. job_tags – проміжна таблиця [21], яка зберігає дані про теги які належать роботам.

Нижче наведений опис кожної з наведених таблиць бази даних (див. табл. 2.10 – 2.15)

Таблиця 2.10

Users		
Поле	Тип	Призначення
Id	Uniqueidentifier, PK	Унікальний ідентифікатор
firstName	VARCHAR(255)	Ім'я
lastName	VARCHAR(255)	Прізвище

Продовження таблиці 2.10

email	VARCHAR(255)	Email адреса
password	VARCHAR(255)	Пароль
phoneNumber	VARCHAR(13)	Номер телефону
cardNumber	VARCHAR(19)	Номер картки
marks	TEXT	Оцінки
userRole	ENUM('Freelancer', 'Employer', 'Admin')	Роль користувача

Таблиця 2.11

Job		
Поле	Тип	Призначення
id	Uniqueidentifier, PK	Унікальний ідентифікатор
title	VARCHAR(255)	Заголовок
description	TEXT	Опис
hourlyRate	VARCHAR(255)	По часова ставка
duration	VARCHAR(255)	Тривалість
englishLevel	ENUM('Advanced', 'Intermediate', 'Elementary')	Рівень англійського
status	ENUM('notStarted', 'atProcess', 'finished')	Статус
ownerConfirm	VARCHAR(255)	Підтвердження замовника
userId	FK, int	Ідентифікатор замовника
file	VARCHAR(255)	Назва збереженого файлу у системі

Продовження таблиці 2.11

fileFreelancer	VARCHAR(255)	Назва збереженого файлу у системі
freelancerConfirm	VARCHAR(255)	Підтвердження фрілансера

Таблиця 2.12

Notification		
Поле	Тип	Призначення
Id	Uniqueidentifier, PK	Унікальний ідентифікатор
message	VARCHAR(255)	Повідомлення
userId	INT, FK	Ідентифікатор фрілансера
jobId	INT, FK	Ідентифікатор роботи
complaintMessage	VARCHAR(255)	Повідомлення скарги
status	ENUM('waiting', 'rejected', 'approved', 'complaint')	Статус

Таблиця 2.13

Bid		
Поле	Тип	Призначення
Id	Uniqueidentifier, PK	Унікальний ідентифікатор
description	VARCHAR(255)	Опис
price	VARCHAR(255)	Ціна запропонована фрілансером
isApproved	VARCHAR(255)	Статус ставки

Продовження таблиці 2.13

userId	INT, FK	Ідентифікатор фрілансера
jobId	INT, FK	Ідентифікатор роботи

Таблиця 2.14

Tags		
Поле	Тип	Призначення
jobId	INT, FK	Ідентифікатор роботи
tagsId	INT, FK	Ідентифікатор тегу

Таблиця 2.15

Job_tags		
Поле	Тип	Призначення
Id	Uniqueidentifier, PK	Унікальний ідентифікатор
name	VARCHAR(255)	Назва тегу

2.3 Проєктування та реалізація алгоритмів роботи системи

Сайт фрілансу побудований на компонентній архітектурі, тому важливо ефективно реалізувати взаємодію між компонентами та іншими структурами даних. На рисунку 2.10 наведено діаграму активності [22], яка наочно представляє взаємодію між клієнтським додатком та сервером, складається вона з наступних кроків:

1. гість заходить на головну сторінку;
2. після входу або реєстрації робиться відповідний запит на сервер;
3. якщо є помилки у введених даних – відображається повідомлення про помилку;
4. дані перевіряються на стороні серверу й повертається помилка у разі їх неправильності;

5. після успішної авторизації або реєстрації, користувач може створювати або шукати роботи;
6. система виконує запити до БД та повертає дані;
7. результат виконання операції відображається користувачу.

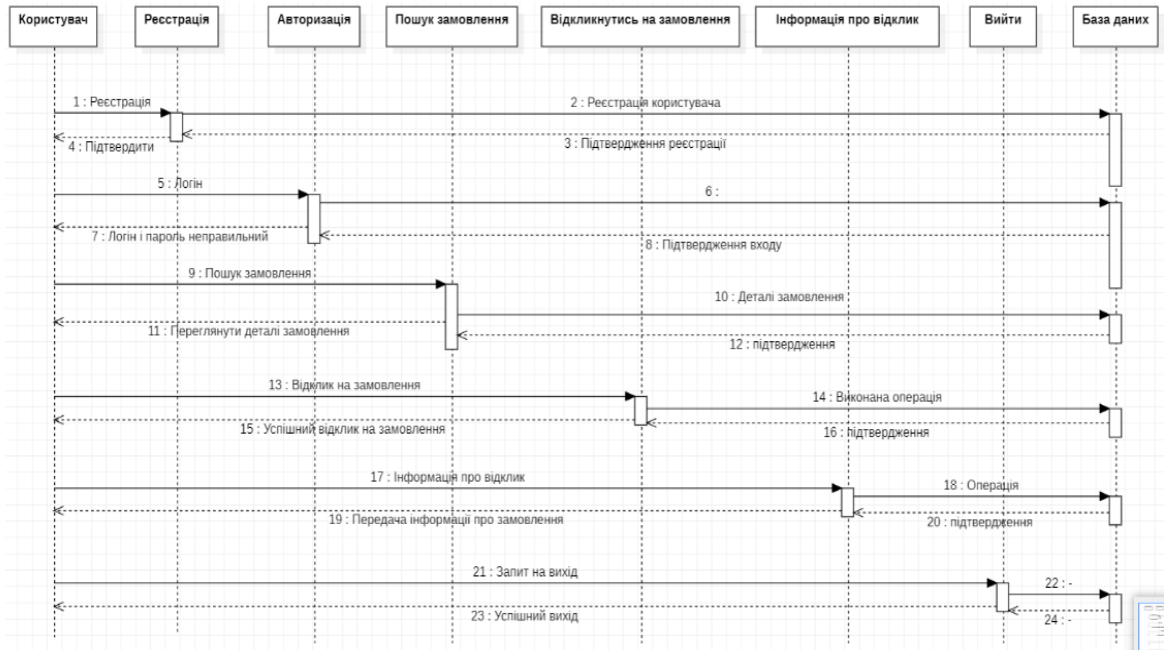


Рис. 2.10 – Діаграма відгуку фрілансера на роботу

Процес створення роботи складається з двох етапів: створення роботи та перевірки на унікальність. Спочатку замовник має створити роботу з відповідною інформацією. Перевірка роботи на унікальність відбувається по заголовку під час обробки запиту на сервері (рис. 2.11).

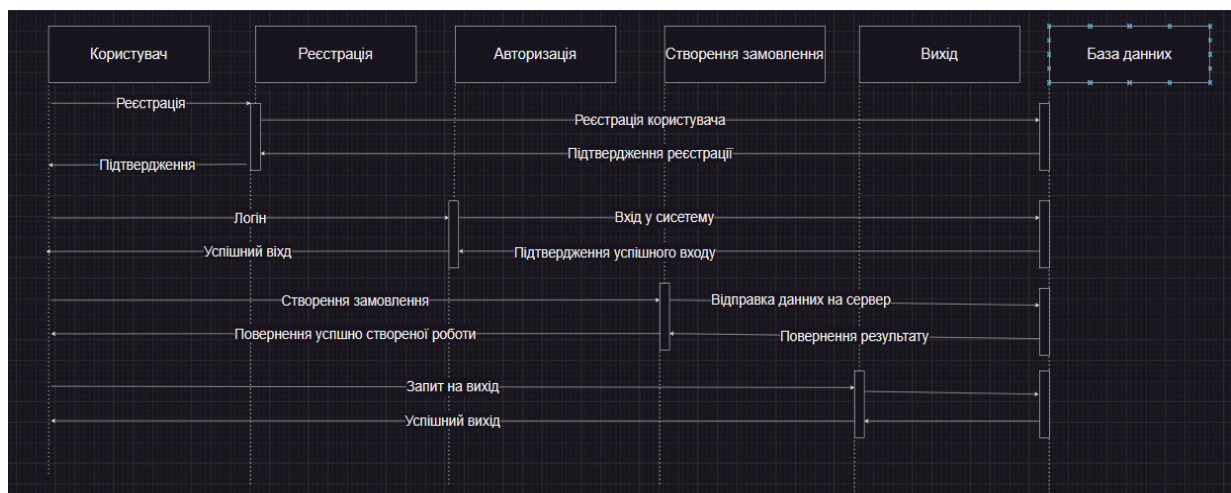


Рис. 2.11 – Діаграма послідовностей створення замовлення

Висновки до другого розділу

У процесі проєктування вебсайту фрілансу, були аналізовані всі можливі варіанти використання програмного продукту відповідно до вимог, що стосуються програмного забезпечення. Отримані результати були використані для створення об'єктно-орієнтованої структури системи, включаючи моделювання взаємодії між складовими частинами відповідно до обраної архітектури.

Крім того, була розроблена база даних на основі сутностей та даних, які використовуються у системі. Зв'язки між таблицями [23] бази даних та моделями-сутностями були налагоджені за допомогою ORM, що дозволило забезпечити зручний доступ та взаємодію з даними у системі.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБДОДАТКОМ ФРІЛАН- СУ

3.1 Структура інтерфейсу вебдодатку

Взаємодія користувачів з системою відбувається за допомогою браузера [24]. При переході на сторінку сайту, користувач бачить перед собою початкову сторінку (рис. 3.1).

[illegible]

Рис. 3.1 – Початкова сторінка

Для отримання додатково повноцінного функціоналу, користувач повинен авторизуватись. Якщо відвідувач не має облікового запису, необхідно пройти етап реєстрації на вебресурсі [25] (рис. 3.2 - 3.3).

Рис. 3.2 - Форма авторизації

* Ваша роль?: Фрілансер Замовник

* Ім'я ?:

* Прізвище ?:

* Номер телефону:

* E-mail: zzombi1@rambler.ua

* Пароль: ●●●●●●●●  

* Повторіть пароль: 

☐ Я прочитав [правила](#)

Зареєструватись

Рис. 3.3 - Форма авторизації

На головній сторінці є пошук за заголовком [26] (рис. 3.4).

[головна сторінка](#) / детальний перегляд роботи

← Телеграм бот

5 \$	2 дні	Elementary	0
Оплата за годину	Час на виконання	Рівень англійського	Кількість відгуків

Я маю телеграм бот з текстами пісень Тексти пісень розташовані за алфавітом, а там вже їх назви, все працює стандартними кнопками в боті. Проблема в тому, що якщо пісень багато, то складно шукати одну якусь Мені потрібно щоб людина могла написати назву або частину тексту пісні і їй видавало її, навіть якщо буде помилка в написанні, то бот повинен видавати типу "можливо ви мали на увазі це" і вибирає максимально підходящий варіант І потрібно щоб у майбутньому я міг додавати нові тексти пісень у бот

Telegram bot

Рис. 3.4 - Результат пошуку заняття за назвою

Для того, щоб отримати детальну інформацію про роботу користувач повинен натиснути на 'побачити більше' (рис. 3.5-3.6).

					ІПЗ.КР.Б-121-22-ПЗ	
						38

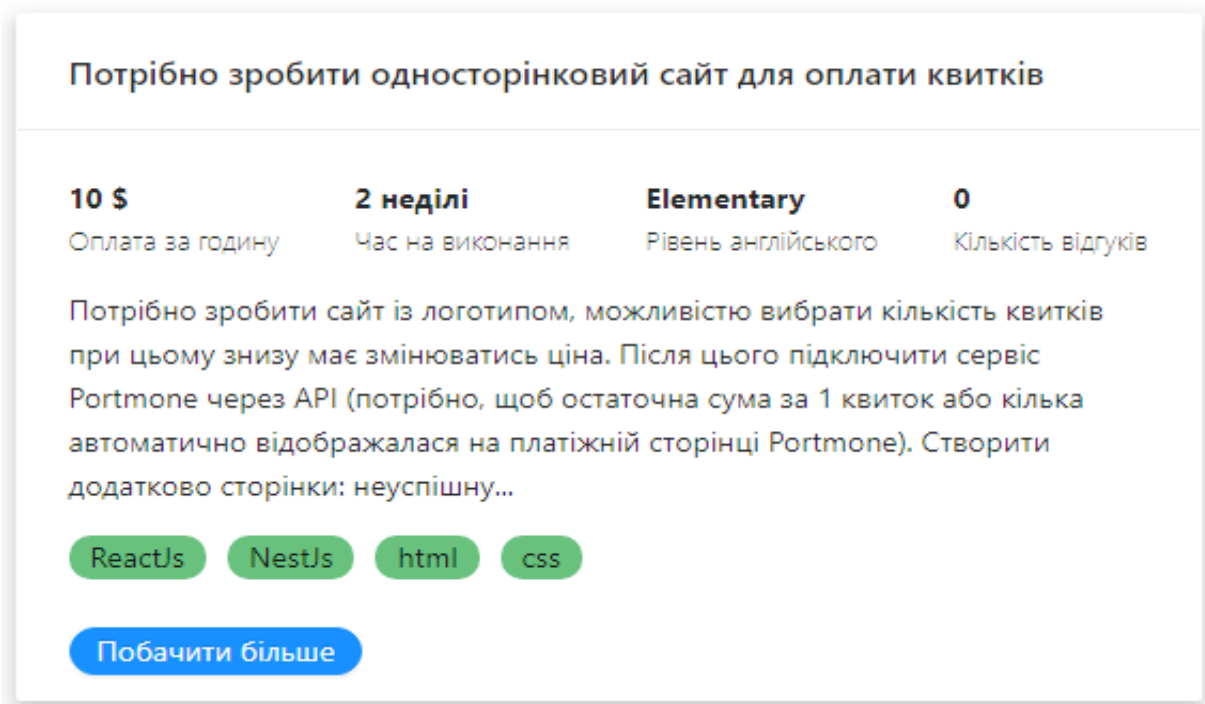


Рис. 3.5 – Картки замовлення

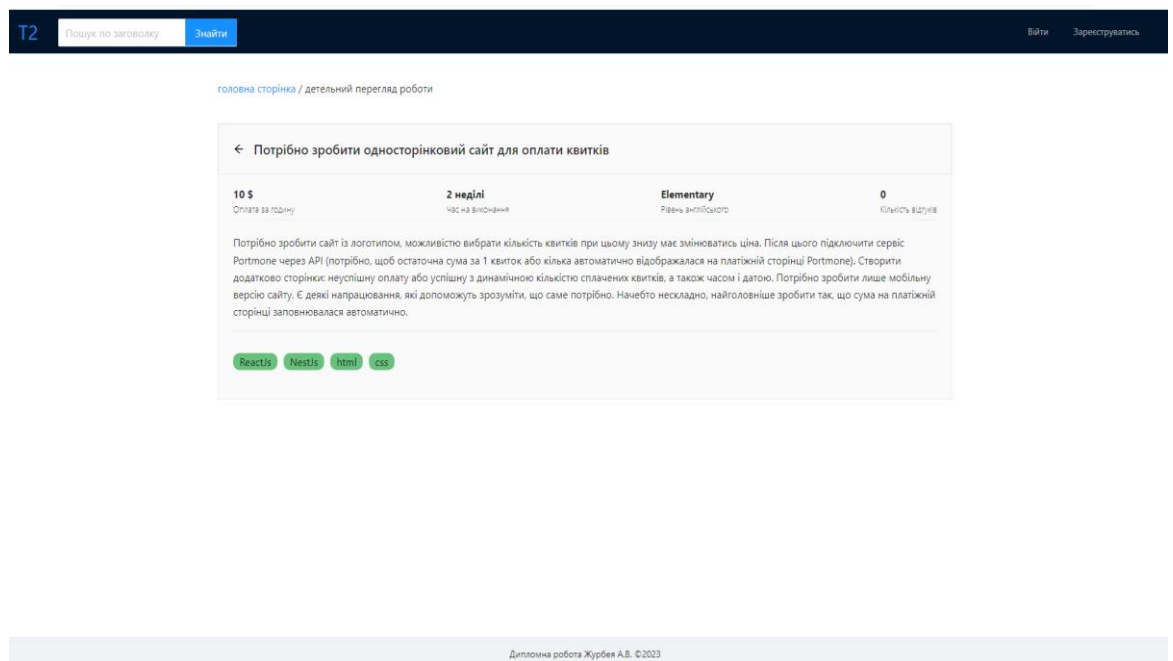


Рис. 3.6 – результат натискання на кнопку

Також на головній сторінці є фільтр по тегам (рис. 3.7-3.8).

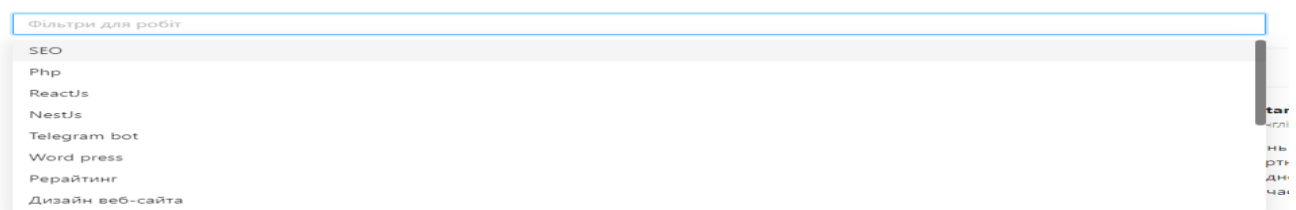


Рис. 3.7 – Вигляд фільтра

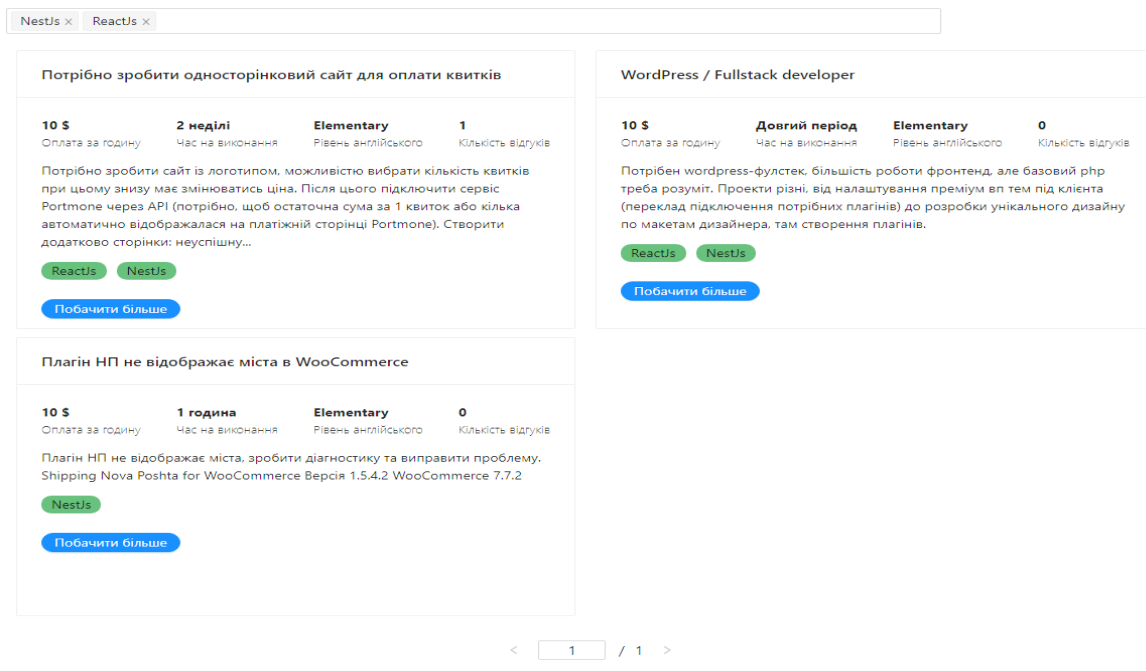


Рис. 3.8 – Результат використання фільтрів

В правому верхньому кутку в нас є вибачаюче меню [27] в котрому контент залежить від ролі користувача (рис. 3.9-3.10).

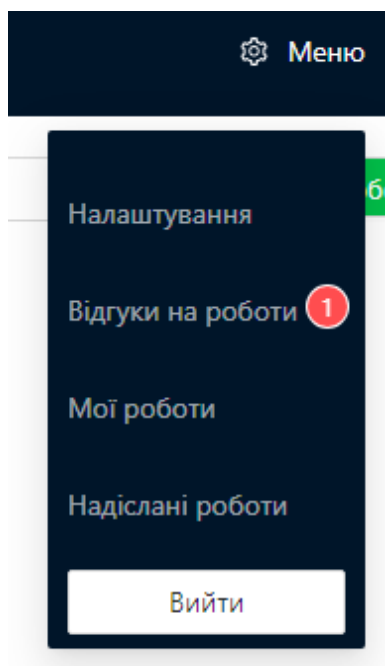


Рис. 3.9 – Меню замовника

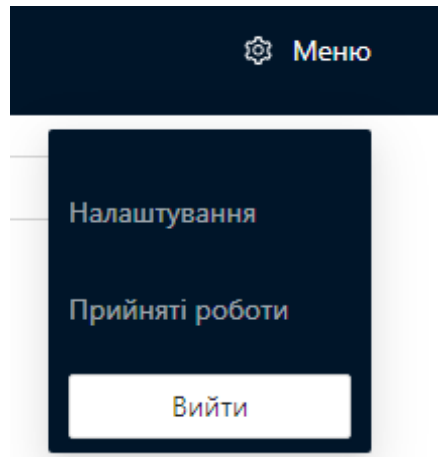


Рис. 3.10 – Меню фрілансера

Біля фільтрів [28] у замовників розположена кнопка створення при натисканні на яку буде перенесено на відповідну сторінку (рис. 3.11-3.12)



Рис. 3.11 – Вигляд кнопки створення замовлення

Рис. 3.12 – Вигляд сторінки створення замовлення

Для ролі фрілансера існує додатковий інтерактивний елемент ‘зробити ставку’ на сторінці перегляду замовлення, при натисканні на яку, відобразиться модальне вікно з формою для заповнення (рис. 3.13-3.14).

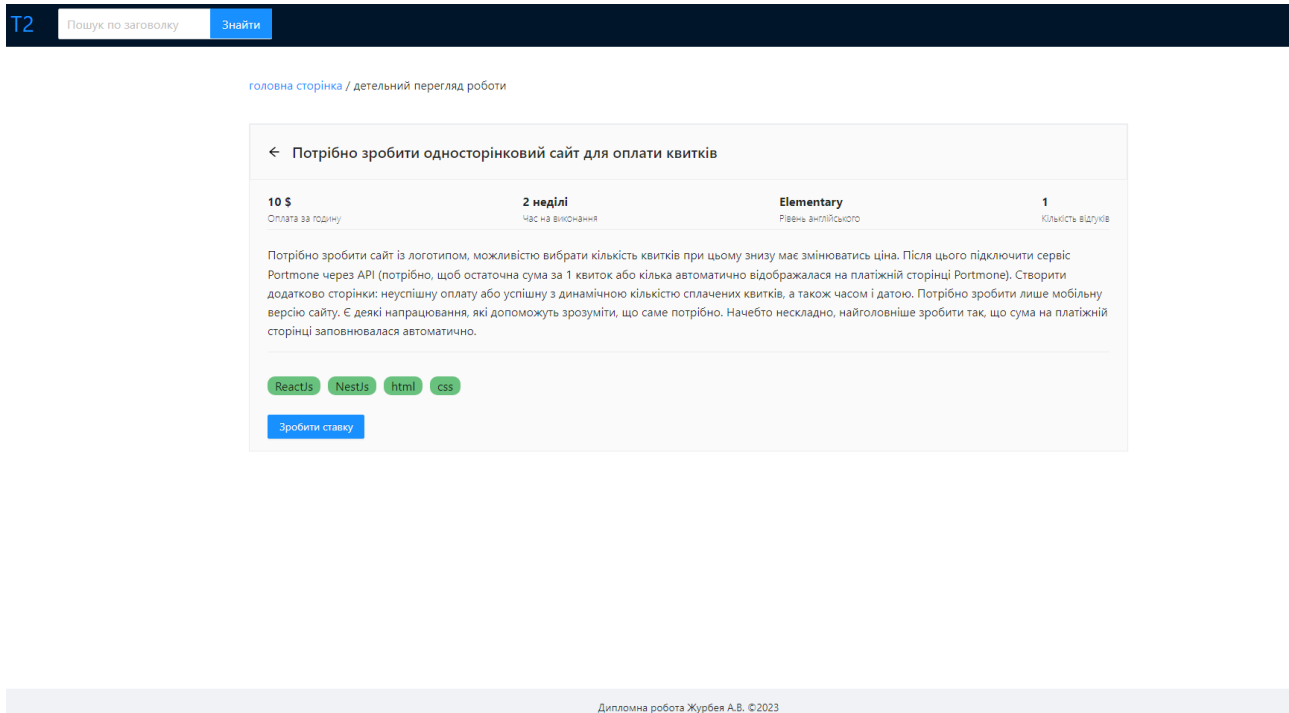


Рис. 3.13 – Вигляд роботи у фрілансера

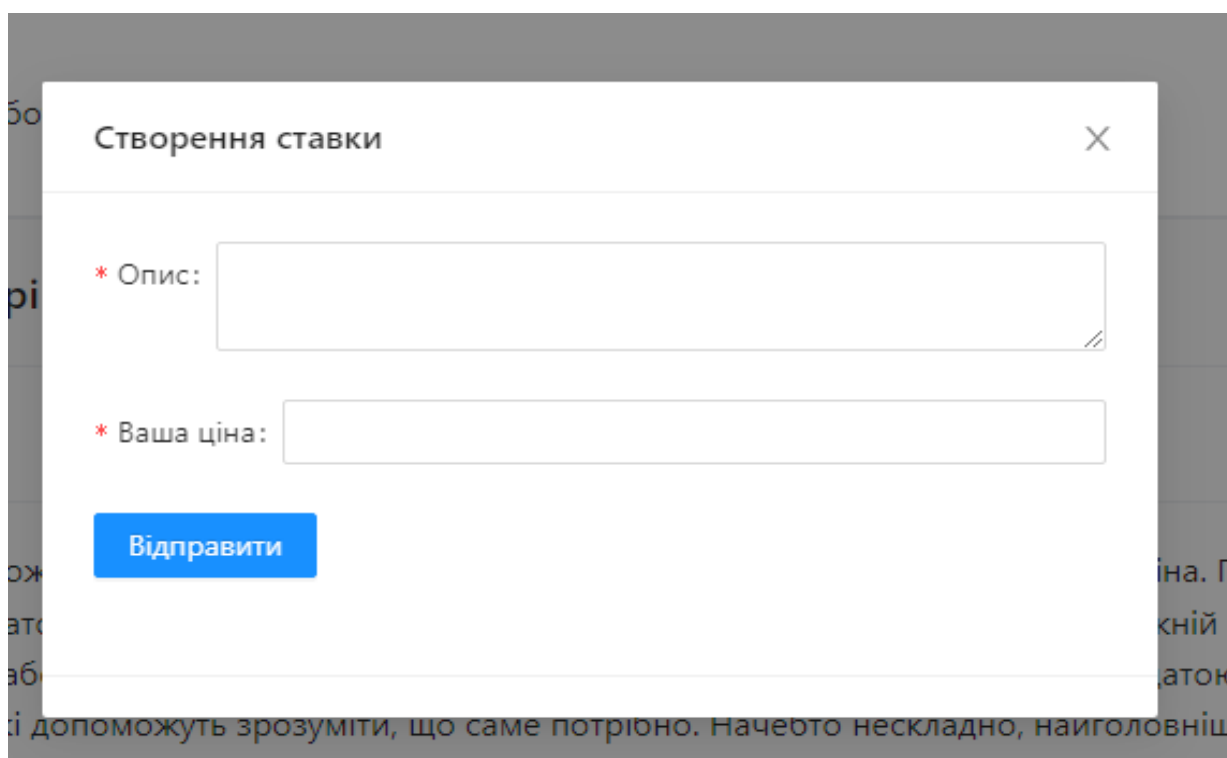


Рис. 3.14 – Вигляд модального вікна

Якщо в замовника є відгуки на роботи, то при розгортанні меню клієнт отримає нотифікацію у вигляді компоненту, зазначеного на рисунку (рис. 3.15) разом з ‘відгуки на роботи’ [29].

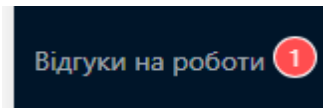


Рис. 3.15 – Нотифікація користувача

Після переходу на сторінку ‘відгуки на роботи’ користувач побачить таблицю з відгуками (рис. 3.16).

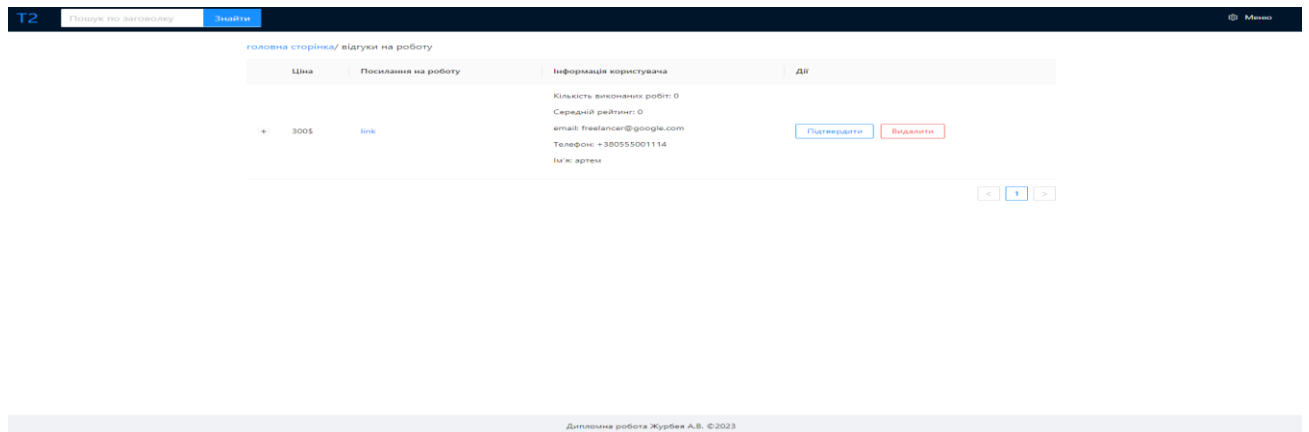


Рис. 3.16 – Сторінка відкликів на роботи

Зліва від ціни є інтерактивний елемент ‘+’, при натисканні на який буде відображено повідомлення, яке написав фрілансер при відклику на замовлення (рис. 3.17).

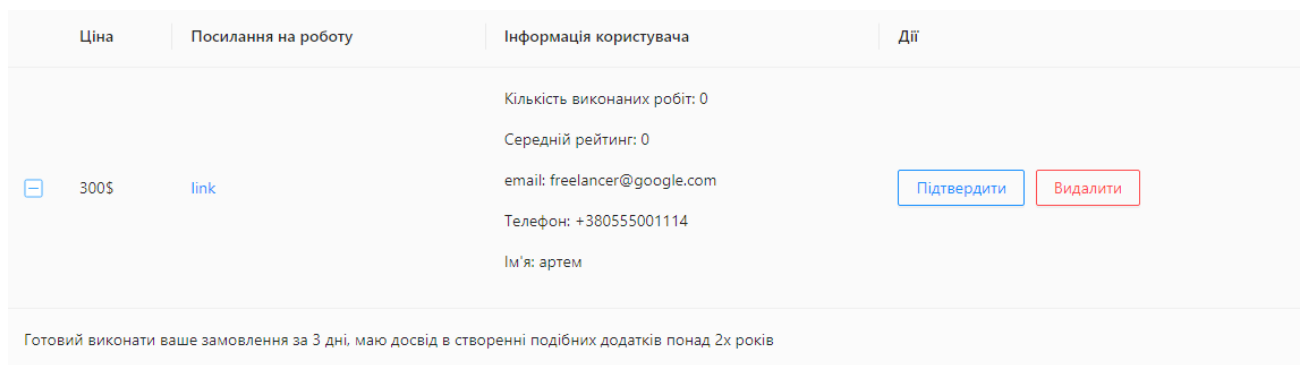


Рис. 3.17 - Вигляд ставки з відкритим повідомленням

Також на сайті є сповіщення про успішне виконання операції (рис 3.18-3.19).



Рис. 3.18 – Запит виконався успішно

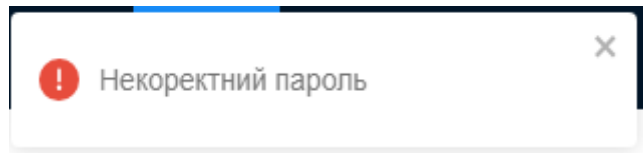


Рис. 3.19 – Запит виконався неуспішно

Якщо користувач побажає відредагувати особисті дані або пароль, можливо відвідати сторінку налаштувань (рис. 3.20-3.21)

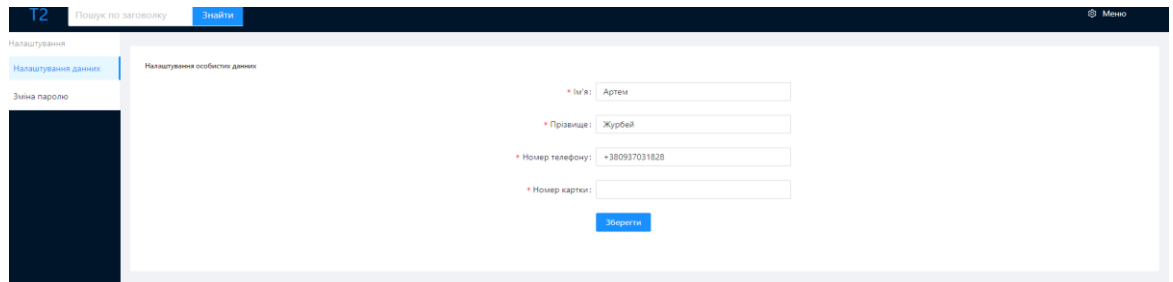


Рис. 3.20 – Сторінка налаштувань, вкладка ‘налаштування особистих даних’

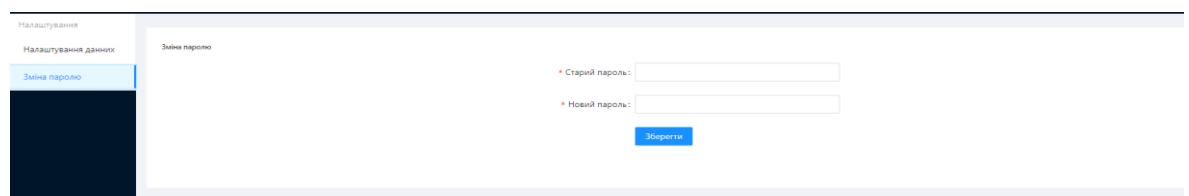


Рис. 3.21 – Сторінка налаштувань, вкладка ‘зміна паролю’

На сторінці ‘мої роботи’ замовник може переглянути особисті замовлення, а також відфільтрувати за бажанням (рис. 3.22-3.24)

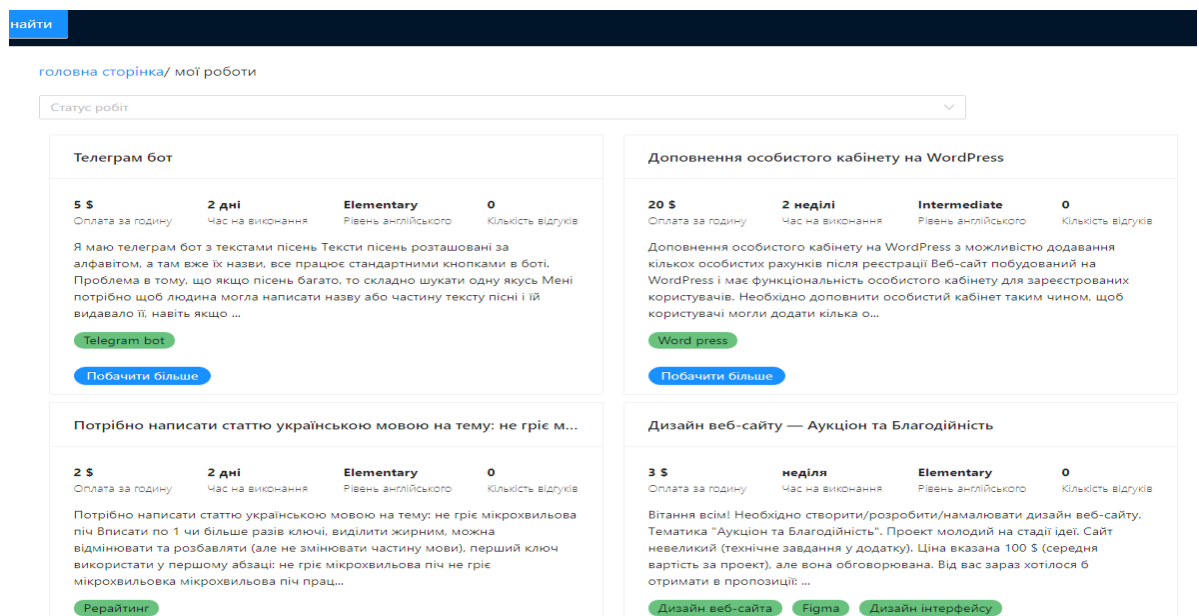


Рис. 3.22 – Сторінка робіт замовника

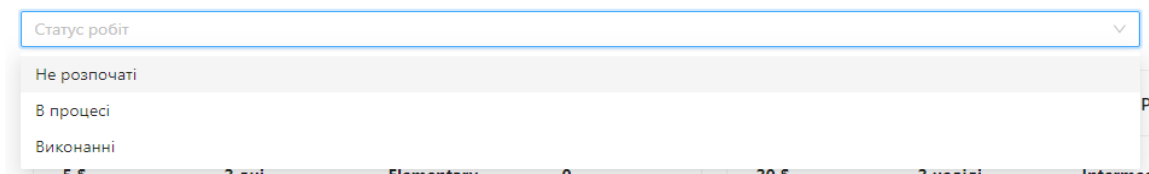


Рис. 3.23 – Фільтри по яким можна відсортувати роботи

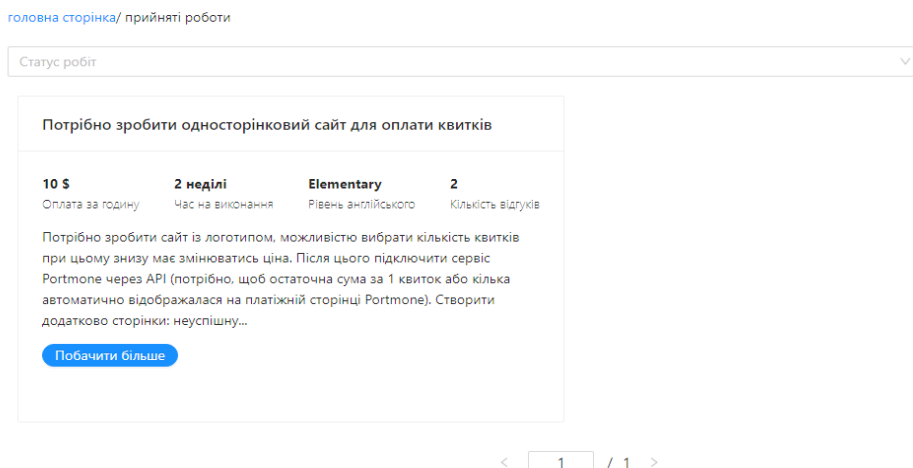


Робіт за таким статусом не знайдено

< 1 / 0 >

Рис. 3.24 – Повідомлення якщо роботи не знайдено за фільтрами

На сторінці ‘прийняті роботи’ фрілансер може переглянути список робіт на які його взяли або котрі він колись виконував (рис.3.25-3.26)



< 1 / 1 >

Рис. 3.25 – Сторінка прийнятих робіт

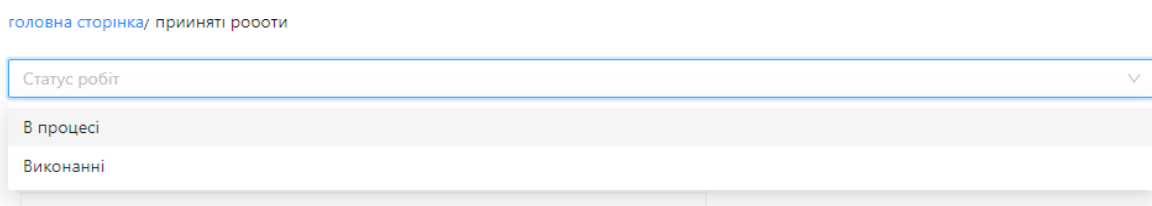


Рис. 3.26 – Фільтри по яким можна переглянути роботи фрілансера

Після успішного завершення роботи фрілансером, сторінка перегляду отримує додатковий інтерактивний елемент ‘завершити роботу’, при натисканні на який відкривається модальне вікно (рис.3.27-3.28)

← Потрібно зробити односторінковий сайт для оплати квитків

10 \$ Оплата за годину	2 неділі Час на виконання	Elementary Рівень англійського	2 Кількість відгуків
----------------------------------	-------------------------------------	--	--------------------------------

Потрібно зробити сайт із логотипом, можливість вибрати кількість квитків при цьому знизу має змінюватись ціна. Після цього підключити сервіс Portmone через API (потрібно, щоб остаточна сума за 1 квиток або кілька автоматично відображалася на платіжній сторінці Portmone). Створити додатково сторінки: неуспішну оплату або успішну з динамічною кількістю сплачених квитків, а також часом і датою. Потрібно зробити лише мобільну версію сайту. Є деякі напрацювання, які допоможуть зрозуміти, що саме потрібно. Начебто нескладно, найголовніше зробити так, що сума на платіжній сторінці заповнювалася автоматично.

ReactJs
NestJs
html
css

Зробити ставку
Закінчити роботу

Рис. 3.27 Вигляд роботи у фрілансера

Відправка роботи

✕

* Повідомлення:

Відправити

Рис. 3.28 –Вікно відправки роботи

У замовників є сторінка на якій вони можуть переглядати надіслані роботи та робити певні дії з ними (рис.3.29-3.31).

головна сторінка/ надіслані роботи

В процесі

Посилання на роботу

+380937031828
Номер телефону

employer@gmail.com
Емейл

Артем
Ім'я

ось ваша робота <https://drive.google.com/drive/my-drive>

Скарга
Відхилити
Підтвердити

< 0 / 0 >

Рис. 3.29 - Сторінка надісланих робіт

					ІПЗ.КР.Б-121-22-ПЗ	46

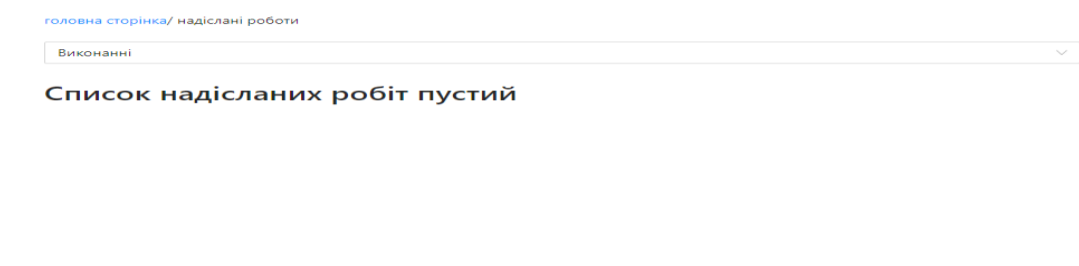


Рис. 3.30 – Вигляд сторінки коли робіт немає

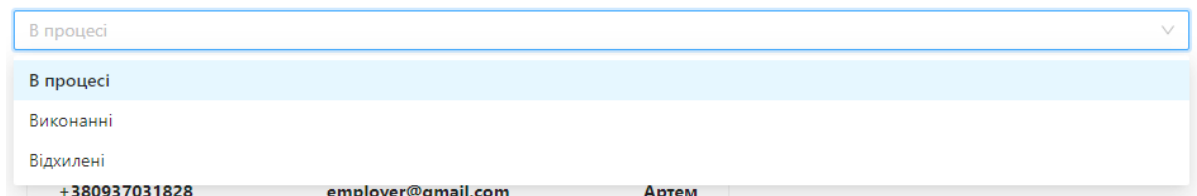


Рис. 3.31 – Теги по яким можна виконувати пошук

В кожній картці надісланої роботи присутні 3 кнопки, які виконують відповідні дії (рис.3.32-3.34).

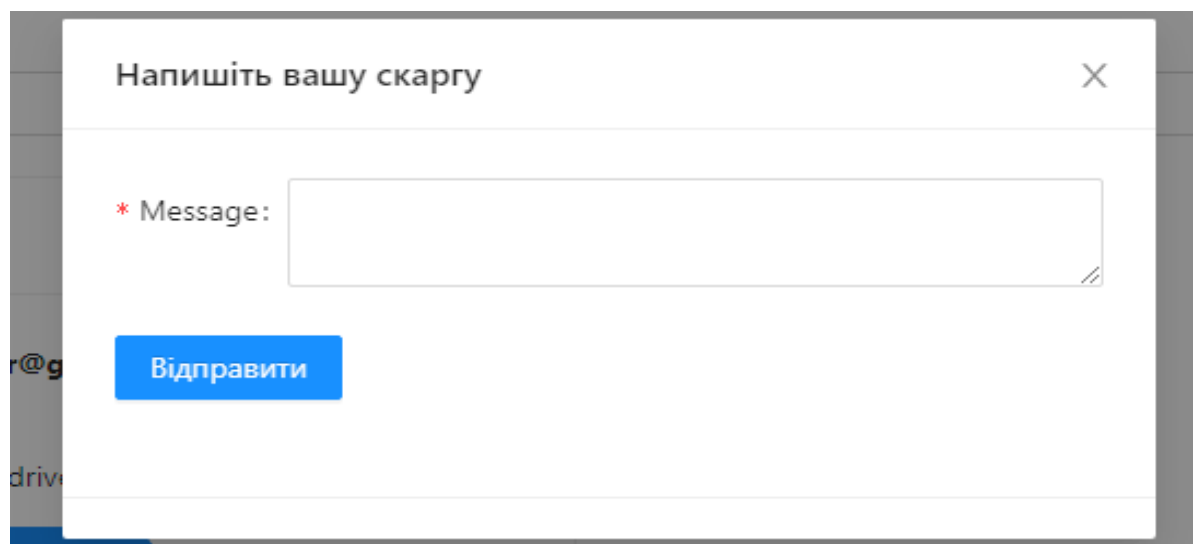


Рис. 3.32 – Відкриття форми для написання скарги

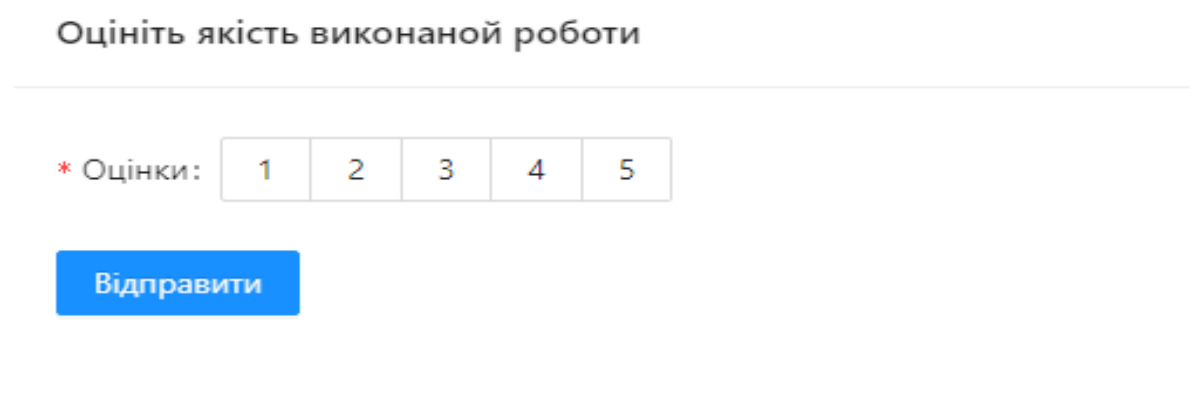


Рис. 3.33 – Відкриття форми для оцінки роботи фрілансера

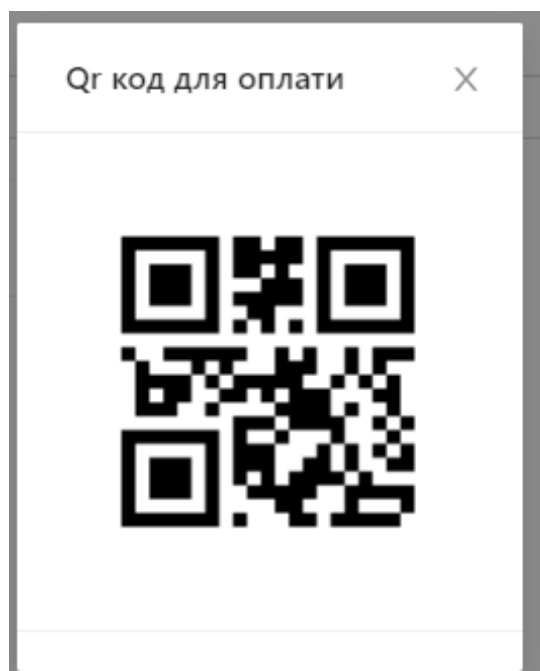


Рис. 3.34 – Генерація qr коду [30] для оплати, після виставлення оцінки

3.2 Тестування роботи програмного комплексу

Важливим етапом розробки програмного забезпечення є процес тестування та виправлення дефектів. Для перевірки коректності роботи сайту виконаємо процес створення замовлення, відгуку на нього та надсилання готової роботи.

1. Вхід в обліковий запис:

передумови: користувач знаходиться на формі авторизації;

кроки: введення облікових даних, натиснення на кнопку "Увійти";

очікуваний результат: користувач буде перенаправлений на головну сторінку;

фактичний результат: користувач був перенаправлений на головну сторінку.

2. Перехід на сторінку створення замовлення:

передумови: користувач аутентифікований у системі;

кроки: перехід на сторінку створення заняття за допомогою натискання кнопки "Створити роботу" на головній сторінці;

очікуваний результат: користувач бачить сторінку створення роботи;

фактичний результат: користувач бачить сторінку створення роботи.

					ІПЗ.КР.Б-121-22-ПЗ	
						48

3. Створення роботи:

передумови: користувач знаходиться на сторінці створення роботи;

кроки: коректне заповнення форми й відправка на сервер;

очікуваний результат: користувач перенаправляється на головну сторінку, відображається повідомлення про успішне створення роботи та вивід картки цієї роботи на головній сторінці;

фактичний результат: користувач був перенаправлений на головну сторінку й було відображене повідомлення про успішне створення роботи, також щойно створену роботу було видно на головній сторінці.

4. Перегляд роботи:

передумови: користувач створив роботу та знаходиться на головній сторінці;

кроки: користувач знаходить створену роботу та натискає на кнопку “Переглянути більше”;

очікуваний результат: система відкриває сторінку детальної інформації про роботу;

фактичний результат: користувача перенесло на сторінку де написана детальна інформація про роботу.

5. Створення ставки на роботу:

передумови: фрілансер знаходиться на сторінці роботи;

кроки: користувач натискає на кнопку “Зробити ставку” після чого з'являється модальне вікно в якому користувач заповнює всі поля та натискає на кнопку “Відправити”;

очікуваний результат: появляється нотифікація про успішне створення;

фактичний результат: з'явилась нотифікація про успішне створення.

6. Підтвердження фрілансера на роботу:

передумови: замовник знаходиться на сторінці відкликів на роботи;

кроки: замовник обирає найкращий відклик та взаємодіє з кнопкою “Підтвердити”, після чого з'являється повідомлення про успішну операцію;

очікуваний результат: у фрілансера на сторінці “Прийняті роботи” з'явиться нова робота;

фактичний результат: у фрілансера на сторінці “Прийняті роботи” з'явилася нова робота.

7. Здача роботи:

передумови: користувач знаходиться на сторінці роботи на яку його підтвердили;

кроки: користувач натискає на кнопку “Закінчити роботу” та заповнює форму яка відкрилась в модальному вікні;

очікуваний результат: буде отримано сповіщення про успішну дію;

фактичний результат: було отримано сповіщення про успішну дію.

8. Завершення роботи:

передумови: користувач знаходиться на сторінці надісланих робіт;

кроки: користувач натискає на кнопку “Підтвердити” та обирає оцінку яку він бажає поставити фрілансеру після чого відкривається інше модальне вікно з qr кодом картки фрілансера на яку потрібно зробити переказ згідно домовленій ціні;

очікуваний результат: буде отримано сповіщення про успішну дію, та робота перейде у розділ “виконані”;

фактичний результат: було отримано сповіщення про успішну дію, та робота перейшла у розділ “виконані”.

Висновки до третього розділу

У даному розділі наведено опис структури для платформи фрілансу. Описано кожен елемент системи та його взаємодію з іншими компонентами.

Також був представлений опис інтерфейсу розробленого програмного продукту. У цьому описі були візуально продемонстровані всі функціональні можливості додатку, залежно від ролі користувача.

Для перевірки коректності роботи додатку було проведене тестування на тестовому прикладі. Тестування було проведено поетапно з метою порівняння очікуваних результатів з фактичними результатами кожного етапу.

ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано сайт фрілансу. Для цього було вирішено наступні завдання:

1. Проведено аналіз предметної області, що дозволило визначити основні вимоги до сайту. На основі цього аналізу було складено набір функціональних можливостей для вебдодатку.

2. З метою розробки конкурентоспроможного програмного забезпечення було проведено порівняльну характеристику платформ, які найбільш наближені до нашої. Це дало змогу сформуванати набір найважливіших характеристик для нашого продукту.

3. Відповідно до обраної архітектури та поставлених вимог були вибрані необхідні інструментальні засоби для реалізації проєкту.

4. Були визначені засоби використання платформи замовниками та фрілансерами. Також була спроектована об'єктно-орієнтована схема моделі, детально описано кожен її структурний елемент.

5. На основі визначених сутностей платформи фріланс була спроектована база даних, яка використовує ORM для збереження даних платформи та виконання операцій над ними.

6. Описано алгоритм взаємодії компонентів системи та логіку виконання операцій з даними.

7. Проведено опис інтерфейсу додатку, його структурних елементів та функціональних можливостей, з урахуванням ролі користувача та інструментів роботи з даними.

8. Також було проведено тестування додатку з метою перевірки коректності його роботи.

Реалізований програмний комплекс готовий до використання. Сайт може, спростити процес пошуку роботи для фрілансерів та знаходження клієнтів для виконавців.

Цей програмний продукт не є кінцевою версією і може бути поліпшений шляхом впровадження нового функціоналу.

					ІПЗ.КР.Б-121-22-ПЗ	
						53

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Upwork. [Електроний ресурс]. Доступ за посиланням: <https://www.upwork.com/>.
2. Fiverr. [Електроний ресурс]. Доступ за посиланням: <https://www.fiverr.com>.
3. FreelanceUa. [Електроний ресурс]. Доступ за посиланням: <https://freelance.ua/>.
4. Чому SOLID — важлива складова мислення програміста. Розбираємося на прикладах з кодом.[Електроний ресурс]. Доступ за посиланням: <https://dou.ua/lenta/articles/solid-principles/>.
5. HTTP протокол. Що і де тестувати? [Електроний ресурс]. Доступ за посиланням: <https://training.qatestlab.com/blog/technical-articles/http-protocol-what-and-where-to-test/>.
6. Use of DTO for Validation in NestJS Application [Електроний ресурс]. Доступ за посиланням: <https://www.linkedin.com/pulse/use-dto-validation-nestjs-application-ruhul-amin>.
7. Що таке монолітна архітектура? [Електроний ресурс]. Доступ за посиланням: <https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/>.
8. Мікросервісна архітектура для початківців. Частина I. [Електроний ресурс]. Доступ за посиланням : <https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/>.
9. Безсерверна та мікросервісна архітектура: що вибрати для свого продукту? [Електроний ресурс]. Доступ за посиланням: <https://hashdork.com/uk/serverless-vs-microservices/>.
- 10.The Reasons why you Must Use Visual Studio Code. [Електроний ресурс]. Доступ за посиланням: <https://blog.devgenius.io/the-reasons-why-you-must-use-visual-studio-code-b522f946a849>.

- 11."React – JavaScript-бібліотека для створення користувацьких інтерфейсів", uk.reactjs.org, 2021. [Електроний ресурс]. Доступ за посиланням: <https://uk.reactjs.org/>.
- 12."Getting Started with Redux | Redux", [Redux.js.org](https://redux.js.org), 2022. [Електроний ресурс]. Доступ за посиланням: <https://redux.js.org/introduction/getting-started>.
- 13.Ant Design of React[Електроний ресурс]. Доступ за посиланням: <https://ant.design/components/overview/>.
- 14.Typescriptlang.org, 2022. [Електроний ресурс]. Доступ за посиланням: <https://www.typescriptlang.org/docs/handbook/2/basic-types.html>.
- 15.12 essential ESLint rules for React [Електроний ресурс]. Доступ за посиланням: <https://blog.logrocket.com/12-essential-eslint-rules-react/>.
- 16.10 reasons why you should use NodeJs. | Microsoft Docs. [Електроний ресурс]. Доступ за посиланням: <https://www.projectpro.io/article/10-reasons-why-you-should-use-nodejs/129>.
- 17.2022 : Why NestJS is the best backend framework. [Електроний ресурс]. Доступ за посиланням: <https://poub330.medium.com/2021-why-nestjs-is-the-best-backend-framework-79d91d7e317c>.
- 18.TypeORM: Object-relational mapping with Node.js. [Електроний ресурс]. Доступ за посиланням: <https://blog.logrocket.com/typeorm-object-relational-mapping-node-js/>.
- 19.MySQL Workbench: If you do not know it yet you are missing out on a great ally.[Електроний ресурс]. Доступ за посиланням: <https://pandorafms.com/blog/mysql-workbench/>.
- 20.Make Your Computer Into a Server in 10 Minutes (free Software). [Електроний ресурс]. Доступ за посиланням: <https://www.instructables.com/Make-Your-Computer-Into-A-Server-in-10-Minutes-fr/>.
- 21.Many-to-many relations. [Електроний ресурс]. Доступ за посиланням: <https://orkhan.gitbook.io/typeorm/docs/many-to-many-relations>.

- 22.UML diagrams – which diagram to use and why. Доступ за посиланням:
<https://drawio-app.com/blog/uml-diagrams/>.
- 23.Database Relationships in MySQL. [Електроний ресурс]. Доступ за посиланням: <https://dotnettutorials.net/lesson/database-relationships-in-mysql/>.
- 24.Chrome, Edge, Firefox, Opera, or Safari: Which Browser Is Best for 2023?[Електроний ресурс]. Доступ за посиланням:
<https://www.pcmag.com/picks/chrome-edge-firefox-opera-or-safari-which-browser-is-best>.
- 25.Authentication.[Електроний ресурс]. Доступ за посиланням:
<https://docs.nestjs.com/security/authentication>.
- 26.How To Build Autocomplete search with Nestjs, Elasticsearch and Vue. [Електроний ресурс]. Доступ за посиланням: <https://dev.to/kop7/how-to-build-autocomplete-search-with-nestjs-elasticsearch-and-vue-12h8>.
- 27.How to create a React Dropdown. [Електроний ресурс]. Доступ за посиланням: <https://www.robinwieruch.de/react-dropdown/>.
- 28.How to Make a Filter Component in React.[Електроний ресурс]. Доступ за посиланням: <https://www.freecodecamp.org/news/how-to-make-a-filter-component-in-react/>.
- 29.What is Badge?. [Електроний ресурс]. Доступ за посиланням:
<https://setproduct.com/blog/badge-ui-design>.
- 30.How to create and read QR codes in Node.js. [Електроний ресурс]. Доступ за посиланням: <https://blog.logrocket.com/create-read-qr-codes-node-js/>.

ДОДАТКИ

					ІПЗ.КР.Б-121-22-ПЗ	
						57

1. Метод, який генерує qr код

```

async generateQr(cardNumber): Promise<object> {
  try {
    return await qrcode.toDataURL(cardNumber);
  } catch (e) {
    throw new HttpException(
      "Oops some happening wrong",
      HttpStatus.INTERNAL_SERVER_ERROR,
    );
  }
}

```

2. Метод, для підтвердження виконаної роботи фрілансером

```

async approveJobFreelancer(id, userId): Promise<string> {
  try {
    const job = await this.jobRepository
      .createQueryBuilder("job")
      .where("job.id = :id", { id })
      .leftJoinAndSelect("job.bids", "bids")
      .where("bids.isApproved = :isApproved", { isApproved: true })
      .getOne();

    if (job.bids[0].userId !== userId) throw new Error();

    await this.jobRepository
      .createQueryBuilder("job")
      .update(Job)
      .where("job.id = :id", { id })
      .set({ freelancerConfirm: true })
      .execute();

    this.checkStatus(id);
    return "success";
  } catch (e) {
    console.log(e)
    throw new HttpException(

```

```

        "Oops, something happen wrong try later",
        HttpStatus.BAD_GATEWAY,
    );
}
}

```

3. Метод для отримання всіх сповіщень про виконану роботу

```

async getNotifications(dto) {
    try {
        const {
            id,
            data: { status = 'waiting' },
        } = dto;

        const page: number = dto.data.page || 1;
        const limit: number = dto.data.limit || 10;
        const offset: number = page * limit - limit;

        const notifications = await this.notificationRepository
            .createQueryBuilder("notification")
                .take(limit)
                .skip(offset)
                .leftJoin('notification.job', 'job')
                .where("job.userId = :id", { id })
                .andWhere('notification.status = :status', { status })
                .leftJoinAndSelect('job.user', 'user')
                .select(['notification', 'user.firstName', 'user.email',
                    'user.phoneNumber', 'user.id'])
                .getRawMany();

        const count = await this.notificationRepository
            .createQueryBuilder('notification')
                .leftJoin('notification.job', 'job')
                .where("job.userId = :id", { id })
                .andWhere('notification.status = :status', { status })
                .getCount()

        return { notifications, count }
    } catch (e) {

```

```

        return e;
    }
}

```

4. Метод для прийняття фрілансера на роботу

```

async approveBid(dto): Promise<string> {
    try {
        const { id, jobId } = dto;

        const job = await this.jobRepository
            .createQueryBuilder("job")
            .leftJoinAndSelect("job.tags", "Tags")
            .leftJoinAndSelect('job.bids', 'bid', 'bid.isApproved = 1')
            .where("job.id = :id", { id: id })
            .getOne();

        if (job.bids.length > 0) throw new HttpException('помилка, ця робота вже ви-
конується іншим працівником', HttpStatus.BAD_REQUEST);

        await this.bidRepository
            .createQueryBuilder('bid')
            .update(Bid)
            .where('bid.id = :id', { id })
            .set({ isApproved: true })
            .execute();

        await this.jobRepository
            .createQueryBuilder('job')
            .update(Job)
            .where('job.id = :jobId', { jobId })
            .set({ status: 'atProcess' })
            .execute();

        return 'success';
    } catch (e) {
        throw new HttpException(e.message, e.status);
    }
}

```

					ИПЗ.КР.Б-121-22-ПЗ	61