МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Веб-орієнтована інформаційна система пошуку виконавців IT-проектів»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ.мз-91с Іванова Тетяна Олександрівна

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2020 р.

Науковий керівник _____ к.т.н., доц., Парфененко Ю. В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«___» _____ 2020 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Іванова Тетяна Олександрівна
(прізвище, ім'я по батькові)

1 Тема проекту *Веб-орієнтована інформаційна система пошуку виконавців ІТ-проектів*
затверджена наказом по університету від «16» листопада 2020 р. №1773-ІІІ

2 Термін здачі студентом закінченого проекту «_07__» ___грудня___ 2020 р.

3 Вхідні дані до проекту

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, моделювання та проектування веб-орієнтованої інформаційної системи пошуку ІТ-проектів, практична реалізація проекту

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Актуальність, мета проекту, функціональні вимоги до ІС, аналіз сайтів для пошуку ІТ проектів, порівняння сайтів для пошуку ІТ проектів, моделювання роботи ІС, декомпозиція, діаграма використання інформаційної системи, засоби реалізації проекту, модель бази даних, архітектура ІС, демонстрація роботи ІС (загальні функції), демонстрація використання інформаційної системи замовником, демонстрація використання інформаційної системи виконавцем, висновки

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Аналітика	25.08.20-31.08.20	
2	Планування роботи над проектом	01.09.20-09.09.20	
3	Розробка дизайну	10.09.20-23.10.20	
4	Верстка сайту	24.10.20-07.11.20	
5	Розробка бази даних	08.11.20-13.11.20	
6	Програмування сайту та тестування	14.11.20-255.11.20	
7	Реліз додатку	26.11.20-30.11.20	
8	Оформлення пояснювальної записки	30.11.20-03.12.20	

Магістрант _____ Іванова Т.О.

Керівник роботи _____ к.т.н., доц. Парфененко Ю.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Веб-орієнтована інформаційна система пошуку виконавців IT-проектів».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 29 найменувань, 3 додатки. Загальний обсяг роботи − 133 сторінки, у тому числі 66 сторінок основного тексту, 3 сторінки списку використаних джерел, 63 сторінки додатків.

Кваліфікаційну роботу магістра присвячено розробці веб-орієнтованої системи пошуку виконавців IT-проектів. В першому розділі роботи проведено аналіз предметної області, а також виконано огляд актуальних досліджень та публікацій, аналіз аналогів, постановку задачі. В другому розділі було сформовано мету та задачі дослідження. Даний етап також включає в себе вибір засобів реалізації. В третьому розділі описано проектування фріланс-системи. Було виконано IDEF0, її декомпозиція, діаграма варіантів використання функціоналу інформаційної системи та спроектована база даних. В четвертому розділі було виконано реалізацію та детальний опис використання програмного додатку зі сторони замовника, виконавця та можливості адміністратора.

Результатом проведеної роботи є система пошуку виконавців IT проектів у вигляді веб-сайту. Практичне значення роботи полягає у тому, що створюваний сервіс буде використовуватись, як відкрита фріланс-платформа, яка, в свою чергу, повинна полегшити знаходження замовлення та комунікацію між клієнтами та фрілансерами.

Результатом роботи є веб-орієнтована інформаційна система пошуку виконавців IT-проектів

Ключові слова: інформаційна система, сервіс, фріланс, замовник, замовлення, виконавець, додаток, звіт, робота, проект.

ЗМІСТ

ВСТУП

У наш час широкої популярності набула робота поза штатом компаній, яку називають фрілансом. Також можна розглядати фріланс як перший крок до власного бізнесу. Основні переваги віддаленої роботи фрілансерів - праця вдома або там де вам зручно, самостійне складання свого робочого графіку та відсутність контролю.

Фріланс в Україні - рід діяльності, який безпосередньо належить до IT-сфери. Найчастіше це проектування, адміністрування сайтів та наповнення їх контентом, розробка веб або графічного дизайну. Фріланс ідеально підійде також і новачкам у сфері IT.

Ідеальний процес роботи на фрілансі виглядає так: обираєте біржу, подаєте заявку на цікаві проекти, виконуєте завдання і отримуєте гроші. На ділі все набагато складніше: важко знайти гарне замовлення і виділитися на тлі інших виконавців.

Хоча існує чимало платформ для людей, які можуть знайти роботу чи отримати допомогу в різних видах діяльності, через збільшення кількості таких платформ та очевидну конкуренцію, яка виникає, багато доступних робочих місць не досягають рівня прожиткового мінімуму, що може бути важким для багатьох людей. Пошук потрібного фахівця може бути дуже складним завданням. Це вимагає глибокого знання культури вашої організації та чіткого розуміння особистості кандидата, сильних сторін, інтересів, стилю роботи та інших характеристик.

Працює також багато спеціалізованих сайтів, покликаних допомогти фрілансерам знайти чергове замовлення. Кожна така біржа має свої умови участі та гарантії оплати праці. Проаналізувавши деякі з них можна дійти висновку, що на деяких низький контроль з боку біржі за діяльністю учасників або не зовсім зручний інтерфейс, а на інших - висока комісія яку беруть на себе організатори біржі за що учасник отримує гарантії оплати та впевненість у замовнику.

Метою даного проекту – створення веб-орієнтованої інформаційної системи пошуку виконавців IT-проектів. Цільовою аудиторією ресурсу будуть замовники тих чи інших проектів та виконавці – фрілансери.

Для досягнення мети необхідно вирішити такі задачі:

- провести аналіз предметної області застосування інформаційних технологій при вирішенні задачі пошуку виконавців IT-проектів та розглянути існуючі веб-сервіси-аналоги;
- виконати планування робіт з розробки веб-орієнтованої інформаційної системи та обрати засоби реалізації;
- виконати проектування та моделювання варіантів використання веб-орієнтованої інформаційної системи пошуку виконавців IT-проектів;
- розробити веб-орієнтовану інформаційну систему пошуку виконавців IT-проектів та виконати її тестування.

Створюваний сервіс повинен полегшити знаходження замовлення та комунікацію між клієнтами та фрілансерами. Головною особливістю даного сервісу буде можливість створення акаунту, як для особистого використання, так і для компанії. Даний тип акаунту дозволяє приєднувати людей до власної команди. Це розширює можливості роботи та підвищує можливість виконавця отримати замовлення й розвиватися в цілому. Крім того, полегшується робота для замовника з вибором виконавця. Можна отримати повний пакет послуг витративши на пошуки менше зусиль.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Дослідження актуальності проблеми

На сьогодні все більшої популярності набуває замовлення виконання завдань в певному напрямку у фрілансерів. Компанії все частіше винаймають фрілансерів для виконання робочих завдань, а не співробітників, які працюють у штатному режимі. У той же час, все більш працівників повністю відмовляються від своєї традиційної роботи, віддаючи перевагу фрілансу. Професійні навички, досвід та прибутковість – головні переваги найму фрілансерів [1].

Спочатку фрілансерів іменували «працівниками без кордонів», оскільки підрядники та їх клієнти можуть належати до будь-якої частині світу. Фрілансери виконують замовлення онлайн, їх робоче місце це комп’ютер з веб-камерою і інтернетом, електронна пошта, мобільний телефон та банківська картка для прийому платежів. І це достатньо для виконання замовлення. Розвиток IT-технологій породив велику кількість професій, які мають особливі стосунки з інформаційними технологіями [2]. У зв’язку з цим фрілансерами частіше стають працівники IT галузі:

– веб-розробники (веб-дизайнери, веб-програмісти, тестувальники програмного забезпечення, контент менеджери тощо),

– розробники програмного забезпечення (розробники ігор, мобільних додатків, баз даних, QA-інженери, розробники back-end та front-end тощо),

– фахівці з мережевих та інформаційних систем (мережеві та серверні адміністратори, адміністратори баз даних, тощо),

– дизайн та мультимедіа (графічні дизайнери, дизайнери логотипів, ілюстратори, аніматори, аудіо та відео виробництво, тощо) та інші.

За даними Elance Україна посіла третє місце за наймами віддалених працівників.

Існує багато причин, чому людей приваблює робота фрілансера. Розглянемо низку переваг:

– Фахівець можете працювати у будь-якій точці світу. Якщо у вас є комп'ютер та Інтернет, це дозволяє вам вільно працювати вдома чи деінде. Це головна причина, чому люди, як правило, працюють вільно, не підпорядковуючись роботодавцю.

– Як фрілансер, можете встановити власну ціну та робочий час. Ви також можете вибрати фактичне замовлення або частину загальної роботи, яку ви хочете виконати. Крім того, можна заробляти гроші в міжнародних валютах.

– Фрілансери можуть працювати самостійно, але розраховують на підтримку від експертів у своїй галузі.

– Фрілансерам не потрібно працювати певний час. Все що потрібно – завершити проект вчасно. Фрілансер має можливість працювати будь-коли: вранці, ввечері або навіть вночі.

Зазвичай, як і будь-яка сфера, робота на фрілансі має багато недоліків [3]:

− На відміну від роботи в офісі, фрілансери повинні бути особливо обережними при роботі з клієнтами та заздалегідь звернути увагу на вартість та спосіб оплати за виконану роботу.

− Роботодавці збирають податки на прибуток та заробітну плату шляхом відрахувань із заробітної плати та направляючи їх уряду. Незалежні підрядники повинні нести переказ цих платежів і можуть вимагати щоквартальних розрахункових податків.

Революція фрілансу займає значну частину ринку, і ця частина зростає з кожним роком. На сьогоднішній день збільшилась кількість інновацій та відбулись значні зміни у широкому діапазоні областей, оскільки цифрові платформи фрілансерів еволюціонують та трансформуються. Інтернет-платформи для товарів та послуг зростають [4]. Фрілансери є постачальниками роботи, необхідної для клієнтів та Інтернет-платформ (таких як Upwork, Guru тощо). Ці платформи утворюють торгову платформу, на якій клієнти та підрядники можуть взаємодіяти,

зазвичай стягуючи комісію з однієї або обох сторін. Таким чином, робота перетікає від фрілансерів до клієнтів, а гроші – від клієнтів фрілансерам. Для залучення клієнтів онлайн-платформа має велику кількість зареєстрованих фрілансерів, а фрілансери, в свою чергу, готові зареєструватися, якщо вони бачать велику кількість клієнтів на платформі. Більшість онлайн-платформ надають безкоштовну реєстрацію для фрілансерів і стягують з клієнтів гроші за доступ до бази даних зареєстрованих фрілансерів.

Виділяють декілька моделей взаємодії фрілансерів та клієнтів [5]:

1. Робота одержується, виконується та доставляється клієнту онлайн,
2. Робота одержується та доставляється клієнту онлайн, а виконується оф лайн,
3. Робота одержується та виконується онлайн, а доставляється клієнту в оф лайн-режимі,
4. Робота отримується за допомогою інтернету, а виконується та доставляється клієнту в оф лайн-режимі.

На рис. 1.1 представлена модель взаємодії між фрілансером та клієнтом.

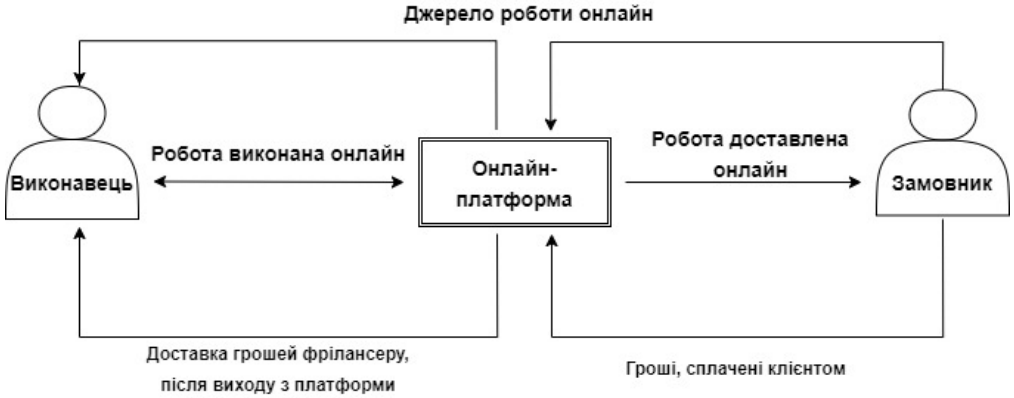

Рисунок 1.1 – Модель інтернет-фрілансу

Як бачимо з рис. 1.1, головним інструментом взаємодії між замовником та виконавцем робіт є онлайн-платформа.

Для успіху на фріланс-платформах працівники постійно повинні бути в інтернеті та бути на постійному взаємозв'язку зі своїми клієнтами. Деякі біржі, такі як oDesk.com, серед фрілансерів-новачків проводять іспити, для того щоб оцінити рівень їх навичок, наприклад з PHP, ASP.net, різних CMS, таких як Joomla та Wordpress. Висновки іспитів фіксуються у профілі фрілансера і це дозволяє клієнту оцінити технічні можливості виконавця [6].

Не дивно, те що 2020 рік вважається розквітом фрілансу, особливо на ринку надання IT-послуг. Згідно статистиці, робочі місця, які спостерігали найбільший попит у роботодавців в другому кварталі, як правило, був пов'язаний з пандемією [7].

Відслідкуємо спостереження перебігу та темпів майбутнього зростання та тенденцій революції по заданій тематиці.

Перш за все це тенденція, яка має назву «Фірми всередині фірм». Найвідомішими представниками цього руху стали Facebook та Amazon, потім різні універмаги, які переорієнтувались на фізичні майданчики брендів, та ін. Зараз ми починаємо спостерігати процес об'єднання і на інформаційних платформах: об'єднання фрілансерів які створюють свої «фірми всередині фірми».

Деякі фріланс-біржі називають їх «міні-командами», яким подобається працювати разом, які мають структуру та керівництво і, як правило, працюють у спеціалізованих сферах. Нещодавнє дослідження в Стенфорді показує, як даного типу об'єднання фрілансерів можуть прискорити інновації.

По-друге, більший акцент на оцінюванні знань та компетентності. По мірі того, як HR потрапляє у гру з відбору фрілансерів, компанії все частіше будуть винаймати осіб, які відповідають культурі та певним навичкам і мають необхідний технічний досвід. Все більше інформаційних платформ оцінюють та кваліфікують фрілансерів на основі міжособистісної ефективності та відслідковують відгуки клієнтів про такого типу компетенції. Як висловився генеральний директор однієї інформаційної системи, для більшості ранніх клієнтів очікуваними навичками та вміннями є технології 80/20 та поведінкові навички. Зрештою, це є інженери, які

винаймали інших інженерів. По мірі того, як HR та рекрути долучилися у гру, правила почали змінюватися.

Крім того, постійний зріст асоціацій фрілансерів. Freelance Union –приклад системи, яка перевірена роками, але це не єдина асоціація, яка надає фрілансерам поради та підтримку [8]. З роками у фахівців збільшуються можливості на базі спільноти.

Одним із прикладів є організація віртуальних мереж VNO, але спостерігаємо добровільні асоціації фрілансерів та вільні гільдії, такі як Hoxsby Collective та IPSE у Великобританії [9].

Більше того, хоча існує багато платформ для людей, які можуть знайти роботу чи отримати допомогу в різних видах діяльності, через збільшення кількості таких платформ та конкуренцію, більшість доступних робочих місць ще не досягли рівня виживання. Для багатьох людей це важке випробування. Пошук потрібного експерта може бути непростим завданням. Це вимагає глибокого розуміння культури компанії та чіткого розуміння особистості кандидата, сильних сторін, інтересів, стилю роботи та інших характеристик.

І хоча традиційний ринок праці продовжує боротися в усьому світі, майбутнє буде залежати від роботи в інтернеті. Кожна робота все частіше виконується за допомогою програмного забезпечення у хмарі. Віддалений веб-портал на робочому місці збільшує дохід індустрії програмного забезпечення. Він також пропонує стартапам можливість створювати нові компанії та наймати для них фрілансерів, тим самим зменшуючи безробіття.

1.2. Аналіз систем пошуку виконавців IT проектів

Процес отримання замовлення, зазвичай на фрілансі виглядає так: фахівець обирає біржу, подає заявку на проект, який його зацікавив, виконує роботу та

отримує гроші. Насправді все складніше: нелегко знайти прийнятну роботу та виділитися на тлі інших виконавців.

Існує чимало спеціальних веб-сайтів, які допомагають фрілансерам отримати чергову роботу. Кожна така біржа має свої умови участі та гарантії оплати праці. Виконаємо аналіз чотирьох фріланс-платформ конкурентів, які мають високі позиції на ринку (табл.1.1).

Таблиця 1.1 – Порівняльна таблиця фріланс-платформ

№	Назва	Переваги	Недоліки
1	Freelancehunt	Вибір способу оплати послуг - користувачі можуть оплачувати послуги сервісу картами, електронними грошима, крипто валютою або готівкою.	Висока конкуренція на деякі види робіт
		Прозора робота - дії співробітників Freelancehunt.com регламентовані правилами сервісу і публічним договором з надання послуг або проведення конкурсів [10]. Також опис послуг та інструкції по роботі на сервісі детально викладені в базі знань.	

Продовження таблиці 1.1

№	Назва	Переваги	Недоліки
1	Freelancehunt	Зворотній зв'язок - команда відповідає на всі питання і побажання клієнтів, роблячи все, щоб сервіс був комфортним і зручним. Для цього створені: форум ідей і пропозицій, паблік в соцмережах, блог про фріланс.	
		Інструменти безпеки - Сейф, Бізнес Сейф	
		Порівняно легке просування по рейтингу	
		Кілька можливостей для заробітку - проекти, конкурси і бонусна програма, в якій ви маєте можливість заробляти і отримувати бонуси, запрошуючи інших користувачів	
2	Weblancer	Якщо виникають питання на початковому етапі роботи, їх можна задати колегам «по цеху».	Обмежує той факт, що для «безпечної угоди» введення / виведення коштів здійснюється тільки на гаманець Webmoney.

Продовження таблиці 1.1

№	Назва	Переваги	Недоліки
2	Weblancer	Проекту вже багато років, це майже «стара школа» фрілансу [11]. Механізм роботи відточений, перевірений і підігнаний під вимоги виконавців та замовників, можна сказати збалансований.	Висока комісія за операцію, яка викликає відчуття подвійної оплати.
		Дана біржа фріланса в Україні, Росії та інших країнах СНД має прив'язку до IP-адреси.	За можливість виконувати проекти на даній біржі потрібно буде сплачувати певну комісію.
			Досить своєрідна робота адміністрації по відношенню до клієнтів.
3	Freelancer	На цій платформі можливо знайти фрілансерів з усього світу, якщо потрібно працювати з наявністю часового поясу чи ні.	Багато неякісних проектів із надмірними вимогами та незначною віддачою.
		Гарна служба підтримки, проста у використанні та використанні [12]. Проблеми можуть бути вирішені за лічені секунди.	Велика кількість фейкових проектів на даному сервісі.

Продовження таблиці 1.1

№	Назва	Переваги	Недоліки
3	Freelancer	Можливість виконувати безпечні платежі.	Автоматизовані заявки фрілансерів, у результаті чого важко зрозуміти, може фрілансер виконати цю роботу чи ні.
		Легко спілкуватися з поточними та майбутніми клієнтами за допомогою чату на сторінці.	
		Тести на компетентність дають клієнтам можливість зрозуміти, чим насправді виділяється фрілансер.	
4	Upwork	Більш високооплачувані проекти. У порівнянні з бюджетними фріланс-сайтами, такими як Fiverr або Freelancer.com, клієнти UpWork частіше платять вищі тарифи за послуги.	Крім оплати за подання пропозицій, UpWork також стягує відсоток за кожен проект на основі ваших загальних рахунків. Плата за послуги коливається від 5% до 20% залежно від заробітку виконавця.
		Однією з основних переваг UpWork є те, що платежі вбудовуються в систему одним натисканням кнопки.	На UpWork існує велика конкуренція.

Продовження таблиці 1.1

№	Назва	Переваги	Недоліки
4	Upwork	Нова структура підвищує і шанси на найм: UpWork має нову структуру, де стягується з фрілансерів невелика плата за подання пропозицій до проектів [13]. Завдяки меншій кількості загальних пропозицій та більш якісним фрілансерам, ця нова структура збільшує шанси на працевлаштування.	На перший погляд, UpWork здається досить сучасним веб-сайтом. Але коли почнеш переглядати та шукати проекти, швидко з'ясується, наскільки застарілий користувальницький досвід.
		Віддалена робота - це величезна вигода від найму на UpWork.	

Кожна із існуючих інформаційних фріланс-платформ забезпечує міст між традиційними продажами та новою цифровою економікою. Проаналізувавши існуючу інформацію, треба виділити розділи для подальшої праці над своєю інформаційною платформою.

Перше за все, потрібно зробити наголос на зручність та надійність у використанні. Візуальна частина є особливою частиною у розробці веб-сайту. Оскільки, це те, що має як і привернути увагу та зацікавити так і змусити користувачів піти зі сторінки.

Друге – це відсоток за використання інформаційної платформи. На поточному етапі не планується стягування плати за реєстрацію на створюваному веб-сайті, але ця функція можливо буде реалізована потім. Відсоток не повинен

бути занадто великим для користувачів, оскільки даний аспект теж впливає на кількість користувачів та загальну активність.

Описуючи переваги, треба звернути увагу на комунікацію. Інформаційна система для фріланс біржі повинна не лише надати користувачам підтримувати постійний зв'язок один з одним, але й надати підтримку з боку адміністрації сервісу. Дана функція буде реалізована через чат.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1. Мета та задачі дослідження

Метою проекту є створення веб-орієнтованої інформаційної системи пошуку виконавців ІТ-проектів, що буде корисною як для замовників тих або інших ІТ проектів, так і для виконавців. Даний сервіс повинен сприяти пошуку замовлень та комунікацію між клієнтами та фрілансерами. Перевагами співпраці з віддаленими фахівцями є і більш низькі ціни на послуги, і можливість проглянути портфоліо фахівця, відгуки від інших замовників, рейтинг та статистику у профілі. Всі ці дані формуються в процесі роботи та їх неможливо підробити.

Цільовою аудиторією веб-орієнтованої інформаційної системи пошуку виконавців ІТ-проектів є:

- програмісти;
- дизайнери;
- SEO-фахівці;
- фахівці з відео та аудіо монтажу;
- фахівці по роботі з текстом (копірайтери, рерайтери);
- менеджери інтернет-проектів;
- адміністратори соціальних мереж;
- розробники мобільних додатків.

Розроблений веб-сайт буде використаний як відкрита біржа для різних ІТ проектів.

Функціональні вимоги до веб-орієнтованої інформаційної системи пошуку виконавців ІТ-проектів:

- Реєстрація та авторизація користувачів.
- Розміщення на сайті замовлень з можливістю прикріплення файлів.
- Перегляд замовлень зареєстрованими виконавцями та клієнтами.

- Розробка пошукової системи для формування списку робіт за різними критеріями.
- Створення заявок на одержання замовлення з можливістю прикріплення файлів.
- Можливість спілкування між клієнтом та робітником за допомогою чату.
- Формування рейтингу та історії проектів.
- Формування оповіщень.
- Блокування користувачів.
- Сервіс підтримки користувачів.

Для роботи з веб-орієнтованою інформаційною системою передбачено три групи користувачів, які повинні пройти авторизацію: адміністратори інформаційної системи (ІС), замовники, та виконавці.

Адміністратори ІС:

- блокувати користувачів;
- відповідати на питання користувачів.

Замовники:

- можуть створювати завдання з різноманітною роботою;
- переглядати, редагувати та видаляти свої завдання;
- спілкуватися з виконавцем роботи;
- оцінювати роботи в портфоліо виконавців;

Виконавці:

- перегляд існуючих завдань на сайті;
- створення заявки на отримання роботи;
- спілкуватися з замовником роботи;
- оцінювати роботи в портфоліо інших виконавців;
- створювати акаунт типу «Команда»;
- генерація резюме.

Головною особливістю даного сервісу є можливість створення акаунту, як для особистого використання, так і для компанії. Даний тип акаунту дозволяє приєднувати людей до власної команди. Дана специфіка розширює можливості роботи та підвищує ймовірність виконавця отримати замовлення й розвиватися в цілому. Крім того, полегшується робота для замовника з вибором виконавця. Можна отримати повний пакет послуг витративши на пошуки менше зусиль.

Інша відмінність – конструктор для генерації резюме відразу на сторінці профілю. Дана можливість надається лише виконавцю, базуючись на вміннях та досвіду роботи.

На відміну від резюме, ведення блогу буде доступно як виконавцю, так і замовнику. Кожен бажаючий зможе робити різні публікації на своєму профілі, переглядати інші теми які їх зацікавили та залишати свої коментарі.

2.2. Вибір технології реалізації веб-орієнтованої інформаційної системи пошуку виконавців IT-проектів

Метою цього проекту є розробка веб-орієнтованої інформаційної системи, а на сьогодні актуальною є розробка таких систем з використанням фреймворків. Майже кожна мова програмування має багато фреймворків. Вибір фреймворку для веб-сайту – це настільки ж важливий крок, як і сам вибір мови програмування.

Програмне забезпечення, яке створюється і використовується розробниками для написання веб-сайтів, називають фреймворком. Це своєрідний каркас для створення інтернет-проектів, не зважаючи на об’єм та складність, і яка складається з різноманітних елементів. Він дозволяє веб-системі, яка розробляється, розкрити свої функціональні можливості, додаючи різні модулі, блоки, програмні пакети та ін. Фреймворк задає проекту, заздалегідь обдуману архітектуру.

Розробку інформаційної системи для біржі фрілансу IT проектів треба розподілити на декілька етапів. Перший етап - це верстка візуальної половини веб-сайту. Для інтерфейсу будуть використані HTML5 та CSS3. Даний інструментарій є необхідним для написання коду сайту на цьому етапі.

Для реалізації додаткової анімації та взаємодії зареєстрованих осіб на сайті буде використовуватись JavaScript. На даний момент часу вона має кілька відомих фреймворків [14]. Найпопулярнішими є React.js, Vue.js та Angular.js. Проаналізуємо інформацію для обраної технології реалізації проекту (табл.2.1).

Таблиця 2.1 – Порівняльна таблиця фреймворків мови JavaScript

№	Назва	Переваги	Недоліки
1	React.js	Стабільна та зріла технологія	React забезпечує лише частину View моделі MVC [15].
		Підвищення продуктивності та швидкості	React - все ще досить нова технологія, і вона розвивається надзвичайно швидко, що може викликати труднощі у розробників.
		Скорочений час виходу на ринок	Відсутність належної документації.
		Тестування та налагодження зручності	

Продовження таблиці 2.1

№	Назва	Переваги	Недоліки
2	Vue.js	Спільнота, що використовують даний фреймворк постійно зростає[16].	Вибаглива при написані коду.
		Підтримка з усіх основних середовищ IDE.	Фреймворк має менше супутніх бібліотек.
		Документація добре написана, зрозуміла та доступна.	
		Фреймворк має великий ступінь налаштування.	
		Автономний фреймворк.	
3	Angular.js	Двостороння прив'язка даних.	Недосвідченість роботи з MVC.
		Покращена продуктивність сервера.	Браузеру може знадобитися час, щоб відтворити сторінки веб-сайтів та програм, розроблених за допомогою фреймворку[17].

Продовження таблиці 2.1

№	Назва	Переваги	Недоліки
3	Angular.js	Швидше прототипування додатків.	Обмежена доступна документація може надалі впливати на процес навчання.

Отже, після проведного аналізу фреймворків можна зробити висновок, що для розробки інформаційної системи пошуку виконавців IT пректів найкращим варіантом є Vue.js.

Фреймворк підходить для створення інтерактивних багатосторінкових додатків. Це дозволяє швидко імпортувати основну бібліотеку та вводити Vue.js на наявні сторінки. Навіть не потрібно використовувати компоненти для більш простих функцій.

Розробники back-end частини працюють із широким колом бібліотек, API, веб-сервісів тощо [18]. Не існує ідеальної основи. Кожен фреймворк має свої переваги і недоліки. Існує можливість робити все, що завгодно, в будь-якій сучасній структурі. Кожен фреймворк надає деякі альтернативи своїм мінусам або ще кілька плюсів, а також деякі інші мінуси. Вони відповідають за впровадження систем баз даних, забезпечуючи належний зв’язок між різними веб-службами, генеруючи серверну функціональність тощо.

Розглянемо основні та найпопулярніші технології для розробки функціональної частини інформаційної системи пошуку виконавців IT проектів в табл. 2.2.

Таблиця 2.2 – Порівняльна таблиця фреймворків для back-end розробки

№	Назва	Переваги	Недоліки
1	Laravel	Простий і швидкий механізм маршрутизації.	Помилки та неточності.
		Поставляється зі своїм власним CLI.	Деякі компанії залишаться з Zend або Symfony, оскільки вони б доклали занадто багато зусиль, щоб побудувати їх знову за допомогою Laravel [19].
		Потужна система шаблонів.	
		Зрозуміла документація.	
		Має усі компоненти для створення майже любого веб-додатку.	
2	Node.js	Міцний стек технологій.	Незрілість оснащення.
		Швидка обробка та модель подій.	Зростаючий попит на досвідчених фахівців.
		Потужна корпоративна підтримка.	Високий поріг входу для розробки[20].
		Безшовна підтримка JSON.	

Продовження таблиці 2.2

3	Spring Boot	Створюйте окремі програми Spring.	В більшості випадків використовується для невеликих систем та мікросервісів.
		Високо масштабований.	Високо масштабований.
		Створений для великомасштабних додатків, які використовують хмарний підхід [21].	Якщо ви не знайомі з іншими проектами екосистеми Spring, використання їх від Spring Boot змусить вас пропустити багато концепці.
		Чудова документація.	

Отже, проаналізувавши всю інформацію, було обрано Laravel. Даний фреймворк – це найсильніший конкурент в екосистемі PHP тому, що він включає функції, необхідні для створення сучасних веб-додатків . Таким чином, це стильний та чистий фреймворк з елегантним синтаксисом для створення веб-додатків.

Для роботи із базою даних буде використовуватися MySQL, адже це безкоштовна система управління реляційними базами даних з відкритим кодом. Сьогодні він широко використовується у всьому світі, замінюючи SQL. Крім того, дана система зазвичай використовується саме з мовою сценаріїв PHP.

3 ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПОШУКУ ВИКОНАВЦІВ ІТ ПРОЕКТІВ

3.1. Структура веб-орієнтованої інформаційної системи

Розробка структури інформаційної системи є найважливішим етапом. Застосовуючи спрощені методи та різні інструменти, проектування інформаційної системи відбувається як багатоетапний процес створення та модернізації.

Якщо підкреслити етап проектування інформаційної системи як окремий етап, то його можна розмістити між аналізом та розробкою. Однак на практиці часто важко або неможливо чітко розділити етапи, оскільки проект, який починається з офіційного визначення мети проекту, зазвичай триватиме на етапах випробувань та впровадження.

Загальну структуру веб-орієнтованої інформаційної системи для пошуку виконавців ІТ проектів можна розділити на декілька частин, а саме до та після авторизації. Головна відмінність – доступ до інформації та певних сторінок сайту.

До авторизації у системі користувачу відкриті деякі модулі (табл.3.1).

Таблиця 3.1 – Сторінки до авторизації

№	Сторінка	Опис
1	Головна	Сторінка із основною інформацією про інформаційну систему, головні переваги та етапи реєстрації.
2	Авторизація	Сторінка для авторизації зареєстрованих користувачів.
3	Реєстрація	Сторінка для реєстрації нових користувачів системи.

Після авторизації користувачеві відкриваються додаткові можливості. Перш за все, це залежить від типу акаунту – виконавець чи замовник, самостійний виконавець чи команда.

Панель меню складається із посилань на ключові модулі даної системи. Розглянемо детальніше, які можливості відкриті для кожного з типу акаунтів у табл.3.2.

Таблиця 3.2 – Таблиця рівнів доступу

Сторінки	Виконавець		Замовник
	Самостійний виконавець	Команда	
Сторінка проектів	+	+	+
Сторінка виконавців	+	+	+
Сторінка повідомлень	+	+	+
Сторінка діалогів	+	+	+
Домашня сторінка користувача	+	+	+
Редагування даних	+	+	+
Портфоліо	+	+	-
Налаштування безпеки	+	+	+
Перегляд портфоліо	+	+	+
Створення замовлення	-	-	+
Перегляд проектів	-	-	+
Висунення пропозиції	+	+	-
Додавання людей до команди	-	+	-
Вихід з системи	+	+	+

Інформаційна система спроектована для якісної взаємодії всіх користувачів з розмежуванням прав доступу. Це продемонстровано у структурі інформаційної системи. Кожен зареєстрований користувач має можливість переглядати інформацію у ролі виконавців та замовників, виконувати роботу з інформацією, здійснювати пошук та інше.

Крім того, було досягнуто всіх цілей, поставлених на початку виконання даної кваліфікаційної роботи. Макети сторінок веб-орієнтованої інформаційної системи показано у додатку Б.

3.2. Структурно-функціональне моделювання процесу пошуку виконавців IT-проектів

IDEF0 - стандарт функціонального моделювання, який використовується на стадії проектування. Кожен блок представляє окремий процес, як і в інших підходах, але IDEF0 відрізняється використанням та розміщенням стрілок [22]. Окрім звичайних входів і виходів, існують ще два типи стрілок, які представляють "елементи управління" та "механізми".

Елементи управління використовуються для керування діяльністю в процесі. Елементи механізми відповідають за діяльність і вказують на трудові, матеріальні, організаційні ресурси.

Розглянемо контекстну діаграму інформаційної системи пошуку виконавців IT-проектів в нотації IDEF0 (рис.3.1). Дана діаграма демонструє основні процеси інформаційної системи. Головним процесом є пошук виконавців IT проектів. На вході маємо запит на пошук виконавців IT проектів, інформацію про користувачів та інформацію про замовлення, а в результаті роботи інформаційної системи отримуємо на виході виконаний проект, підвищення рейтингу виконавців та звіт з виконаної роботи. Механізмами є користувач, інформаційна система, веб-сервер та апаратне забезпечення. Усі процеси регулюються часом виконання проекту, функціями системи та технічним завданням.

Рисунок 3.1 – Контекстна діаграма пошуку виконавців IT-проектів в нотації IDEF0

Наступним етапом було проведено декомпозицію контекстної діаграми на функціональні блоки: авторизація у системі, створення проекту на виконання, перегляд пропозиції від виконавця, вибір виконавця, узгодження деталей замовлення та виконання, формування відгуку (рис. 3.2). Всі ці блоки тісно пов’язані між собою і результат виконання одного блоку передається в інший блок. На виході отримуємо виконаний проект, звіт з виконання робіт та підвищення рейтингу виконавця.

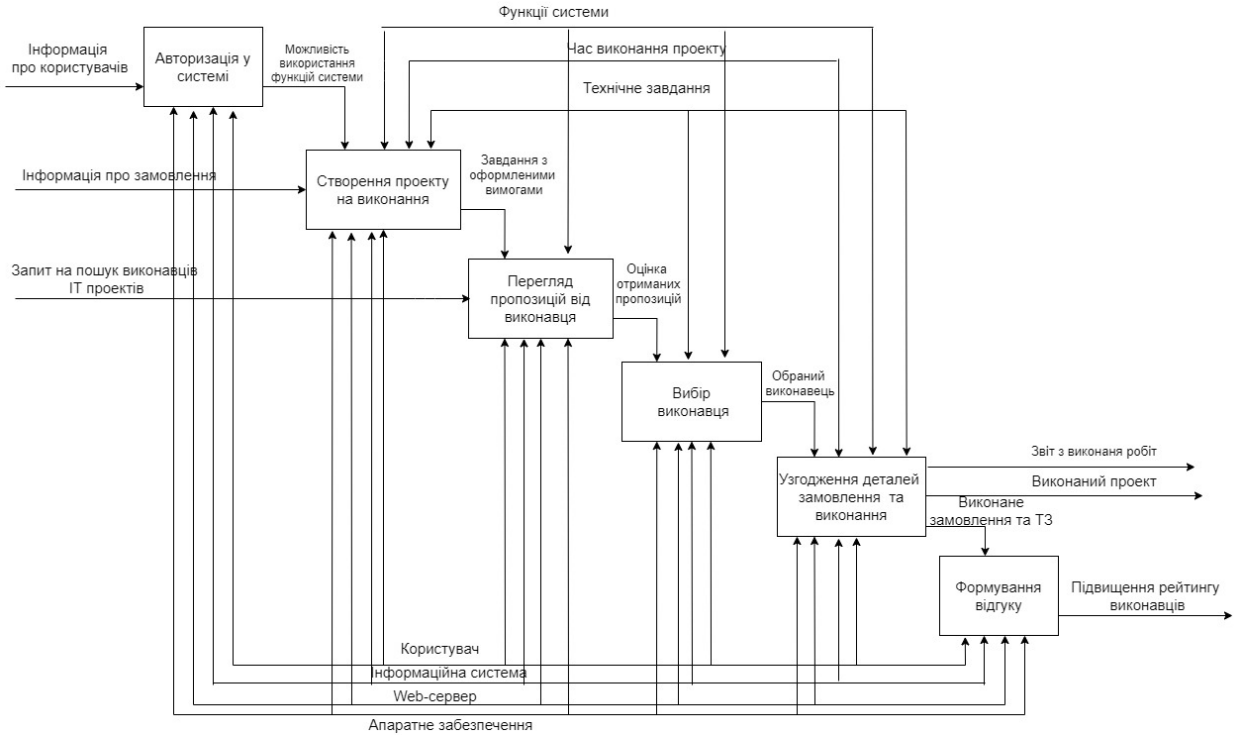

Рисунок 3.2 – Діаграма декомпозиції пошуку виконавців IT-проектів в нотації IDEF0

3.3. Моделювання варіантів використання

Мета діаграм варіантів використання – показати динамічні аспекти системи. Однак це визначення є занадто загальним, щоб описати мету, оскільки інші чотири діаграми (діяльність, послідовність, співпраця та діаграма стану) також мають те саме призначення. Ми розглянемо якесь конкретне призначення, яке відрізнятиме його від інших чотирьох діаграм.

Діаграми варіантів використання використовуються для збору системних вимог, включаючи внутрішні та зовнішні впливи. Ці вимоги в основному є вимогами до дизайну. Отже, коли система буде проаналізована для збору її функціональних можливостей, будуть підготовлені випадки використання та визначені актори.

Коли початкове завдання виконане, діаграми варіантів використання моделюються для подання зовнішнього виду.

Мета діаграм варіантів використання можуть бути наступними:

– Використовується для збору системних вимог.

– Використовується для отримання зовнішнього вигляду системи.

– Визначте зовнішні та внутрішні фактори, що впливають на систему.

– Показати взаємодію між вимогами акторів.

Розглянемо детальніше акторів та варіанти використання в табл. 3.3 та табл. 3.4.

Табл. 3.3 – Опис акторів

№	Актор	Опис діяльності
1	Замовник	Користувач, що має можливість створити замовлення в інформаційній системі.
2	Виконавець	Користувач, що має можливість виконувати замовлення, створювати портфоліо та спілкуватися з замовниками.
3	Адміністратор	Користувач, що має найбільші можливості на сайті, а саме редагувати та видаляти дані на сайті.

Табл. 3.4 – Опис варіантів використання

№	Назва варіанту використання	Опис
1	Авторизація	Дозволяє авторизуватися на сайті.
2	Реєстрація	Дозволяє зареєструватися новому користувачеві на сайті.

Продовження таблиці 3.4

№	Назва варіанту використання	Опис
3	Редагування інформації на сайті	Дозволяє змінювати загальну інформацію на сайті.
4	Перегляд замовлень та користувачів	Дозволяє переглядати зареєстрованих користувачів та замовлень на сайті.
5	Створення замовлення	Дозволяє замовнику створювати нові замовлення.
6	Вибір виконавця	Користувач має можливість обрати виконавця для свого створеного замовлення.
7	Залишення пропозиції про виконання	Дозволяє виконавцю залишати свої пропозиції на виконання замовлення, доступно лише виконавцям.
8	Створення резюме	Дозволяє створювати резюме на основі інформації на сайті.
9	Формування відгуку	Формування відгуку після виконання замовлення.
10	Створення портфоліо	Можливість додавати роботи до свого портфоліо.
11	Додавання людей до команди	Дозволяє користувачу типу «команда» додавати користувачів-виконавців до своєї команди.
12	Редагування даних на сторінці	Дозволяє користувачам редагувати особисту інформацію на сайті.

Сформувавши представлення про інформацію та функціонал інформаційної системи, було розроблено діаграму варіантів використання (рис.3.3).

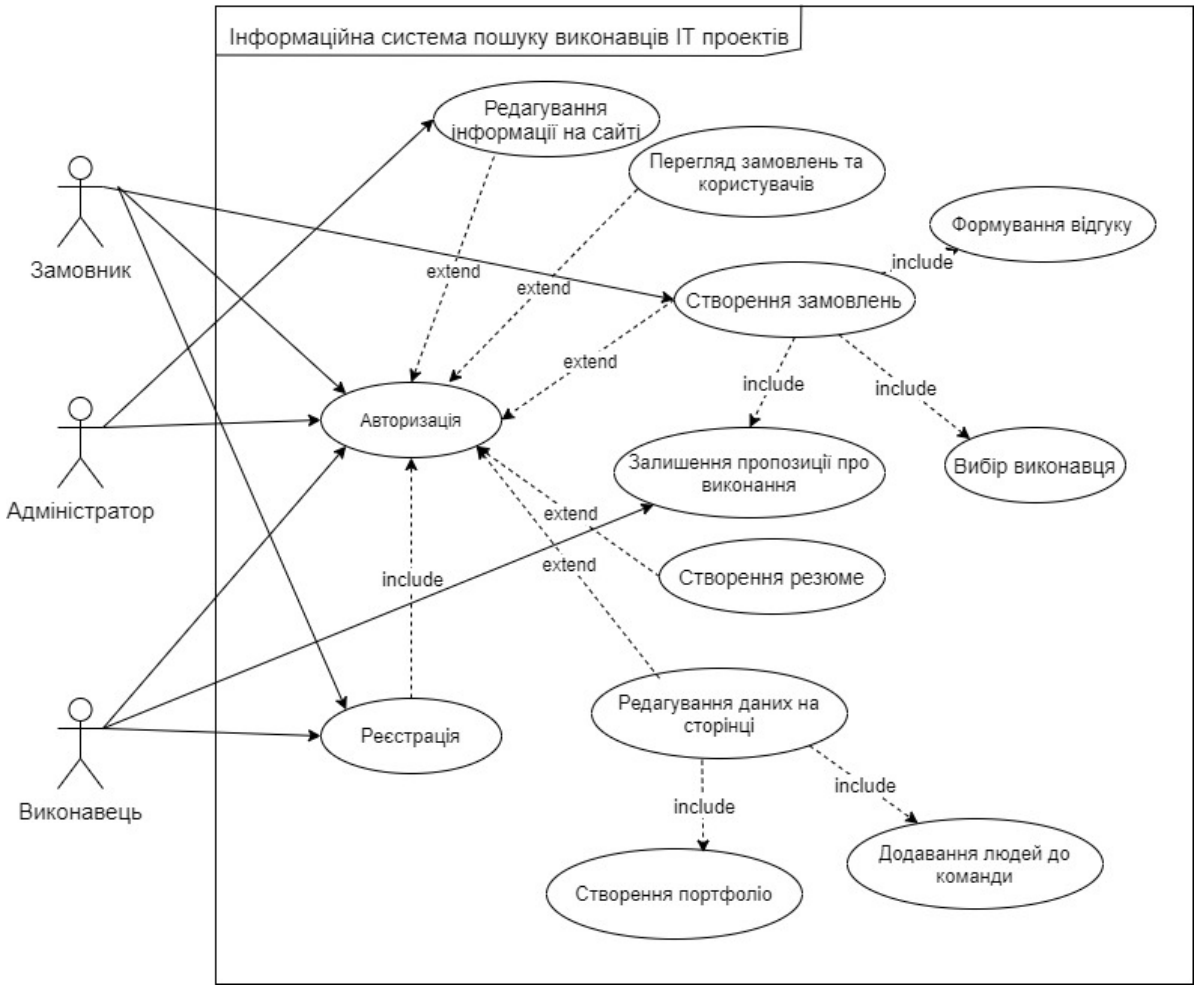

Рисунок 3.3 – Діаграма варіантів використання інформаційної системи пошуку виконавців IT-проектів

3.4. Моделювання діаграм діяльності

Діаграма діяльності – це технологія, що дозволяє описувати логіку процедур, бізнес-процеси і потоки робіт [24]. У багатьох випадках вони схожі на блок-схеми, але принципова різниця між діаграмою діяльності та позначенням блок-схеми полягає в тому, що перша підтримує паралельні процеси.

Діаграма діяльності дозволяє кожному, хто виконує цей процес, вибрати послідовність операцій. Іншими словами, діаграма лише встановлює правила обов'язкової послідовності операцій, яких слід дотримуватися [25]. Це важливо для моделювання бізнес-процесів, оскільки ці процеси зазвичай працюють паралельно. Ці діаграми також дуже корисні при розробці паралельних алгоритмів, в яких незалежні потоки можуть працювати паралельно.

На рис. 3.4-3.8 зображено діаграми діяльності модулів інформаційної системи пошуку виконавців IT-проектів.

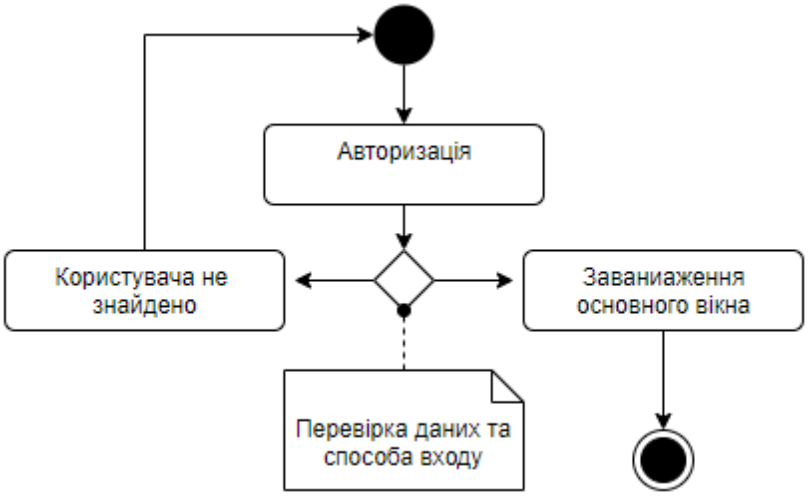

Рисунок 3.4 – Діаграма діяльності модулю авторизації

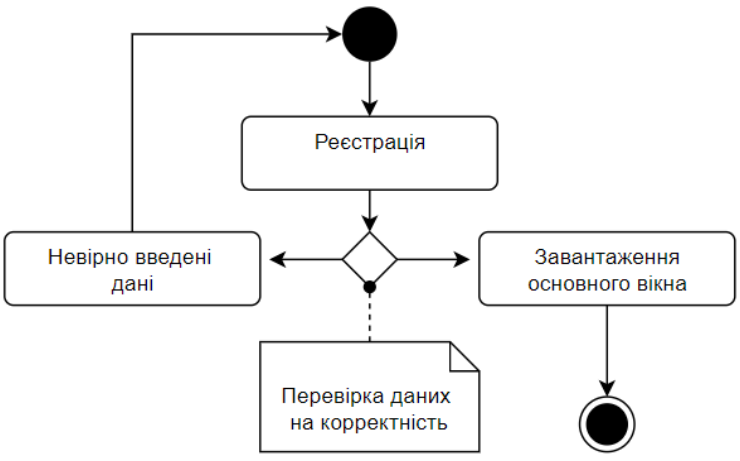

Рисунок 3.5 – Діаграма діяльності модулю реєстрації

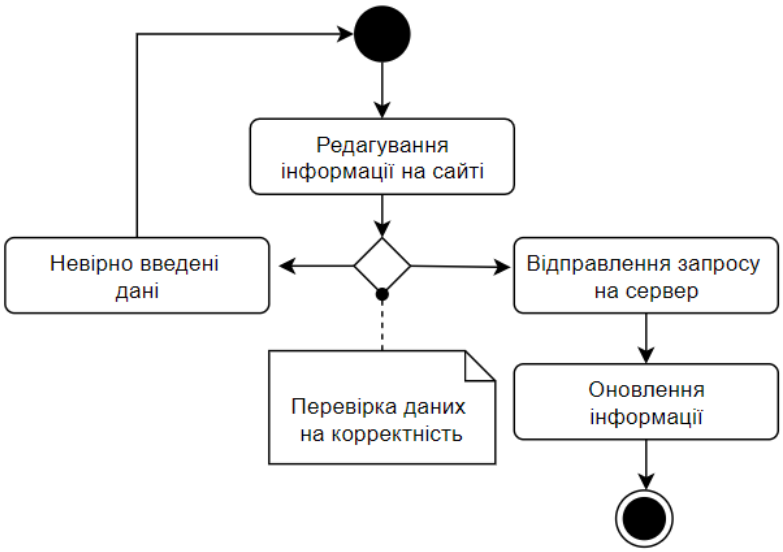

Рисунок 3.6 – Діаграма діяльності модулю редагування інформації на сайті

Рисунок 3.7 – Діаграма діяльності модулю перегляду інформації

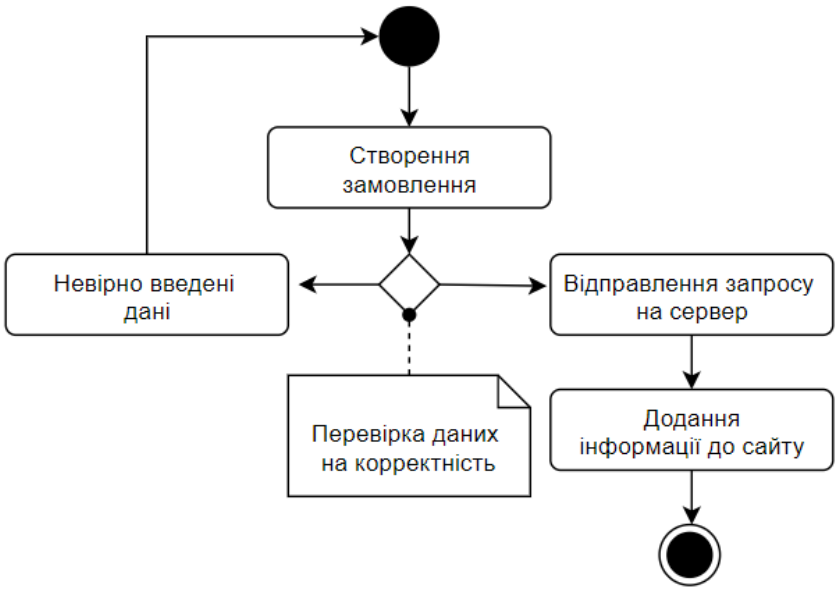

Рисунок 3.8 – Діаграма діяльності модулю створення замовлення

3.5. Проектування бази даних

Основні користувачі системи – замовник та виконавець [26]. За допомогою цієї системи вони можуть отримувати різні відомості про виконавців та замовників, завдання, портфоліо, спеціалізації та інше. Уся інформація, необхідна для роботи інформаційної системи, зберігається у базі даних (БД).

При розробці бази даних було створено такі сутності як:

– account: інформація про зареєстрованого користувача, використовується для ідентифікації за поштою; знаходиться інформація для авторизації.

– account_status: статуси зареєстрованих користувачів.

– person: загальна інформація про користувача, така як ім’я, прізвище, вік, стать та інше.

– person_status: статус користувача, а саме фрілансер чи замовник.

– team: інформація акаунта типу «команда».

– account_specialization: спеціалізація акаунту.

- specialization: можливі спеціалізації на сайті.
- skills: навички, за певними спеціалізаціями.
- portfolio: портфоліо користувача.
- order: створене замовлення з описом, ціною за часом виконання.
- order_account: таблиця для встановлення виконавця замовлення.
- request: пропозиція на виконання замовлення.
- info: інформація на сайті.

Зобразимо відносини в ER-діаграмі (рис.3.9). Проаналізувавши сутність, використовувані в моделі інформаційної системи, перейдемо до реалізації структури БД [27]. Для цього представимо імена необхідних таблиць, атрибутів, типів, їх призначення та обмеження в табл. 3.5.

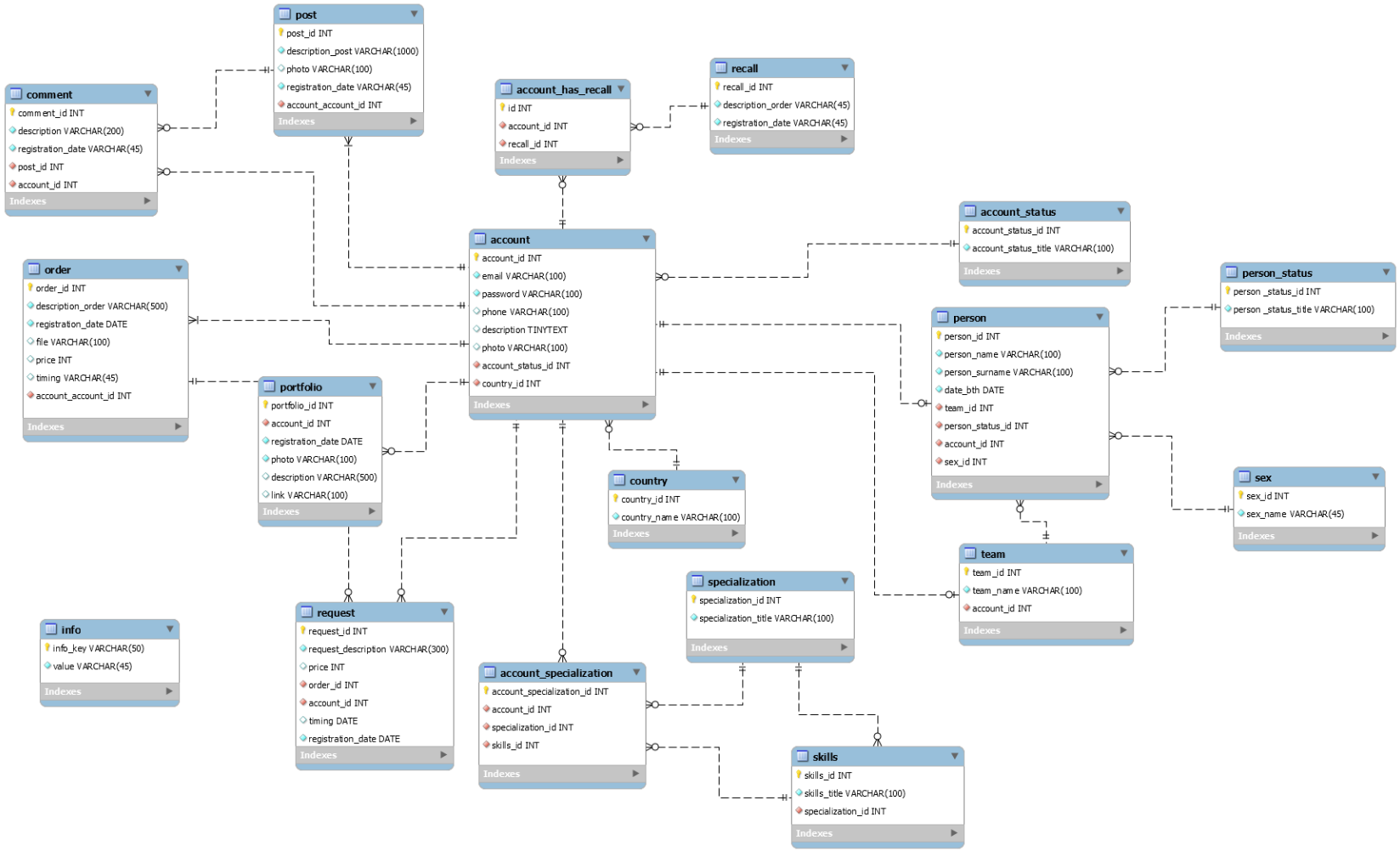

Рисунок 3.9 – ER-діаграма

Таблиця 3.5 – Інформація за таблицями ER-діаграми

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
1	account	account_id	Ідентифікатор акаунту	INTEGER	PK	Не пустий
		email	Єлектронна пошта	VARCHAR(100)		Не пустий
		password	Пароль від акаунту	VARCHAR(100)		Не пустий
		account_status_id	Статус акаунту	INTEGER	FK	Не пустий
		phone	Номер телефону	VARCHAR(100)		
		description	Коротка інформація (резюме)	TEXT		
		photo	Фото чи логотип	VARCHAR(100)		
		country	Країна проживання/роботи	VARCHAR(100)		
		registration_date	Дата реєстрації	DATE		Не пустий
2	account_status	account_status_id	Ідентифікатор статусу акаунту	INTEGER	PK	Не пустий
		account_status_title	Назва статусу	VARCHAR(100)		Не пустий
3	person	person_id	Ідентифікатор людини	INTEGER	PK	Не пустий
		account_id	Ідентифікатор акаунту	INTEGER	FK	Не пустий
		person_name	Ім’я людини	VARCHAR(100)		Не пустий
		person_surname	Прізвище людини	VARCHAR(100)		Не пустий
		date_bth	Дата народження людини	DATE		Не пустий
		sex	Стать людини	BOOLEAN		Не пустий

Продовження таблиці 3.5

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
3	person	sex	Стать людини	BOOLEAN		Не пустий
		person_status_id	Ідентифікатор статусу людини	INTEGER	FK	Не пустий
		team_id	Ідентифікатор команди	INTEGER	FK	
4	person_status	person _status_id	Ідентифікатор статусу людини	INTEGER	PK	Не пустий
		person _status_title	Назва статусу	VARCHAR(100)		Не пустий
5	team	team_id	Ідентифікатор команди	INTEGER	PK	Не пустий
		account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		team_name	Назва команди	VARCHAR(100)		Не пустий
6	account_specialization	account_specialization_id	Ідентифікатор спеціалізації аккаунта	INTEGER	PK	Не пустий
		account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		specialization_id	Ідентифікатор спеціалізації	INTEGER	FK	Не пустий
		skills_id	Ідентифікатор навичок	INTEGER	FK	
7	specialization	specialization_id	Ідентифікатор спеціалізації	INTEGER	PK	Не пустий
		specialization_title	Назва спеціалізації	VARCHAR(100)		Не пустий

Продовження таблиці 3.5

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
8	skills	skills_id	Ідентифікатор навичок	INTEGER	PK	Не пустий
		skills_title	Назва навичок	VARCHAR(100)		Не пустий
9	portfolio	portfolio_id	Ідентифікатор портфоліо	INTEGER	PK	Не пустий
		account_id	Ідентифікатор акаунту	INTEGER	FK	Не пустий
		registration_date	Дата додавання роботи	DATE		Не пустий
		photo	Фото/скрін роботи	VARCHAR(100)		Не пустий
		description	Опис роботи	TEXT		
		link	Посилання на роботу	VARCHAR(100)		
10	order	order_id	Ідентифікатор замовлення	INTEGER	PK	Не пустий
		description_order	Опис замовлення	TEXT		Не пустий
		registration_date	Дата реєстрації замовлення	DATE		Не пустий
		file	Додатковий файл замовлення	VARCHAR(100)		
		price	Ціна замовлення	INTEGER		
		timing	Час виконання/Дедлайн	DATETIME		
11	order_account	order_account_id	Ідентифікатор сформованого замовлення для виконання	INTEGER	PK	Не пустий

Продовження таблиці 3.5

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
11	order_account	account_id_customer	Ідентифікатор акаунта замовника	INTEGER	FK	Не пустий
		account_id_executor	Ідентифікатор акаунта виконавця	INTEGER	FK	
		order_id	Ідентифікатор замовлення	INTEGER	FK	Не пустий
		registration_date	Дата обрання виконавця	DATE		Не пустий
12	request	request_id	Ідентифікатор заявки	INTEGER	PK	Не пустий
		account_id	Ідентифікатор акаунта	INTEGER	FK	Не пустий
		order_id	Ідентифікатор замовлення	INTEGER	FK	Не пустий
		request_description	Текст заявки	VARCHAR(300)		Не пустий
		price	Ціна від виконавця	INTEGER		
		timing	Запропонований час виконання	DATETIME		
		registration_date	Дата написання заявки	DATE		Не пустий
13	recall	recall_id	Ідентифікатор відгуку	INTEGER	PK	Не пустий
		account_id_customer	Ідентифікатор акаунта замовника	INTEGER	FK	Не пустий
		account_id_executor	Ідентифікатор акаунта виконавця	INTEGER	FK	Не пустий
		description_order	Текст відгуку	VARCHAR(200)		Не пустий

Продовження таблиці 3.5

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
13	recall	registration_date	Дата написання відгуку	DATE		Не пустий
14	post	post_id	Ідентифікатор посту блога	INTEGER	PK	Не пустий
		account_id	Ідентифікатор акаунта	INTEGER	FK	Не пустий
		description_post	Текст посту	TEXT		Не пустий
		photo	Фото посту	VARCHAR(200)		
		registration_date	Дата написання посту	DATE		Не пустий
15	comment	comment_id	Ідентифікатор коментару	INTEGER	PK	Не пустий
		account_id	Ідентифікатор акаунта	INTEGER	FK	Не пустий
		description_ comment	Текст коментару	VARCHAR(200)		Не пустий
		registration_date	Дата написання коментарю	DATE		Не пустий
16	info	info_key	Ключ запису	VARCHAR(100)	PK	Не пустий
		value	Значення	TEXT		Не пустий

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПОШУКУ ВИКОНАВЦІВ ІТ ПРОЕКТІВ

4.1 Програмна реалізація

Програмна реалізація додатку була виконана за допомогою фреймворків Laravel та Vue.js. Розглянемо детальніше структуру інформаційної системи (рис.4.1).

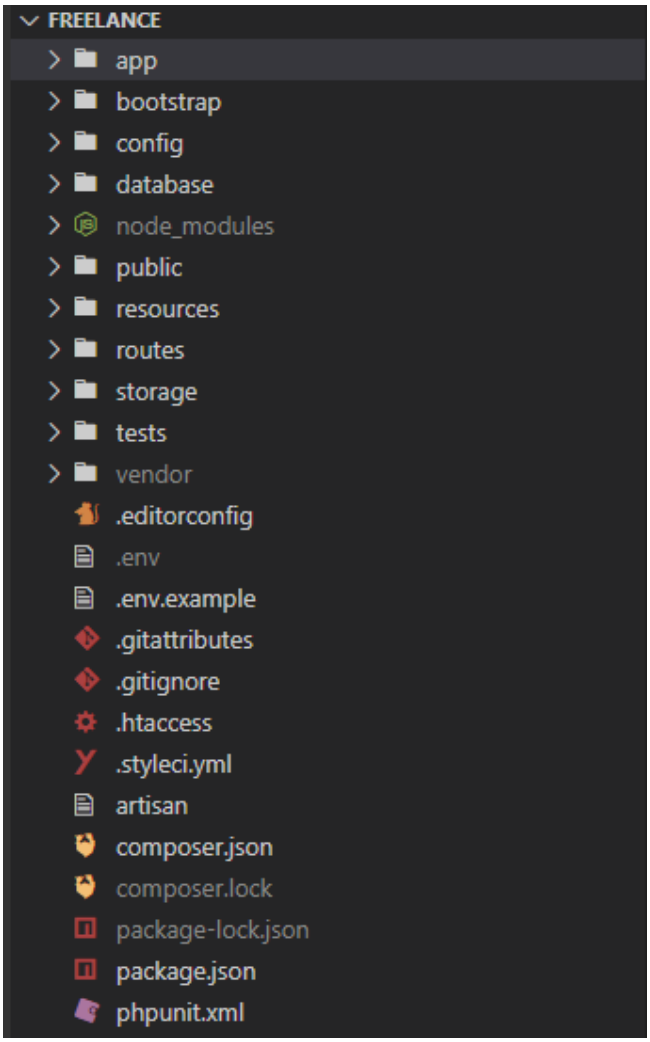

Рисунок 4.1 – Структура проекту (основні папки та додаткові файли)

Таблиця 4.1 – Опис основних папок проекту

№	Пакет	Короткий опис
1	«app»	Містить код ядра додатку.
2	«bootstrap»	Містить файли, які завантажують фреймворк і налаштовують автозагрузку.
3	«config»	Містить всі конфігураційні файли додатку
4	«database»	Містить міграції і класи для наповнення початковими даними БД.
5	«public»	Містить файл index.php, який є вхідною точкою для всіх запитів, що надходять в додаток. Також ця папка містить ресурси, такі як зображення, JavaScript, CSS.
6	«resources»	Містить початковий код, а також сирі, некомпільовані ресурси, такі як LESS, SASS, JavaScript.
7	«routes»	Містить всі визначення маршрутів додатку.
8	«storage»	Містить скомпільовані Blade-шаблони, файл-сесії, кешу файлів і інші файли, створені фреймворком.
9	«tests»	Містить автотести.

Архітектура сайту – це структура сторінок і програмної частини сайту. У неї входить навігація, мережа посилань, «хлібні крихти», сторінки категорій, файли карти сайту і так далі [28].

Робота зі структурою – одна з методик SEO. Вона впливає на роботу користувачів з ресурсом і на сприйняття його пошуковими роботами. Гарно вибудувана архітектура направляє користувачів і пошукових роботів на важливі сторінки, допомагає їм знайти на сайті те, що вони шукають.

Правильна архітектура допомагає користувачам і пошуковим системам знаходити те, що вони шукають [29]. Крім того, вона говорить системі про значимість і релевантності вашого контенту. Вона спрямовує користувача і

пошукових роботів на найважливіші сторінки, розповідає, що собою являє ваш контент.

Робота над архітектурою - робота для того, щоб сайт був простим і зрозумілим, зручним і приємним. Докладний опис сторінок web-додатку в табл.4.2.

Таблиця 4.2 – Опис сторінок додатку

№	Назва	Опис сторінки
1	Головна сторінка	Головна сторінка – перша сторінка, на яку потрапляють відвідувачі та користувачі ІС, у яких вона повинна викликати інтерес та дати можливість отримати весь спектр послуг. На даній сторінці можливо: – Ознайомитись з короткою інформацією про систему; – Виконати реєстрацію чи вхід в особистий кабінет; – Перехід на сторінку детального опису «про проект»; – Зв’язок з адміністратором ІС.
2	Форма авторизації	Дана форма складається з таких полів: – Email; – Пароль користувача; – Відновлення паролю; – Реєстрація; – Кнопки «Вхід».
3	Форма реєстрації	Реєстраційна форма складається з таких полів, які необхідно заповнити: – Прізвище; – Ім’я; – По-батькові;

Продовження таблиці 4.2

№	Назва	Опис сторінки
3	Форма реєстрації	– Стать (вибір одного із варіантів); – Дата народження (шаблон дати в форматі: день:місяць:рік); – Телефон (вибір одного із варіантів, можливість додати додатковий телефон); – Країна; – E-mail; – Пароль; – Підтвердження паролю.
4	Особистий кабінет користувача	Ця сторінка має містити: – Навігаційне меню або сайдбар до якого можуть входити пошук роботи, профіль користувача, чат, оповіщення і тд відповідно до функцій користувачів; – Контент(відповідно до меню). – Портфоліо – Конструктор резюме – Команда (за наявності)
5	Сторінка налаштувань аккаунту	Ця сторінка має містити: – Блок редагування основної інформації; – Заміна головного фото аккаунта; – Створення та формування резюме.
6	Перегляд замовників та виконавців	Ця сторінка має містити: – Фільтр; – Інформація про замовника чи виконавця.

Продовження таблиці 4.2

№	Назва	Опис сторінки
7	Сторінка з замовленнями	Ця сторінка має містити: – Фільтр; – Пошук; – Створенні вже замовлення.
8	Сторінка обраного замовлення	Ця сторінка має містити: – Замовник; – Інформація про замовлення; – Ціна та час виконання; – Пропозиції виконавців.
9	Блог	Ця сторінка має містити: – Топ публікації; – Фільтр; – Пошук; – Можливість створити власну публікацію.
10	Повідомлення	Ця сторінка потрібна для отримання та відправлення повідомлення одного користувача іншому.

4.2 Використання програмного додатку замовником

Робота інформаційної системи пошуку виконавців IT проектів розпочинається з головної сторінки, де користувач має можливість продивитися інформацію, зареєструвати та авторизуватися (рис.4.2-4.4).

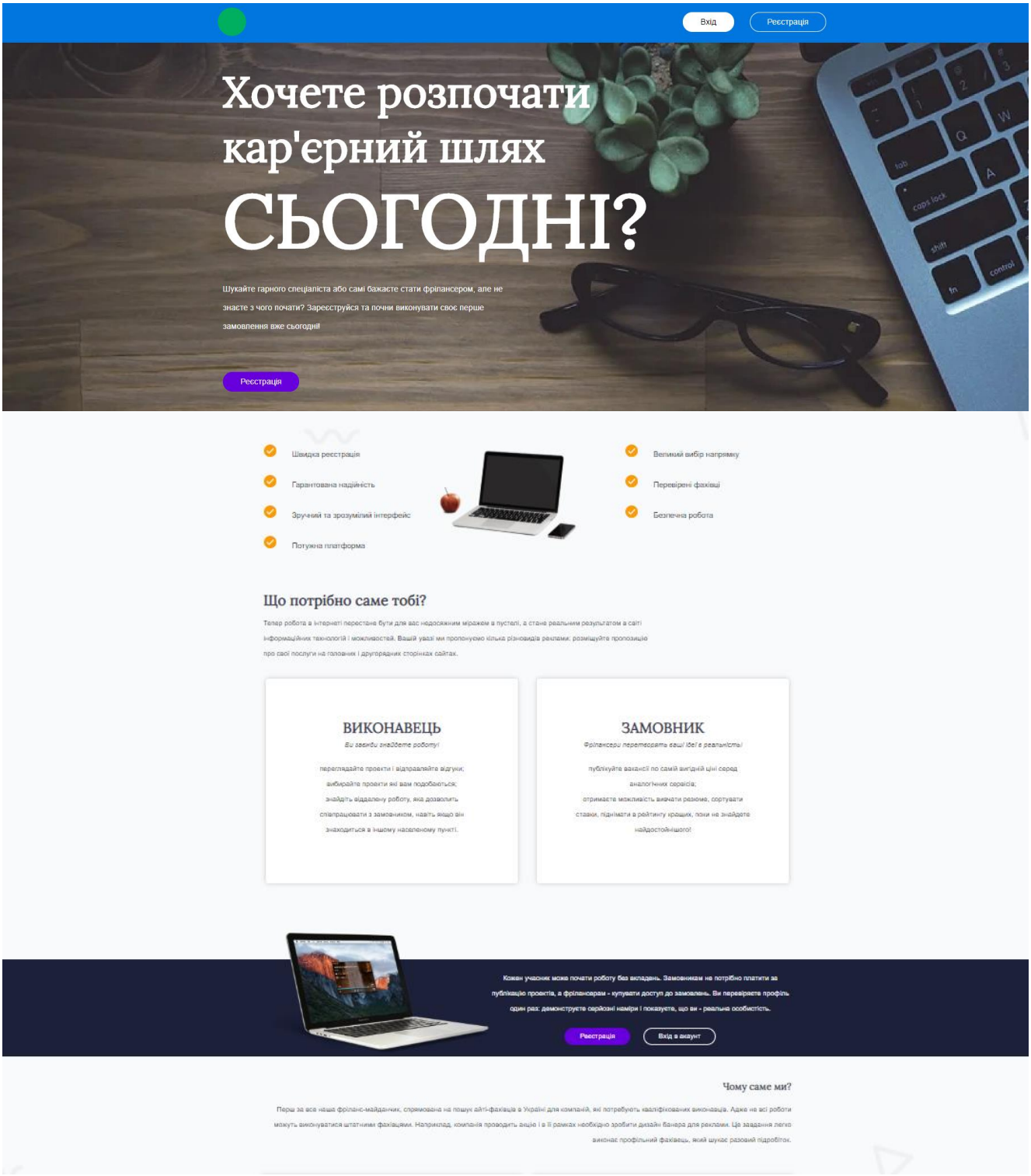

Рисунок 4.2 – Головна сторінка сайту

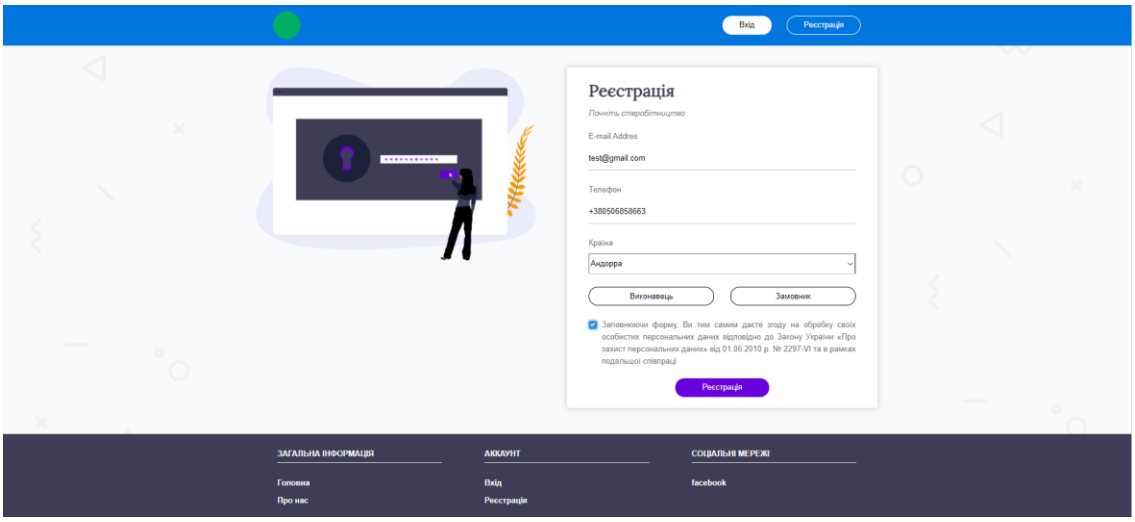

Рисунок 4.3 – Сторінка реєстрації

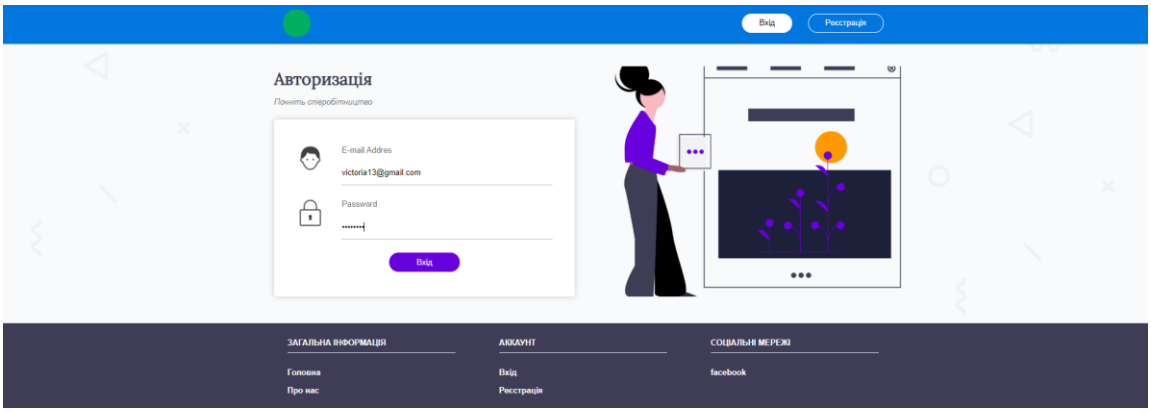

Рисунок 4.4 – Сторінка авторизації

Після авторизації та реєстрації відбувається перехід на головну сторінку (рис.4.5). Особиста сторінка користувача із можливістю перейти до редагування даних представлені на рис. 4.6-4.7.

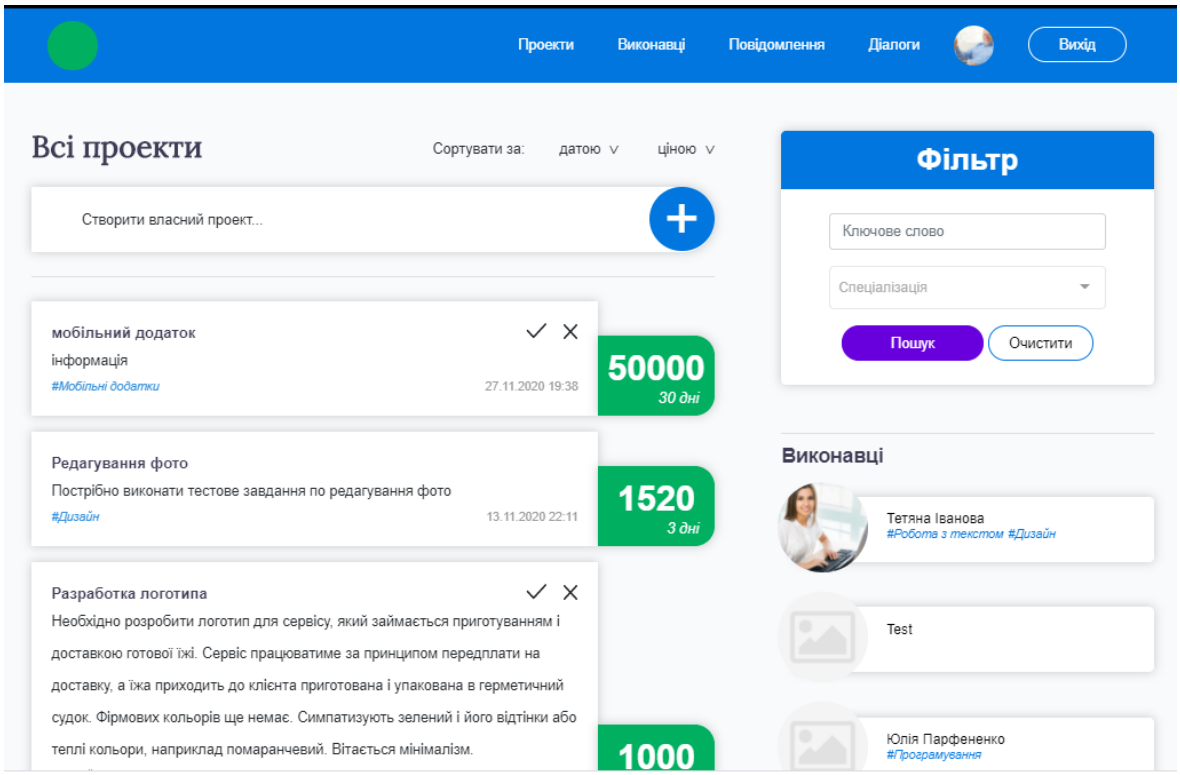

Рисунок 4.5 – Головна сторінка після авторизації

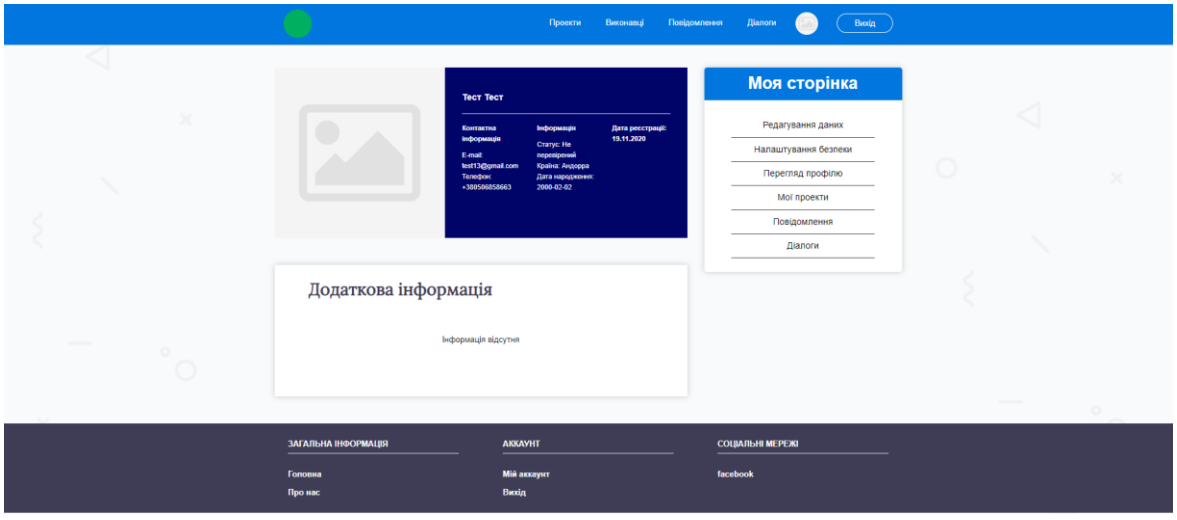

Рисунок 4.6 – Особиста сторінка користувача

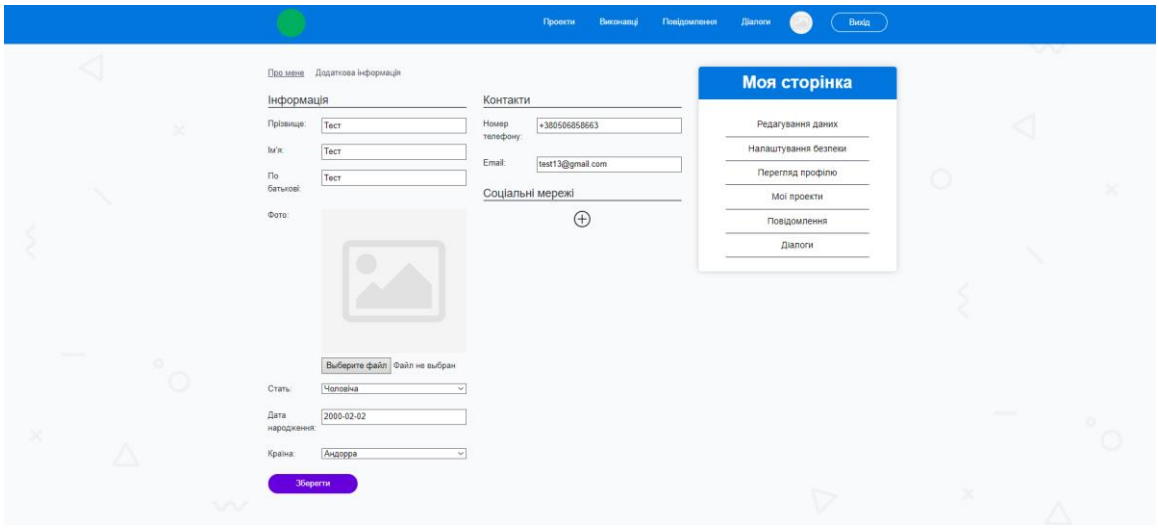

Рисунок 4.7 – Редагування даних користувача

Кожен користувач має можливість виконати фільтрацію замовлень за ключовим словом або за спеціалізацією (рис.4.8) а також переглядати виконавців на сайті (рис.4.9).

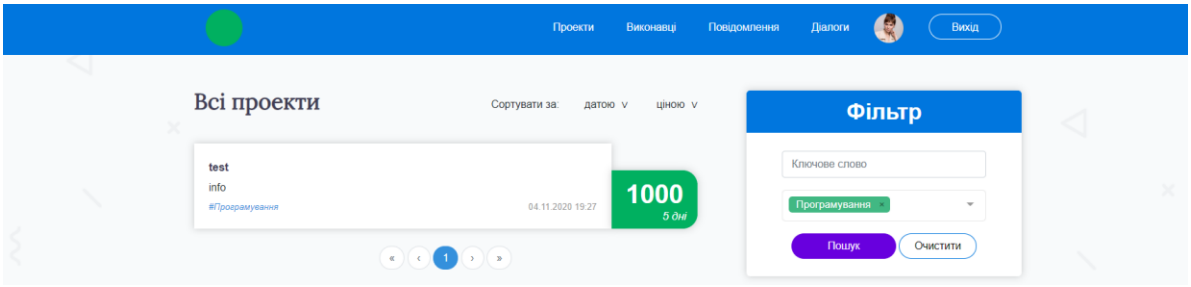

Рисунок 4.8 – Виконання пошуку проектів

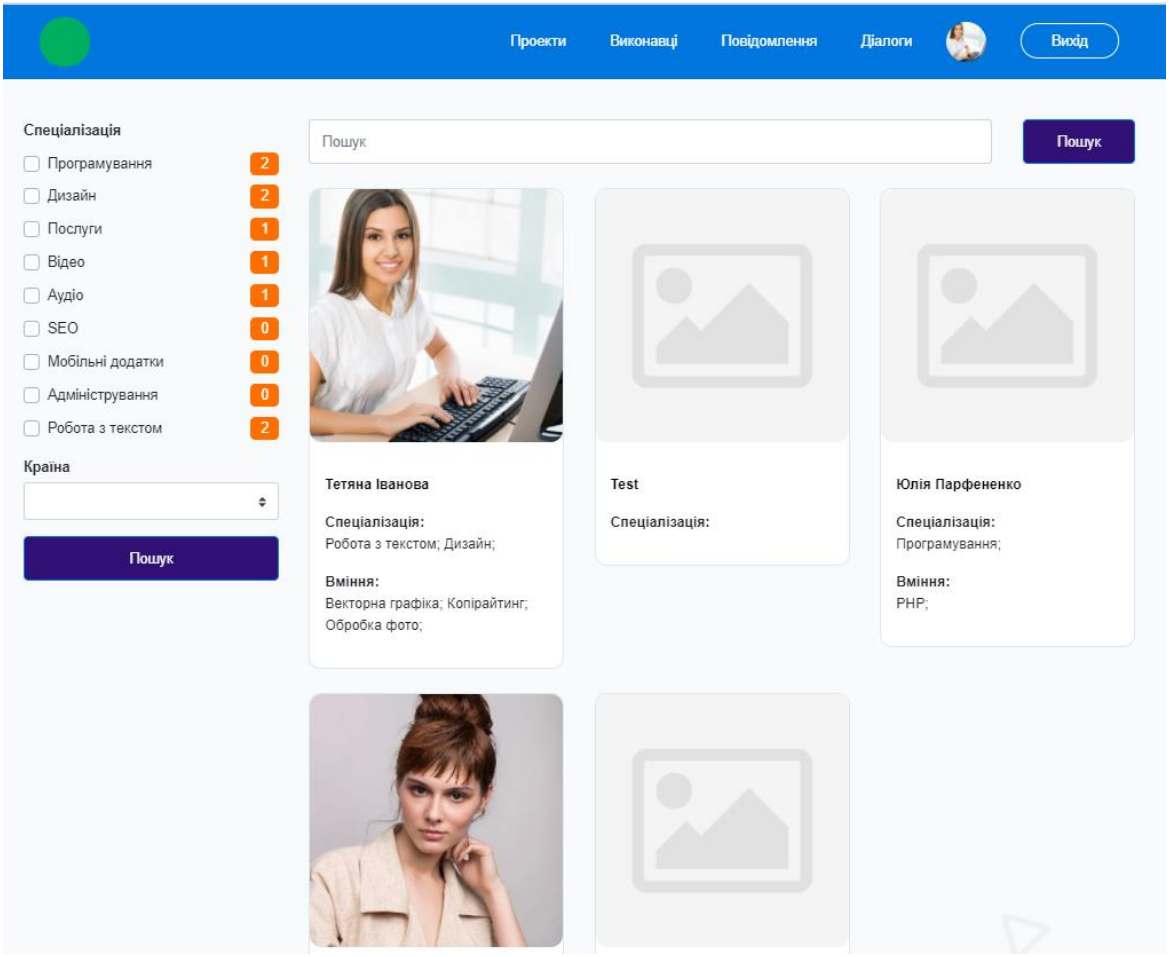

Рисунок 4.9 – Перегляд та пошук кандидатів

Користувач типу «замовник» має можливість створювати власний проект і також його редагувати. Для цього потрібно заповнити форму, що представлена на рис. 4.10-4.11.

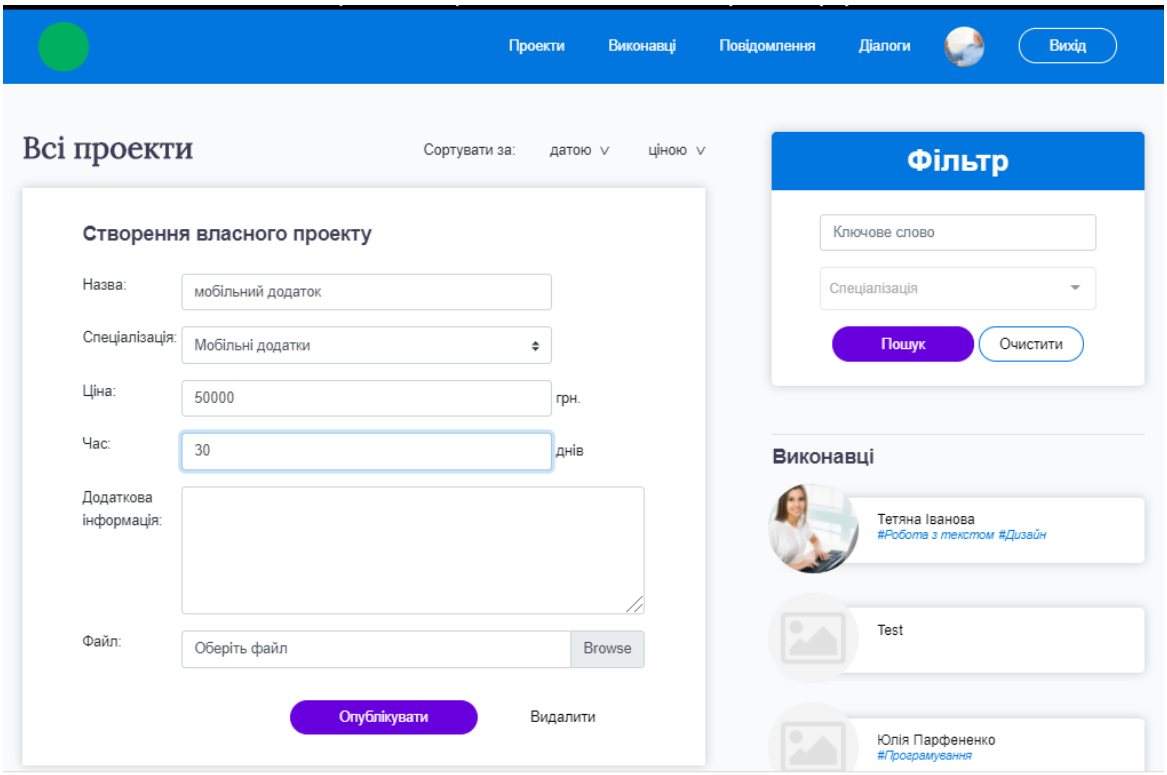

Рисунок 4.10 – Приклад створення проекту

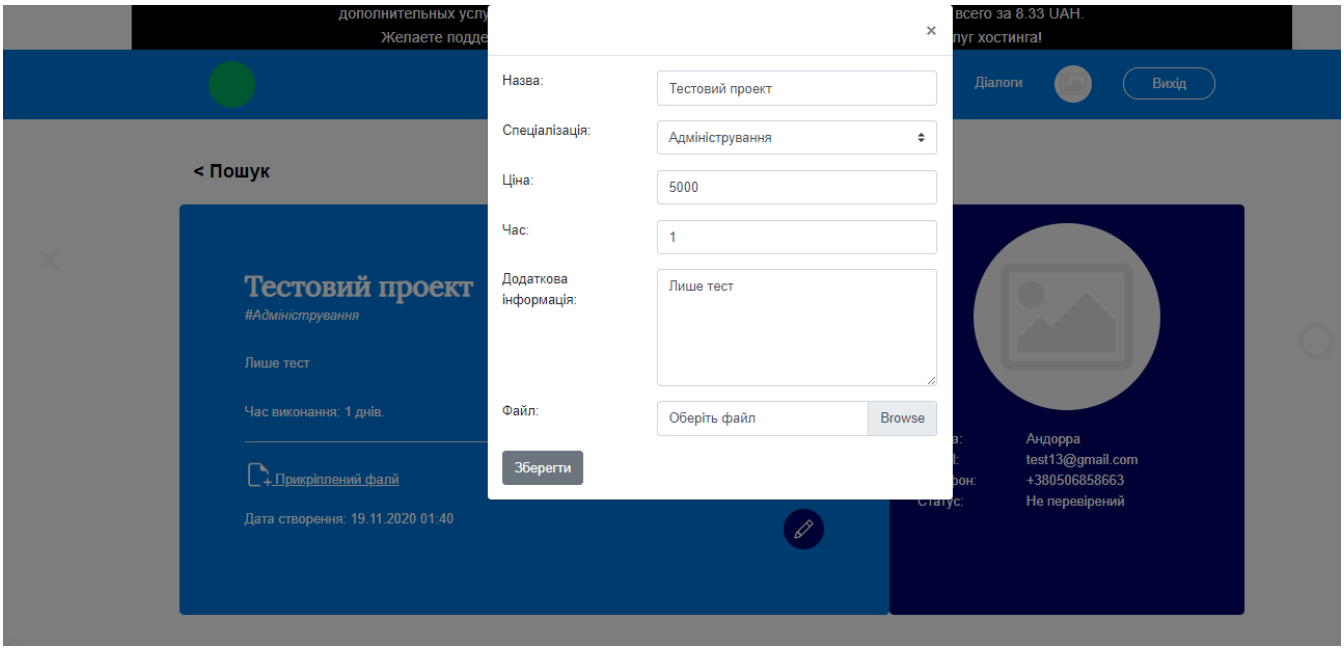

Рисунок 4.11 – Можливість редагування замовлення

Після створення проекту користувач отримує оповіщення про отримані пропозиції на виконання (рис.4.12), на основі яких буде обрано виконавця. Після того, як виконавця було обрано, подача пропозицій за даним проектом автоматично припиняється (рис. 4.13).

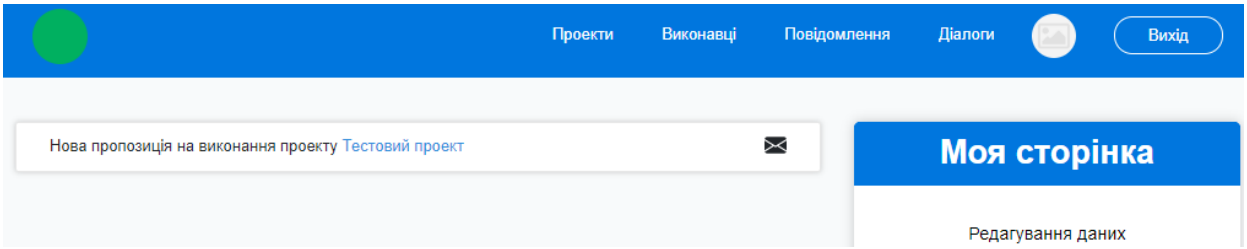

Рисунок 4.12 – Оповіщення про отримання нової пропозиції від виконавця

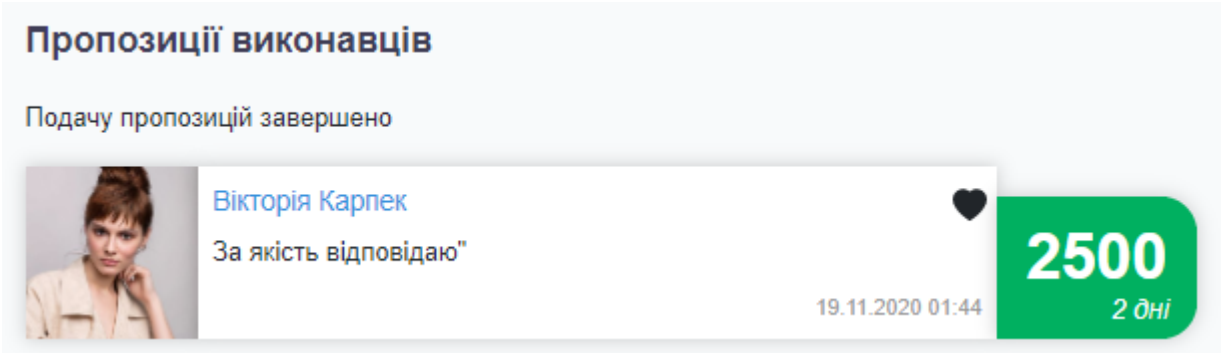

Рисунок 4.13 – Обрання виконавця замовлення

Крім того, будь-який користувач має можливість написати повідомлення іншому користувачеві (рис.4.14).

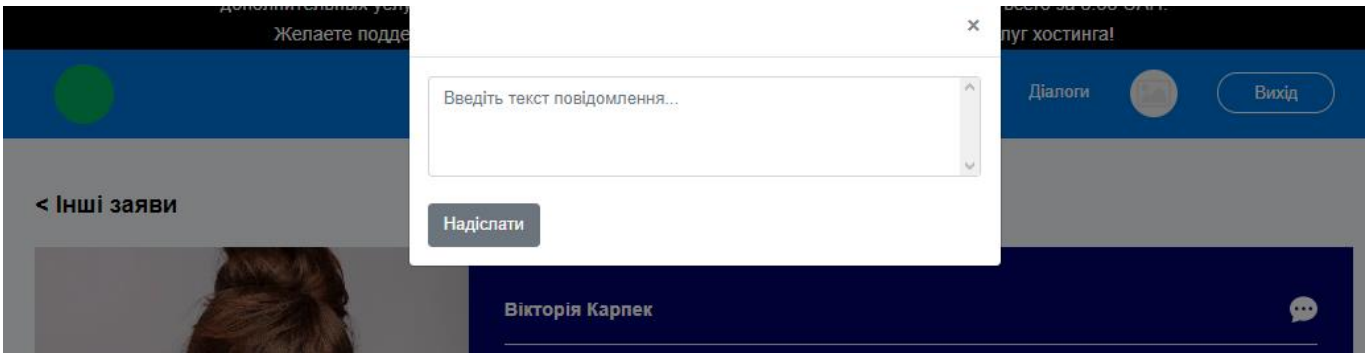

Рисунок 4.14 – Можливість написання повідомлення

4.3 Використання програмного додатку виконавцем

Користувач типу «Виконавець» має можливість створення власної пропозиції для виконання проекту будь-якої тематики (рис.4.15).

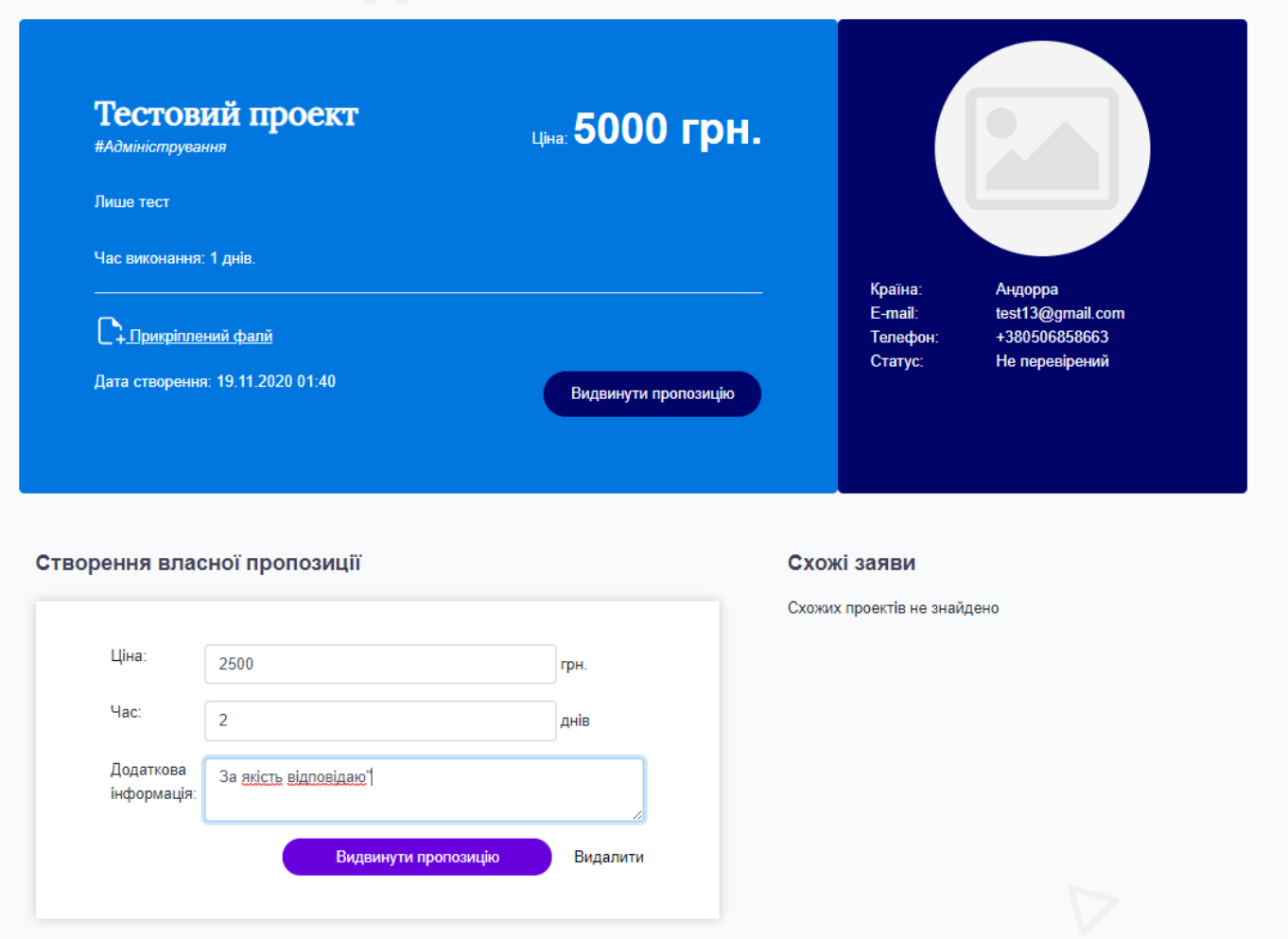

Рисунок 4.15 – Висунення пропозиції

При обранні виконавця користувач отримує повідомлення, що представлено на рис.4.16.

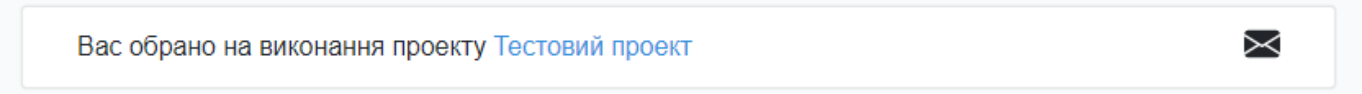

Рисунок 4.16 – Повідомлення про обрання на виконання проекту

Важливою відмінністю даної інформаційної системи є створення портфоліо. Розглянемо сторінку створення роботи для портфоліо та його відображення на сайті (рис. 4.17-4.19).

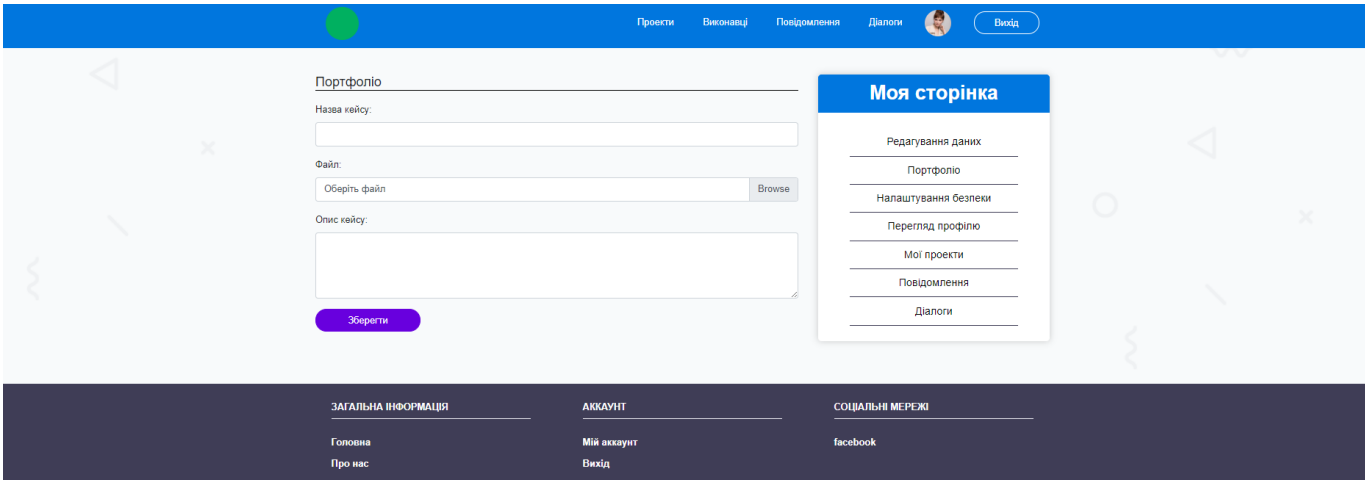

Рисунок 4.17 – Сторінка редагування портфоліо

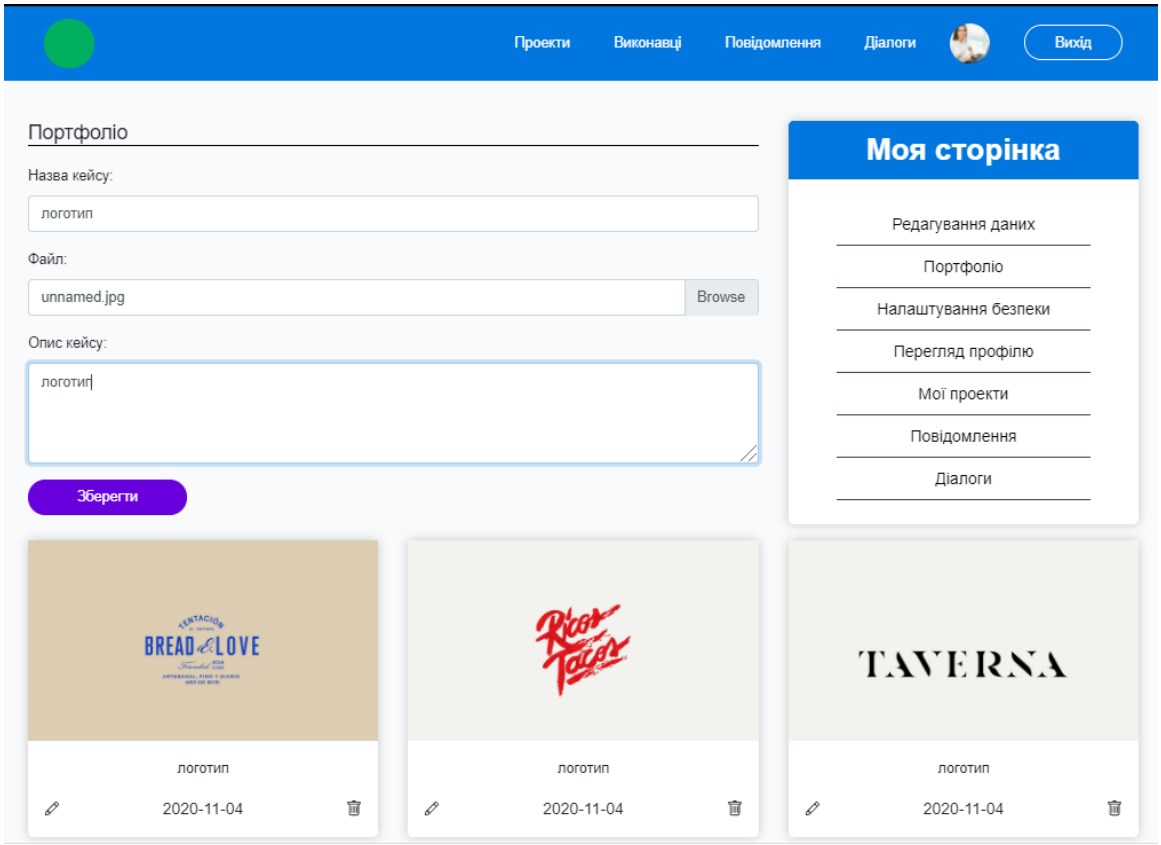

Рисунок 4.18 – Приклад створення роботи в портфоліо

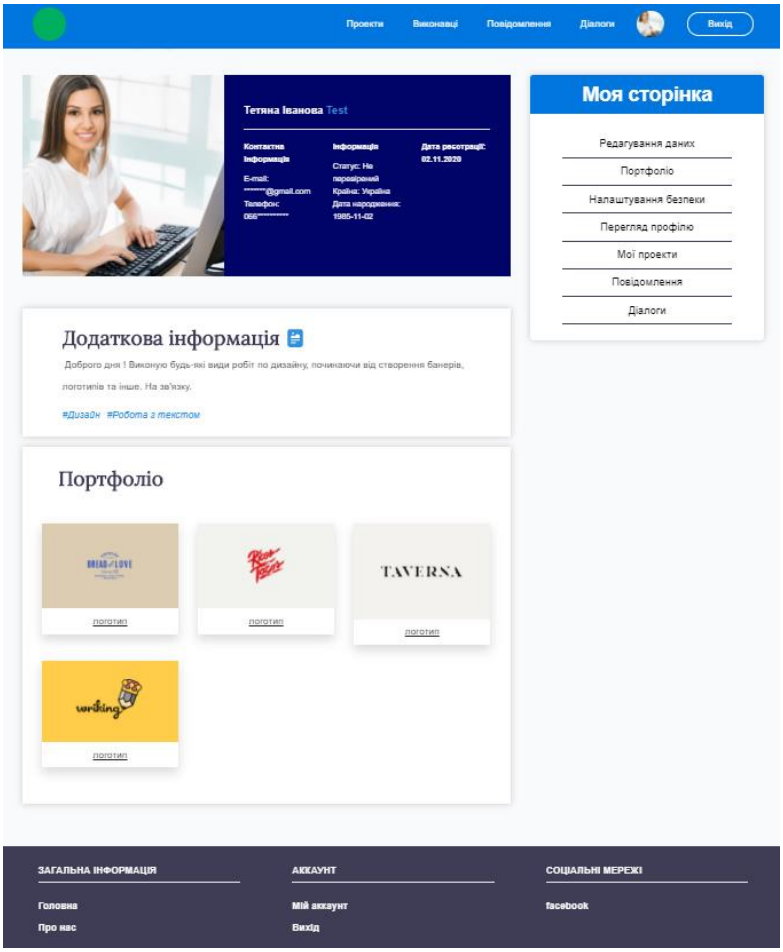

Рисунок 4.19 – Відображення портфоліо на сторінці

Користувач типу «виконавець» на вкладці Редагування даних має можливість обрати «Діяльність», за якою він буде виконувати замовлення та ввести інформацію про себе на вкладці «Додаткова інформація» (рис. 4.20-4.21).

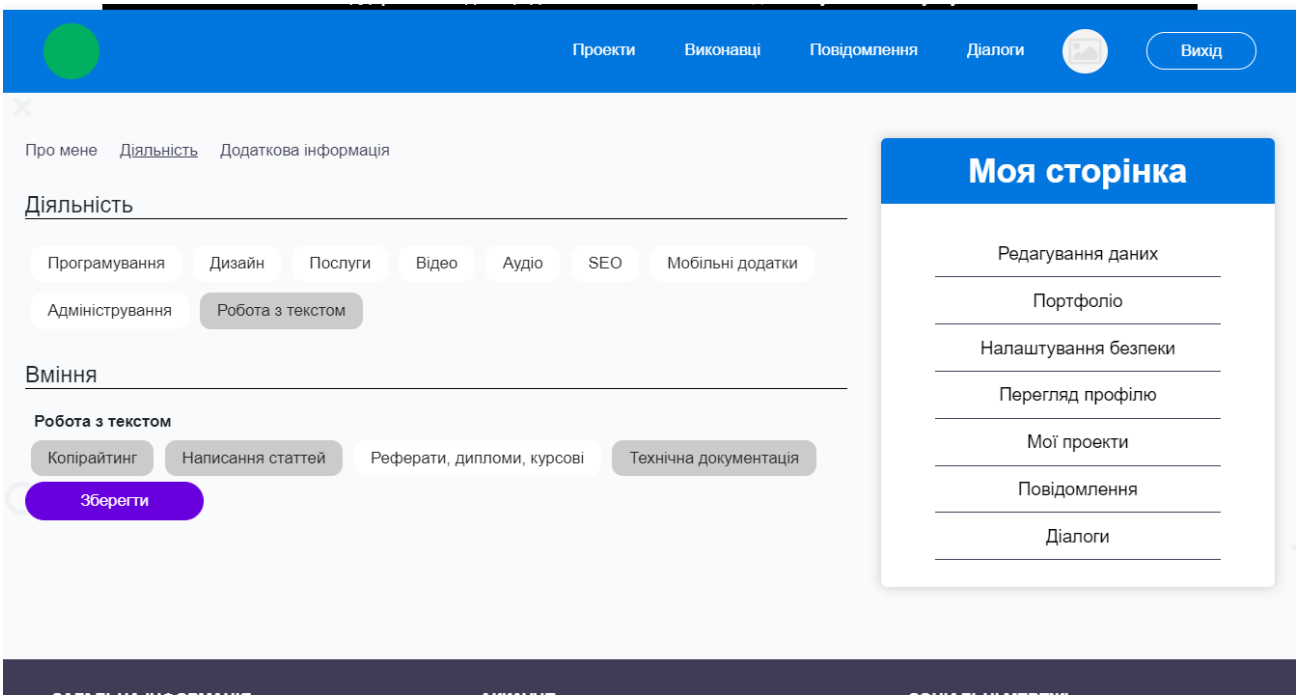

Рисунок 4.20 – Інформація про діяльність виконавця

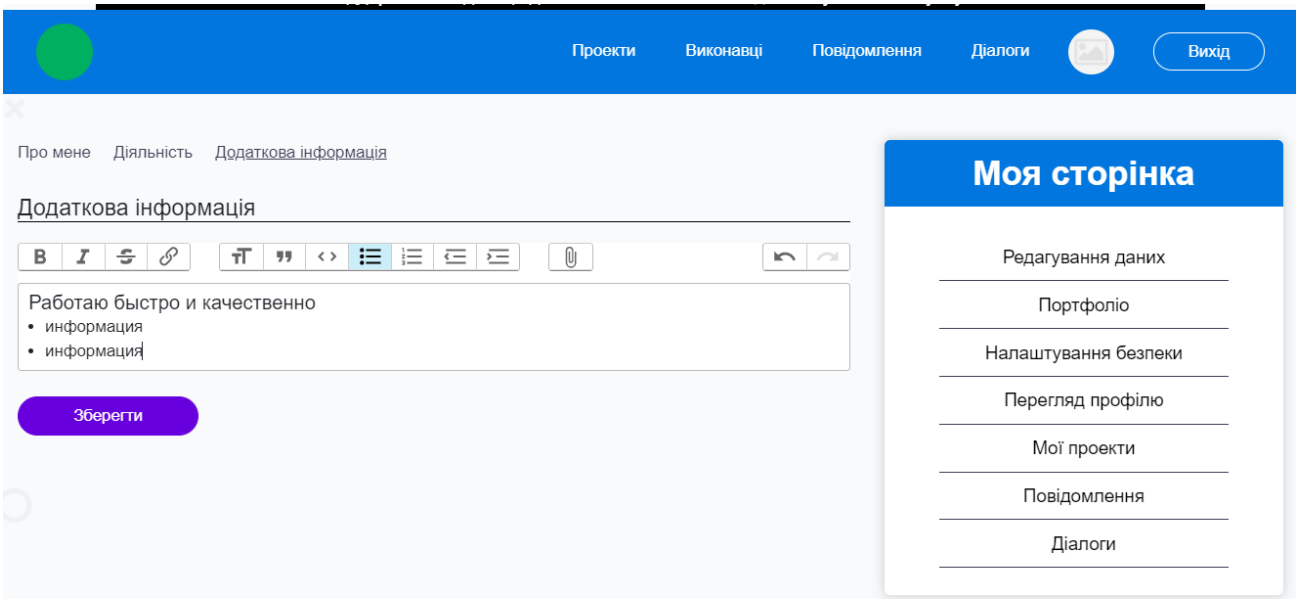

Рисунок 4.21 – Додаткова інформація про виконавця

Базуючись на усій інформації, на особистій сторінці користувача можна створити резюме. Сформоване резюме відкриється на новій вкладці браузера, звідки є можливість його завантаження або роздруковування. Кнопка знаходиться на сторінці аккаунту поряд із «Додаткова інформація» (рис.4.22-4.23).

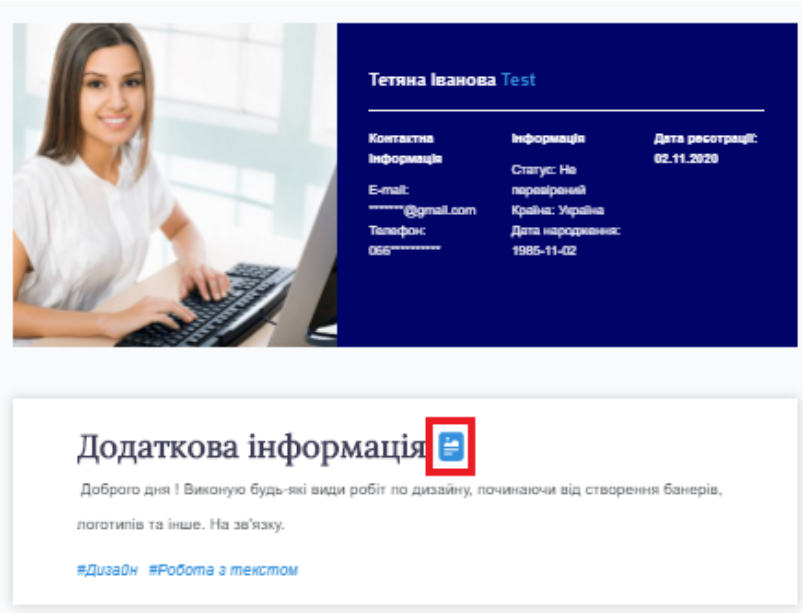

Рисунок 4.22 – Кнопка формування резюме

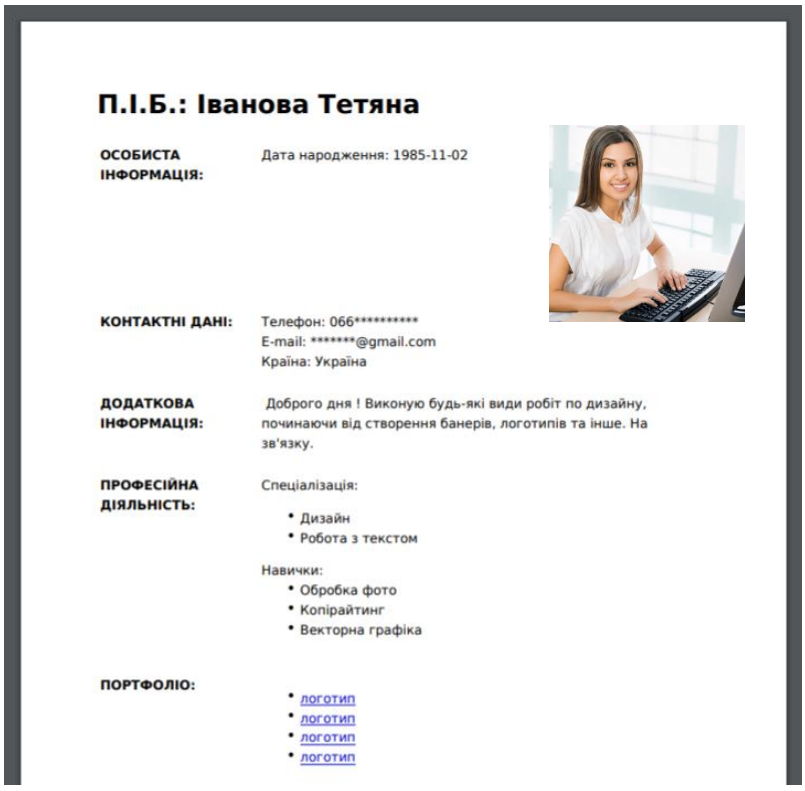

Рисунок 4.23 – Сформоване резюме

При реєстрації можна обрати тип аккаунту «Команда» (рис.4.24) в якому є можливість в майбутньому додавати користувачів до своєї команди. Перейшовши на сторінку користувача потрібно клацнути на відповідну іконку (рис.4.25-4.26).

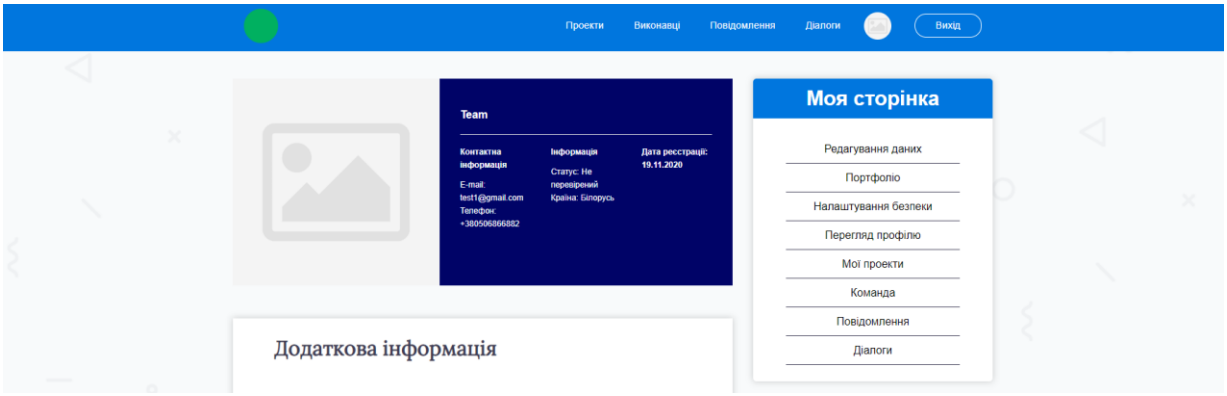

Рисунок 4.24 – Аккаунт команди

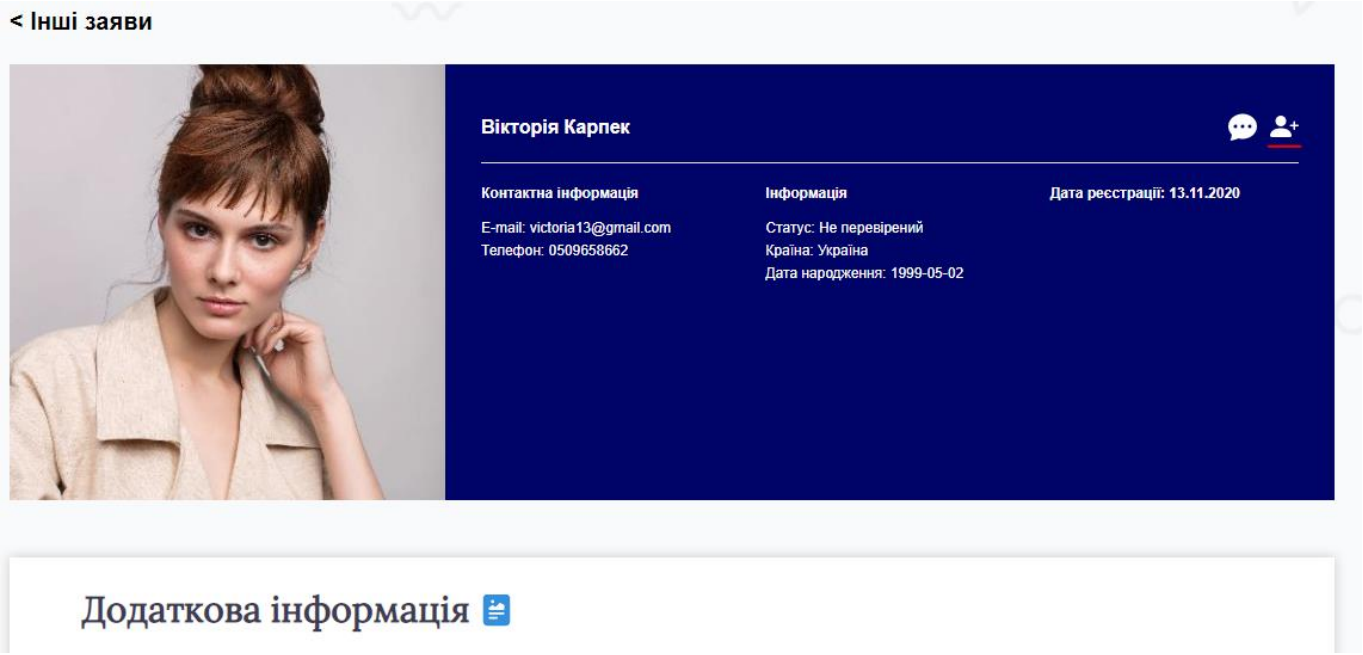

Рисунок 4.25 – Можливість додання виконавця до команди

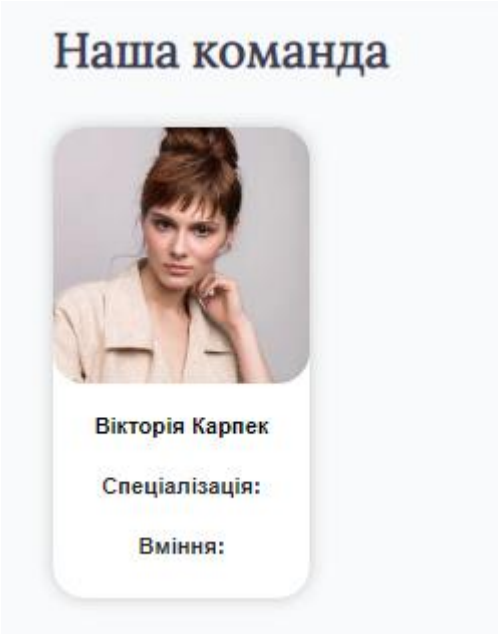

Рисунок 4.26 – Відображення команди на сторінці

4.4 Адміністрування сайту

Для переходу до адміністративної панелі, потрібно перейти на сторінку авторизації (рис.4.27).

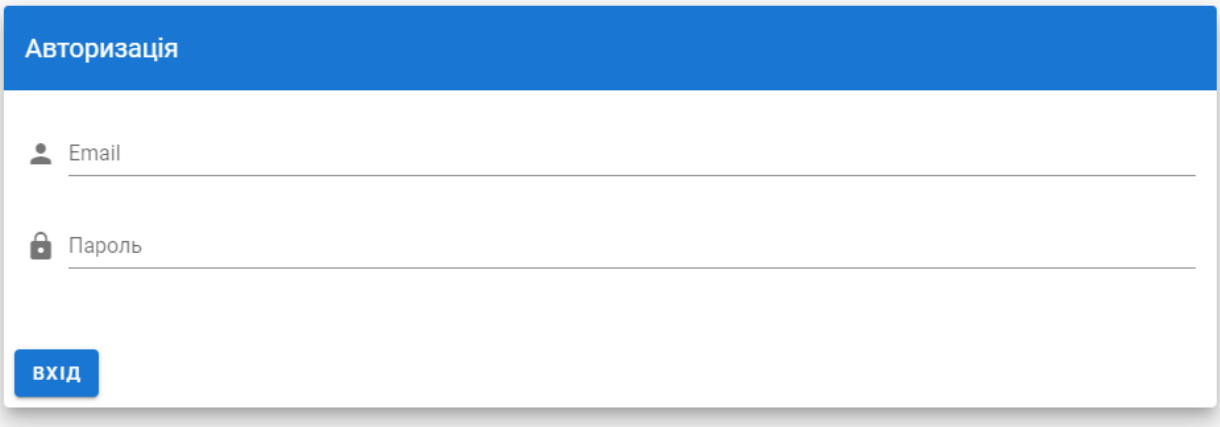

Рисунок 4.27 – Сторінка авторизація адміністратора

Адміністратор має можливість редагувати інформацію на сайті, а саме: редагувати контент, додавання або видалення спеціалізацій та вмінь користувачів, видаляти проекти та змінювати статус користувачів системи (рис.4.28-4.30).

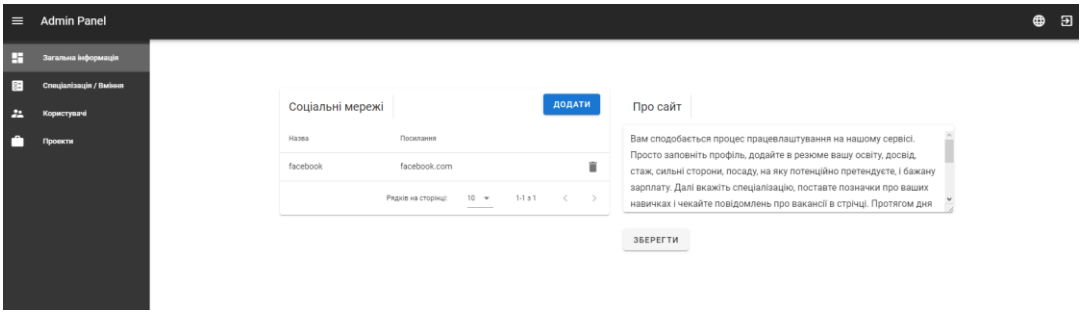

Рисунок 4.28 – Редагування загальної інформації на сайті

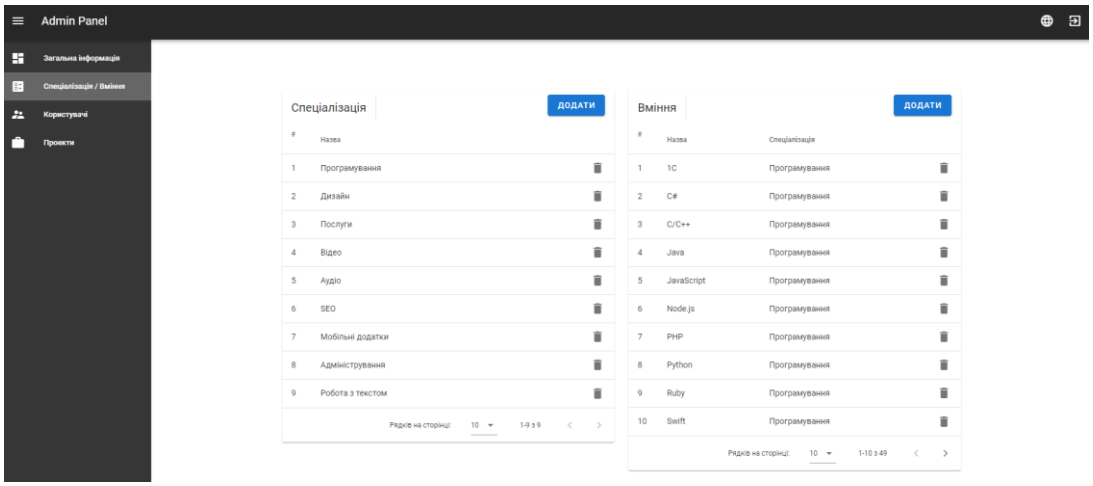

Рисунок 4.29 – Додавання та видалення спеціалізацій

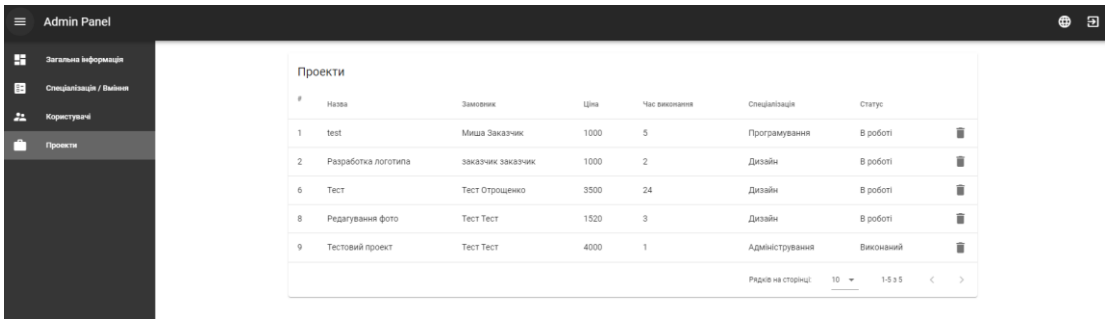

Рисунок 4.30 – Список проектів на сайті та їх статус

ВИСНОВКИ

На сьогодні фріланс перебуває у розквіті, особливо на ринку надання IT-послуг. Існує чимало спеціальних веб-сайтів, які допомогають фрілансерам отримати чергову роботу. Проаналізувавши деякі з них, а також визначивши їх функціональні можливості та недоліки, було визначена важливість розробки роботи в цьому напрямку, і, як результат, вирішено створити власну веб-орієнтовану інформаційну систему пошуку виконавців IT-проектів.

При реалізації даного проекту були вирішені такі задачі як: аналіз предметної області та формування вимог до створюваної системи; були обрані методи та інструменти реалізації веб-орієнтованої інформаційної системи; змодельовано інформаційну систему, спроектовано її інтерфейс та структуру бази даних; реалізовано пошуку виконавців IT-проектів і протестовано її роботу.

Основними функціями розробленої системи є реєстрація та авторизація користувачів, розміщення на сайті завдань замовників з можливістю прикріплення файлів, перегляд завдань зареєстрованими користувачами, створення пошуку для формування списку робіт, створення заявок на отримання замовлення, можливість спілкування між замовником та виконавцем, формування оповіщень, формування рейтингу та історії проекту, блокування користувачів.

Для виконавця система дозволяє виконувати замовлення, створювати портфоліо та спілкуватися з замовниками.

Для клієнта – створити замовлення в інформаційній системі, обрати виконавця та спілкування з виконавцем за допомогою чату. Головною особливістю створеного сервісу є можливість створення аккаунту, як для особистого використання, так й для команди розробників чи компанії. Дана можливість значно розширює можливості роботи та підвищує можливості користувача. Можна отримати повний пакет послуг, витративши на пошуки менше зусиль.

Розроблена веб-орієнтована система пошуку виконавців ІТ-проектів буде використана як відкрита фріланс-платформа, яка повинна полегшити знаходження замовлення та комунікацію між клієнтами та виконавцями. Інтерфейс веб-системи досить зручний, зрозумілий і надійний у використанні, а забезпечення захисту даних дозволяє перехід на відповідні сторінки тільки авторизованим користувачам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Freelancer: A Conceptual Review [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://www.researchgate.net/figure/Top-hiring-countries-and-top-freelance-countries_fig3_282729746

2. Freelancing Pros and Cons [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.thebalancecareers.com/should-you-work-for-yourself-4035267

3. Future of Work in a Digital Era: The Potential and Challenges for Online Freelancing and Microwork in India [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://icrier.org/pdf/Online_Freelancing%20_ICRIER.pdf

4. 15 Important Trends In Freelancing: Why This Matters Now Cons [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.forbes.com/sites/jonyounger/2018/10/11/fifteen-important-trends-in-freelancing-why-this-matters-now/#60f009703c10

5. A Study on The Freelancing Remote Job Websites. Dr. N. Fathima Thabassum [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.cscjournals.org/manuscript/Journals/IJBRM/Volume4/Issue1/IJBRM-141.pdf

6. Фриланс-сервис Freelancehunt.com Cons [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://freelancehunt.com/blog/freelancehunt-com-frilans-siervis-dlia-poiska-i-vzaimodieistviia-frilansierov-i-zakazchikov/

7. The pandemic has boosted freelance work — and hiring for these jobs is booming [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.cnbc.com/2020/07/07/freelance-work-grows-amid-covid-19-math-stats-game-hiring-in-demand.html

8. Freelanser are driving the economy. Let’s act like it. [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.freelancersunion.org/

9. The Hoxby Collective: Putting Heart And Soul Into The Freelance Revolution [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.forbes.com/sites/jonyounger/2019/06/01/the-hoxby-collective-putting-heart-and-soul-into-the-freelance-revolution/?sh=2b630ae47958

10. Огляд безкоштовної біржі фрілансу Freelancehunt [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://awayne.biz/obzor-birzhi-frilansa-freelancehunt/

11. Сайт Weblancer - популярна біржа фріланса в Україні та СНД Freelancehunt [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: http://tods-blog.com.ua/freelance/weblancer/

12. Freelancer.com Reviews [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.trustradius.com/products/freelancer-com/reviews?qs=pros-and-cons

13. Pros and Cons of Using UpWork (as a Freelancer) [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/pros-cons-using-upwork-freelancer-andrew-debell/

14. Топ-5 JS-фреймворків для фронтенд-розробки в 2020 році [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/476286/

15. React JS Pros and Cons in 2020 [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://pagepro.co/blog/react-js-pros-and-cons-in-2020/

16. React vs Vue: What is the best choice for 2020? [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.mindk.com/blog/react-vs-vue/

17. The pros and cons of choosing AngularJS [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://jaxenter.com/the-pros-and-cons-of-choosing-angularjs-124850.html

18. Top 8 Best Backend Frameworks AngularJS [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.keycdn.com/blog/best-backend-frameworks

19. Advantages and Disadvantages of Laravel AngularJS [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://medium.com/@saschathattil/advantages-and-disadvantages-of-laravel-224ecc09021a

20. The Good and the Bad of Node.js Web App Development [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/

21. Spring Boot or Not to Spring Boot? [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://rabbitstack.github.io/spring/spring-boot-or-not-to-spring-boot/ (дата звернення: 25.10.2020).

22. INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0) . Draft Federal Information Processing Standards Publication 183 ,1993 December 21

23. INTEGRATION DEFINITION FOR INFORMATION MODELING (IDEF1X), Draft Federal Information Processing Standards Publication 184 1993 December

24. Sheer, A. ARIS-Business Process Modelling / Sheer A. – Springer-Verlag, Berlin,1998.

25. Booch, G. The Unified Modeling Language User Guide / Booch, G., Rumbaugh, J., Jacobson, I. – Second Edition, AddisonWesley, 2005.

26. Методичні вказівки до лабораторних робіт з курсу «Організація баз даних та баз знань». Укладач В.О. Нелюбов. – Ужгород: Видавничий центр ЗакДУ, 2010.

27. Система управління базами даних Access. Навчальний посібник «Організація баз даних і баз знань»/ Укладач В.О. Нелюбов. – Ужгород: Редакційно-видавничій відділ, 2015.

28. Круг Стів. Вебдизайн: Книга Стіва Кола або не змушуй мене думати [Текст] / Стів Круг. — Символ-Плюс, 2001.

29. Райс Ерік. Інформаційно архітектурний підхід до створення успішних веб-сайтів. [Текст] / Ерік Райс. — Addison Wesley, 2000

Додаток А. Планування робіт

А.1 Ідентифікація мети проекту

Завжди виникають завдання, які в силу тих чи інших причин виконати складно. Причини різні - не вистачає знань, немає часу, немає бажання витрачати час на виконання одноманітної і нудної роботи. І не важливо, чи є ви співробітником фірми, власником власного бізнесу або студентом. В цьому випадку і при наявності грошей на картці, можливо зареєструватися на біржі віддаленої роботи та розмістити своє завдання в якості замовника.

З іншого боку людина, яка, в даний момент має вільний час і вміє щось робити, також може зареєструватися на сайті і, як виконавець, обрати собі відповідне завдання, виконати його та заробити гроші.

Отже, створювана інформаційна система - це посередник, яка зможе підібрати виконавця для ваших завдань у зручний для вас час і за призначену вами ціну, а також повинна полегшити знаходження замовлення та комунікацію між клієтами та фрілансерами.

Для конкретизації мети проекту було використано технологію SMART. SMART є абревіатурою, розшифровка якої: Specific, Measurable, Achievable, Relevant, Time bound.

У табл. А.1 відображений результат конкретизації мети проекту з використанням методології SMART.

Таблиця А.1 – Конкретизація мети проекту методологією SMART

Specific (конкретна)	Розробка веб-орієнтованої інформаційної системи для пошуку виконавців IT-проектів.
Measurable (вимірювана)	Результативність вимірюється показником кількості зареєстрованих осіб в інформаційній системі, а також кількості осіб, які знайшли виконавців для своїх проектів та кількості залишених відгуків.
Achievable (досяжна)	Реалізація інформаційної системи виконується за допомогою HTML5, CSS3, JavaScript, Vue.js, BootstrapVue - front-end; MySQL, PHP, Laravel - back-end.
Relevant (реалістична)	Розробник проекту має достатній рівень знань, програмне й апаратне забезпечення для реалізації веб-орієнтованої інформаційної системи та її подальшої підтримки.
Time-framed (обмежена у часі)	Інформаційна система має бути створена та протестована згідно з календарним планом.

Замовники мають можливість швидко знайти за допомогою біржі фахівця, який впорається з поставленим завданням краще і швидше. А якщо у вас кілька завдань, то виконувати їх одному виконавцю або вибрати кількох, - це вже вам вирішувати. Біржу віддаленої роботи можна використовувати як ваш особистий відділ кадрів. При цьому ви можете повністю зосередитися на більш важливих своїх справах.

Для виконання ваших завдань будуть підібрані виконавці з найбільш високим рейтингом, тобто це означає, що даний виконавець вже виконував якісно подібні завдання. Можна почитати відгуки інших замовників про його роботі і зробити вибір. Тільки після того, як ви переконалися в правильності і якості

виконання поставленого завдання, ви робите оплату роботи. Якщо є які-небудь зауваження, ви маєте право вимагати доробити завдання. У разі виникнення суперечок, всі питання направляються в арбітраж, де фахівці біржі допомагають вирішити конфліктні ситуації.

Виконавці або фрілансери, співпрацюючи з біржею, мають можливість заробляти гроші. Буде це основним заробітком чи додатковим - ваша особиста справа, чи плануєте свій робочий графік тільки ви. Займайтеся своєю звичайною справою і вибирайте, коли буде бажання і можливість, собі завдання зі списку, який буде вам постійно надаватися. Більшість завдань не вимагає високої спеціалізації, і виконати їх може практично кожен. Але від якості виконаної роботи, кількості виконаних завдань буде залежати ваш рейтинг. Проявивши свої знання і вміння, ви можете отримати можливість виконувати більш складні завдання, а це, в поєднанні з високим рейтингом, веде до більш серйозного заробітку.

Для роботи з системою передбачено 3 групи користувачів, які повинні пройти авторизацію:

- Адміністратори ІС;
- Замовники;
- Виконавці.

Основні функції ІС

1. Реєстрація та авторизація користувачів.
2. Розміщення на сайті завдань замовників з можливістю прикріплення файлів.
3. Перегляд завдань зареєстрованими користувачами.
4. Створення пошуку для формування списку робіт за різними критеріями.
5. Створення заявок на отримання роботи з можливістю прикріплення файлів.
6. Можливість спілкування між замовником та виконавцем за допомогою чату.
7. Формування рейтингу та історії проектів.

8. Формування оповіщень.
9. Блокування користувачів.
10. Сервіс підтримки користувачів.
11. Оплата

А.2 Планування змісту структури робіт IT-проекту

А.2.1. Планування змісту структури робіт IT-проекту (WBS)

Ті, хто має справу зі складними IT-проектами керівники підтвердять, що поділ завдань на більш дрібні і керовані частини робить робочий процес набагато простіше. Процес, який допоможе структурувати кожен етап проекту і враховувати всі поставлені завдання це ієрархічна структура робіт WBS (Work Breakdown Structure).

WBS заснована на графічній природі, яка допомагає менеджерам проектів передбачити результати, засновані на різних сценаріях. Процес часто описується як структура відгалуження, яка охоплює всі етапи проекту в організованому порядку. WBS також може бути представлена у вигляді табличного списку завдань і елементів в плані розбивки робіт діаграм Ганта.

Менеджери використовують структуру декомпозиції, щоб структурувати і ділити проекти на легко керовані компоненти. Вони, в свою чергу, поділяються до тих пір, поки вони не призначаються конкретного фахівця в команді.

Загальна WBS структура представлена на рис. А.1.

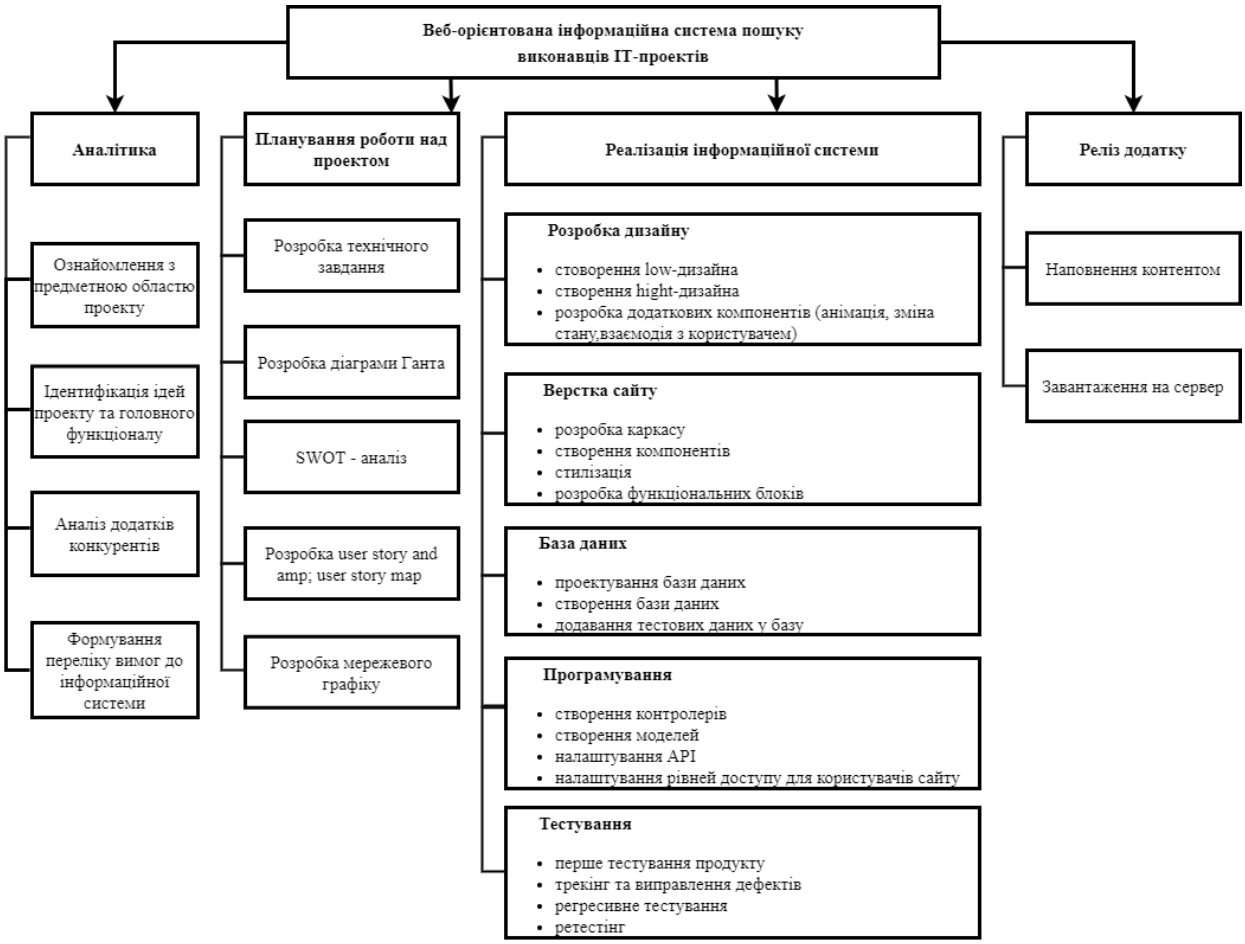

Рисунок А.1 – WBS структура проекту

А.2.2. Планування структури організації (OBS)

Структура розподілу організації або OBS – це ієрархічна модель, що описує встановлені організаційні рамки для планування проектів, управління ресурсами, відстеження часу та витрат, розподілу витрат, звітування про доходи / прибуток та управління роботою.

Загальна OBS структура представлена на рис А.2.

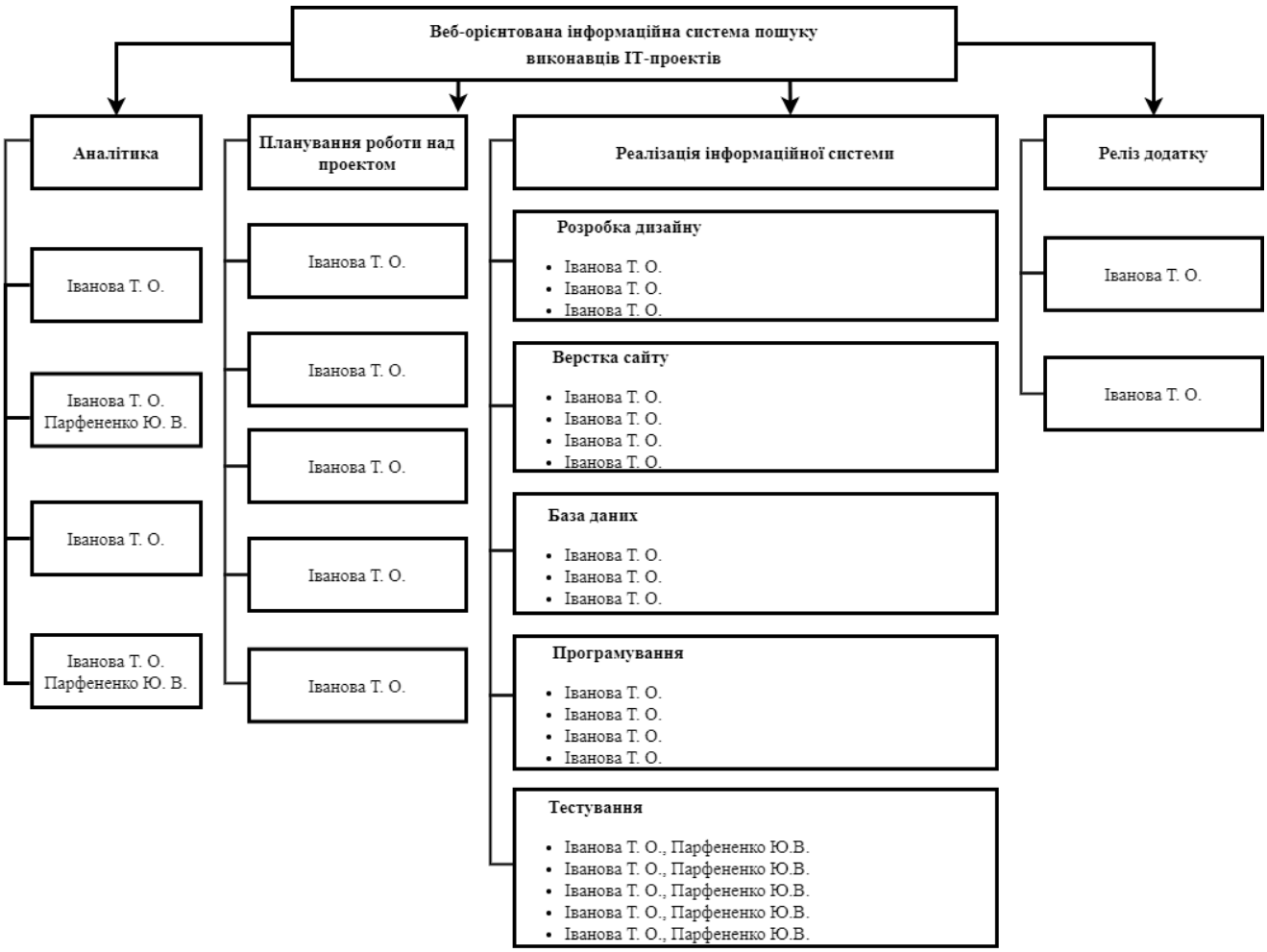

Рисунок А.2 – OBS структура проекту

А.3 Побудова календарного графіку виконання ІТ-проекту

Діаграма Ганта - це горизонтальна діаграма з тимчасової шкалою, яка використовується для ілюстрації плану робіт за проектом з прив'язкою до часу.

За допомогою діаграм Ганта керівники проектів і менеджери по продукту розбивають проекти на робочі завдання для зручності управління, підтримують порядок в роботі і роблять залежності між завданнями наочними.

Діаграми Ганта дозволяють спростити складові проекти. За допомогою цього засобу можна досить наочно і зручно для узагальнення представити велику кількість даних. Завдяки цій діограмі велика кількість зацікавлених осіб, команд або їх учасників не стане проблемою для запису завдань, як і часті зміни обсягу роботи. Ще одна перевага використання діаграми Ганта полягає в тому, що вона дає загальне уявлення про проект в цілому, в тому числі про всі контрольних точках і терміни виконання. Діаграму Ганта можна уявити як ефективний засіб раннього попередження.

Розглянемо створену діаграму Ганта до заданої інформаційної системи (рис.А.3-4).

	Длительнс	Название задачи	Названия ресурсов
1	**65 дней**	**«Розробка інформаційної системи для фріланс біржи»**	**Іванова Т.О.; Парфененко Ю.В.**
2	**4 дней**	**1 Аналітика**	Іванова Т.О.;Парфененко Ю
3	2 дней	1.1 Ознайомлення з предметної областю проекту	Іванова Т.О.
4	2 дней	1.2 Ідентифікація ідей проекту та головного функціс	Іванова Т.О.;Парфененко Ю
5	1 день	1.3 Аналіз додатків конкурентів	Іванова Т.О.
6	1 день	1.4 Формування переліку вимог до інформаційної сі	Іванова Т.О.;Парфененко Ю
7	**9 дней**	**2. Планування роботи над проектом**	Іванова Т.О.
8	2 дней	2.1 Розробка технічного завдання	Іванова Т.О.
9	2 дней	2.2 Розробка Діаграмми Ганта	Іванова Т.О.
10	2 дней	2.3 SWOT-аналіз	Іванова Т.О.
11	2 дней	2.4 Розробка user story & user story map	Іванова Т.О.
12	1 день	2.5 Розробка Мережевого графіку	Іванова Т.О.
13	**47 дней**	**3. Реалізація**	Іванова Т.О.;Парфененко Ю
14	**13 дней**	**3.1 Розробка дизайну**	Іванова Т.О.
15	5 дней	3.1.1 Створення low-дизайну	Іванова Т.О.
16	5 дней	3.1.2 Створення hight-дизайну	Іванова Т.О.
17	3 дней	3.1.3 Розробка додаткових компонентів (анімаціі	Іванова Т.О.
18	**15 дней**	**3.2 Верстка сайту**	Іванова Т.О.
19	5 дней	3.2.1 Розробка каркасу	Іванова Т.О.
20	3 дней	3.2.2 Створення компонентів	Іванова Т.О.
21	2 дней	3.2.3 Стилізація	Іванова Т.О.
22	5 дней	3.2.4 Розробка функціональних блоків	Іванова Т.О.
23	**5 дней**	**3.3 База даних**	Іванова Т.О.
24	2 дней	3.3.1 Проектування бази даних	Іванова Т.О.
25	1 день	3.3.2 Створення бази даних	Іванова Т.О.
26	2 дней	3.3.3 Додання тестових даних в базу	Іванова Т.О.
27	**10 дней**	**3.4 Програмування**	Іванова Т.О.
28	2 дней	3.4.1 Створення контролерів	Іванова Т.О.
29	2 дней	3.4.2 Створення моделей	Іванова Т.О.
30	3 дней	3.4.3 Налаштування API	Іванова Т.О.
31	3 дней	3.4.4 Налаштування рівнів доступу для користувач	Іванова Т.О.
32	**4 дней**	**3.5 Тестування**	Іванова Т.О.
33	1 день	3.5.1 Перше тестування продукту	Іванова Т.О.;Парфененко Ю.
34	1 день	3.5.2 Трекінг та виправлення дефектів	Іванова Т.О.;Парфененко Ю.
35	1 день	3.5.3 Регресивне тестування	Іванова Т.О.;Парфененко Ю.
36	1 день	3.5.4 Ретестинг	Іванова Т.О.;Парфененко Ю.
37	**5 дней**	**4 Реліз додатку**	Іванова Т.О.
38	3 дней	4.1 Наповнення контентом	Іванова Т.О.
39	2 дней	4.2 Завантаження на сервер	Іванова Т.О.

Рисунок А.3 – Діаграма Ганта (інформація)

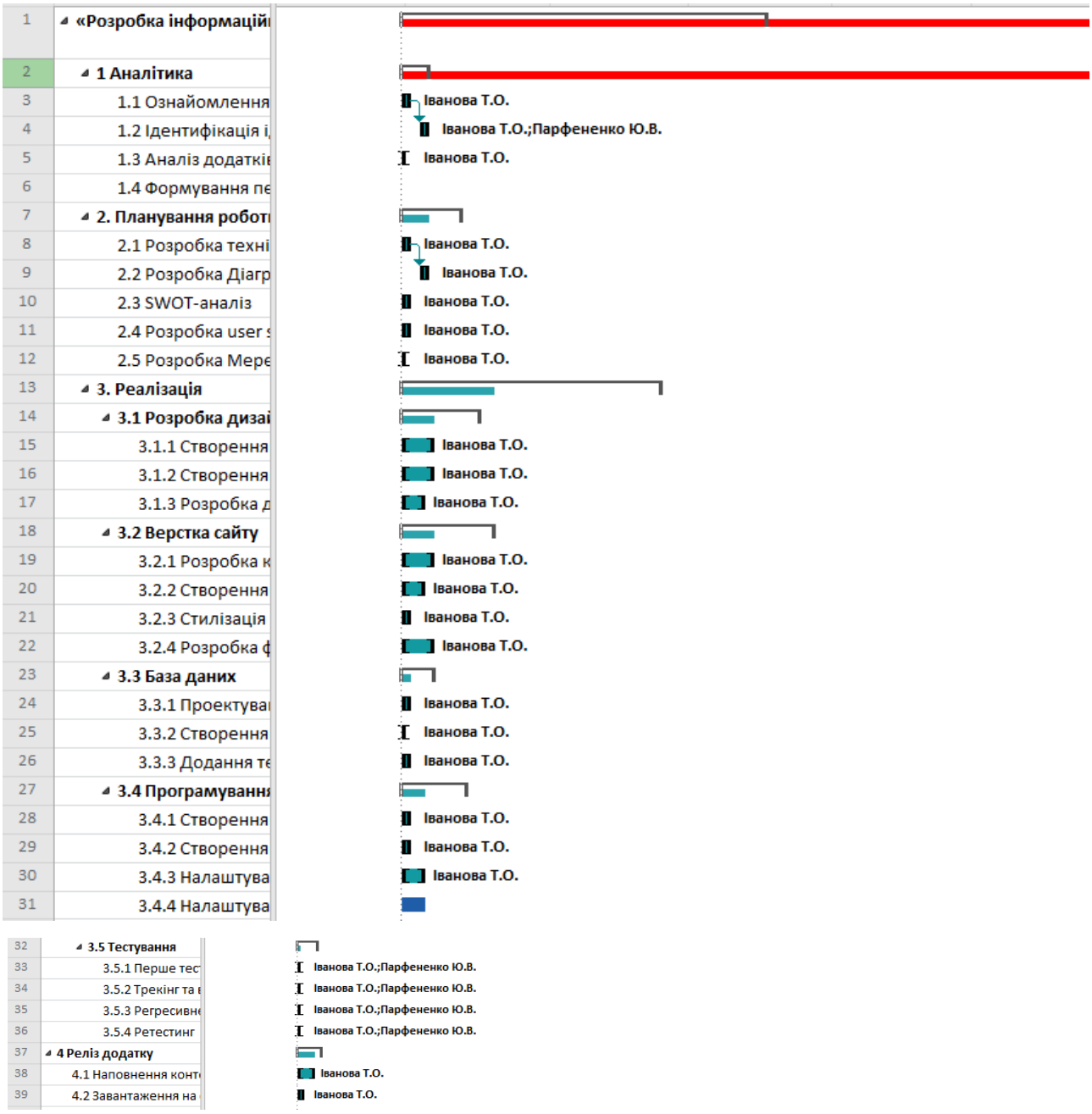

Рисунок А.4 – Діаграма Ганта (графік)

А.4 Планування ризиків проекту

Ризиком є ймовірна подія, яка у випадку її виникнення може як негативно, так і позитивно вплинути на конкретний проект. Управління ризиком – це процес зміни ризиків та реагування на події під час виконання проекту [9]. При реалізації проекту важливою частиною є моніторинг ризиків. Отже, в даному випадку можна виділити деякі ризики.

Під час процесу аналізу для визначення числових значень ймовірності появи ступеня впливу, застосовувалася методика експертних оцінок. Виходячи цих оцінок можливо знайти ранг ризиків: R = P * L, де

- R – ранг ризику;
- P – ймовірність виникнення;
- L – ступінь впливу.

Шкала оцінки ризику може відповідати емпіричній шкалі оцінки ризику:

- 5 балів - критичний ризик (0,81 - 1);
- 4 бали - максимальний ризик (0,61 - 0,8);
- 3 бали - високий ризик (0,41 - 0,6);
- 2 бали - нормальний ризик (0,31 - 0,4); 1 бал - малий ризик (0 - 0,3).

Оцінки ризиків проекту наведено в табл. А.2.

Таблиця А.2 – Таблиця ризиків

№	Об’єкт ризику	Ризик	P	L	R
1	Час	Зміна пріоритету проекту	0,15	0,3	0,15

Продовження таблиці А.2

№	Об’єкт ризику	Ризик	P	L	R
2	Якість	Системи, що не відповідають задачам , грубі помилки в алгоритмах процесів, критичні збої системи	0,1	0,4	0,04
3	Бюджет	Поява поза планових робіт по проекту	0,7	0,2	0,14
4	Трудові ресурси та їх кваліфікація	Неможливість участі в запланованих роботах по проекту необхідних співробітників зі сторони замовника та виконавця у зв’язку з відпусткою, відрядженням та ін.	0,1	0,1	0,01
5	Інтеграція	Не вірна інформація, щодо зовнішніх систем, з яких передбачена взаємодія в межах проекту	0,5	0,4	0,2
6	Ринок	Розширення функціональних характеристик програмних продуктів, які використовувалися замовником в межах мети проекту	0,1	0,05	0,005

Продовження таблиці А.2

№	Об’єкт ризику	Ризик	P	L	R
7	Ринок	Зміна вимог замовника, економічні зміни, посилення конкуренції, втрата позицій на ринку;	0,5	0,1	0,05
8	Бюджет	Зриви планів робіт; неправильна стратегія; персонал без потрібної кваліфікації; переплати за роботу/матеріали; непогоджені частини проекту; невірний кошторис.	0,5	0,2	0,1

Додаток Б. Макети сторінок інформаційної системи

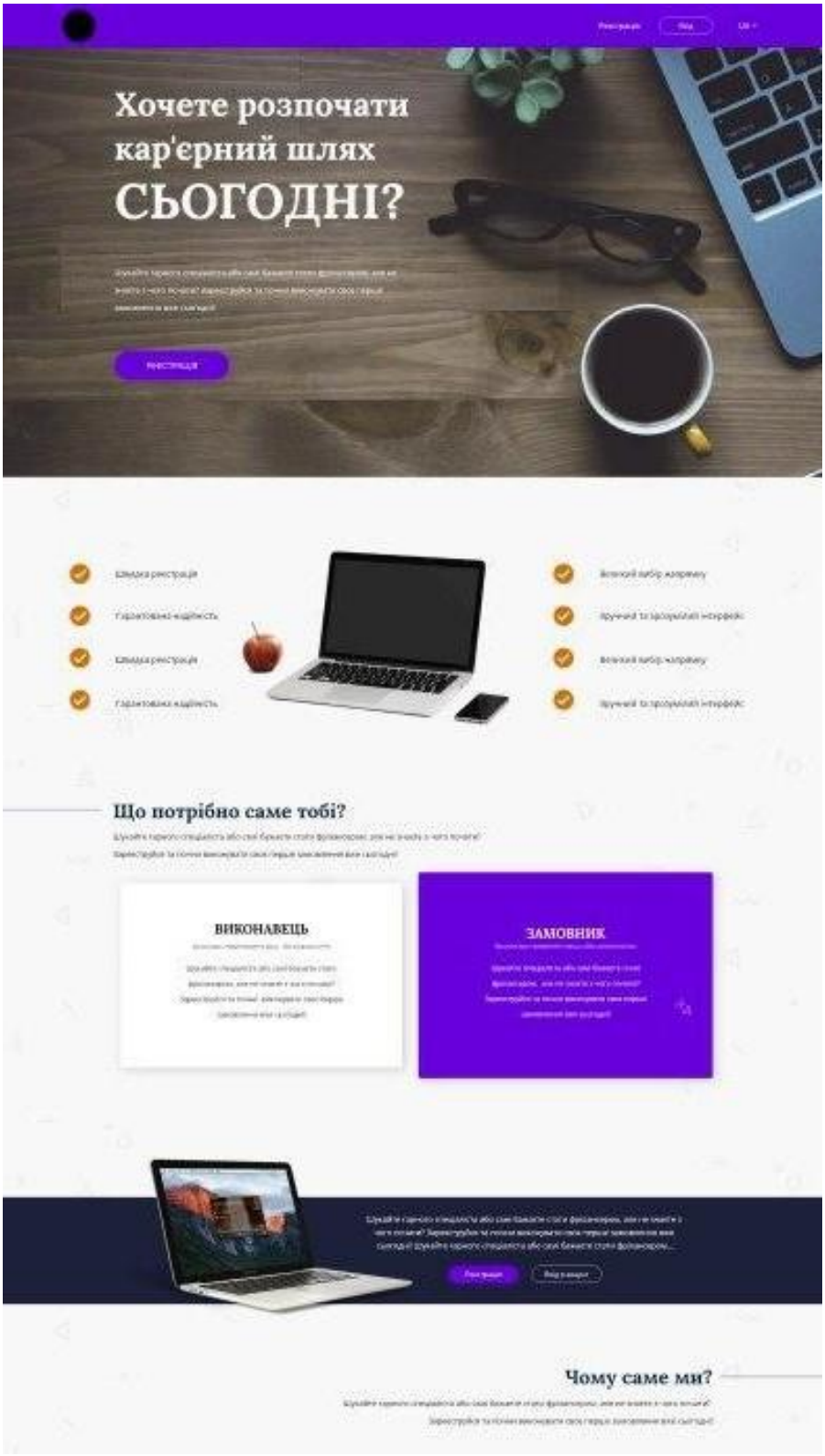

Рисунок Б.1 – Головна сторінка сайту

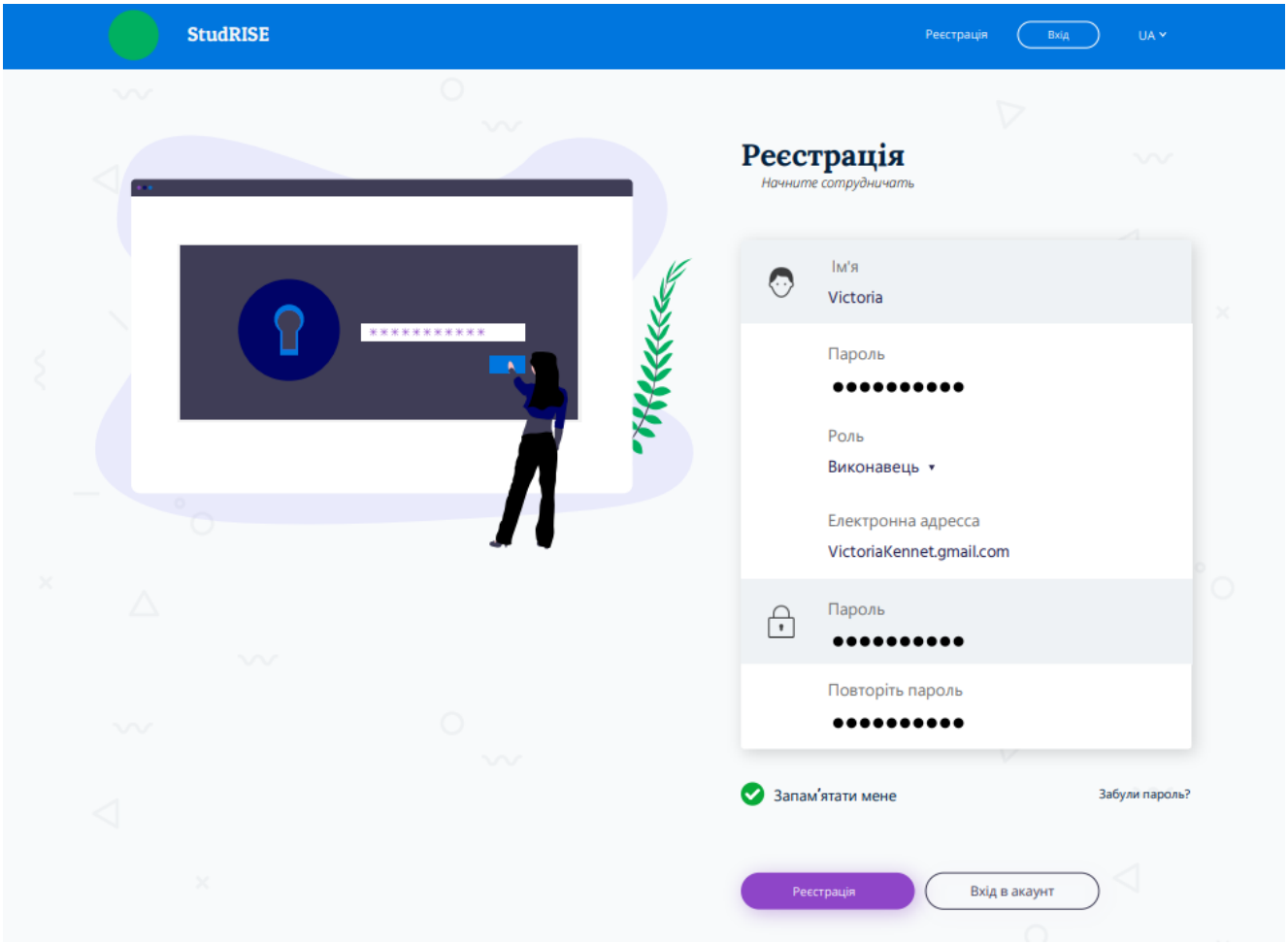

Рисунок Б.2 – Сторінка реєстрації

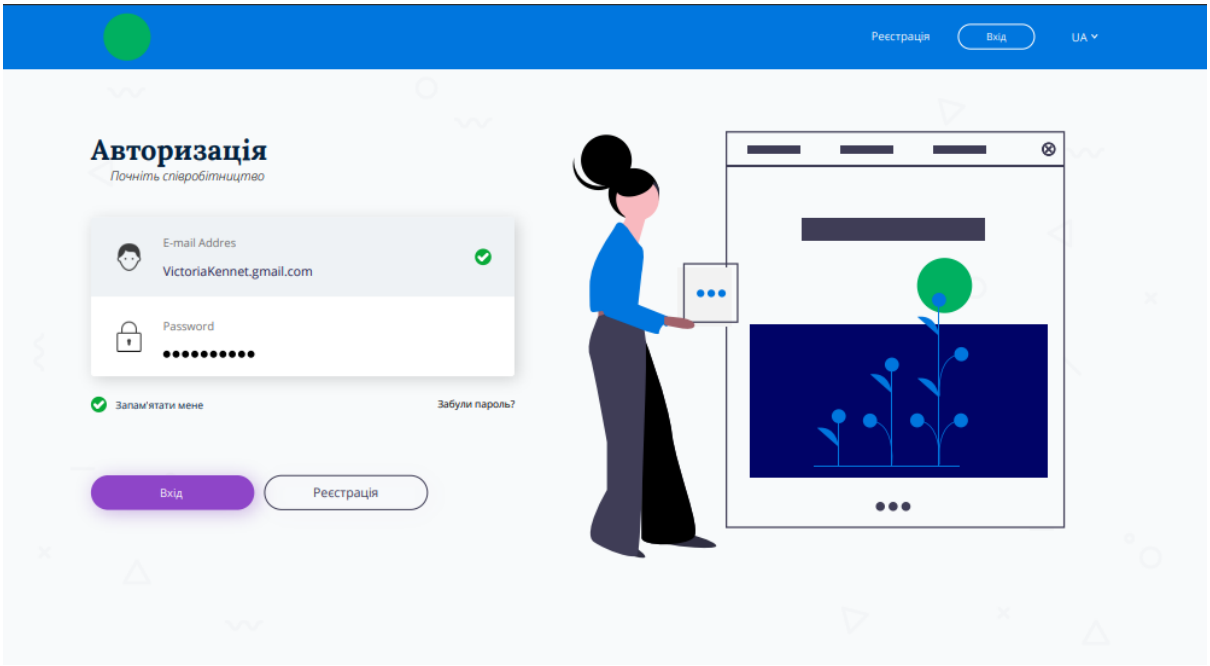

Рисунок Б.3 – Сторінка авторизації

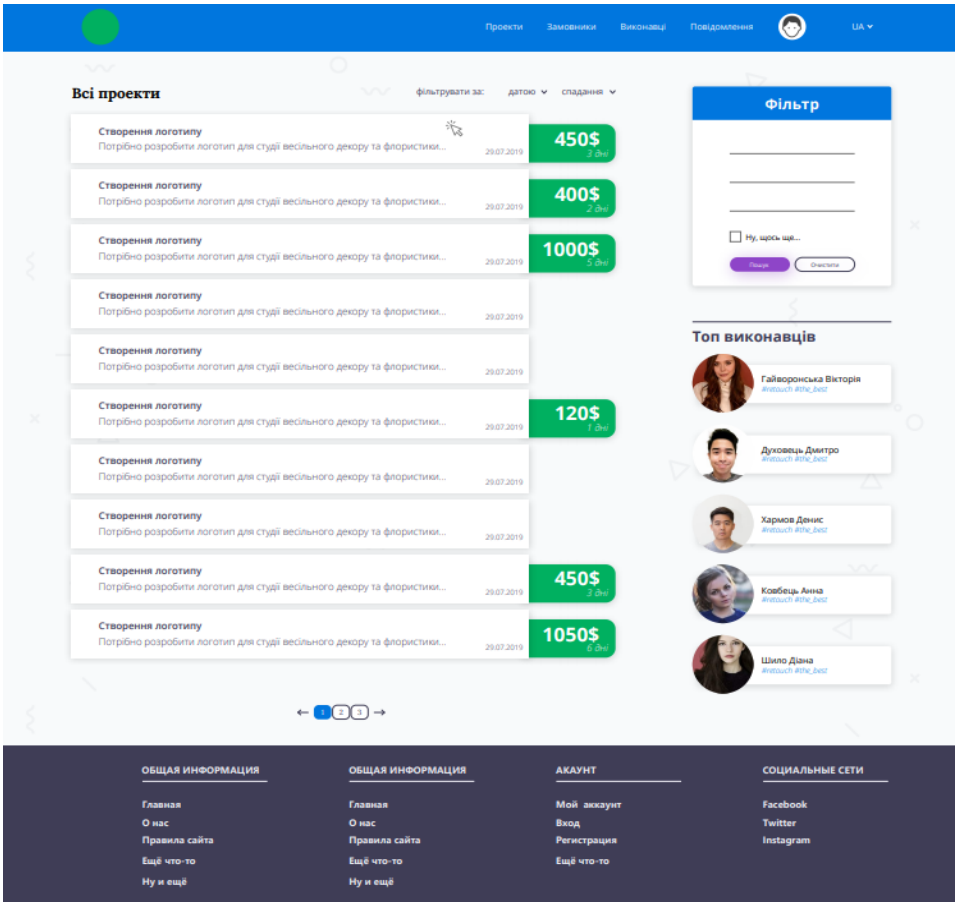

Рисунок Б.4 – Виконання пошуку проекту

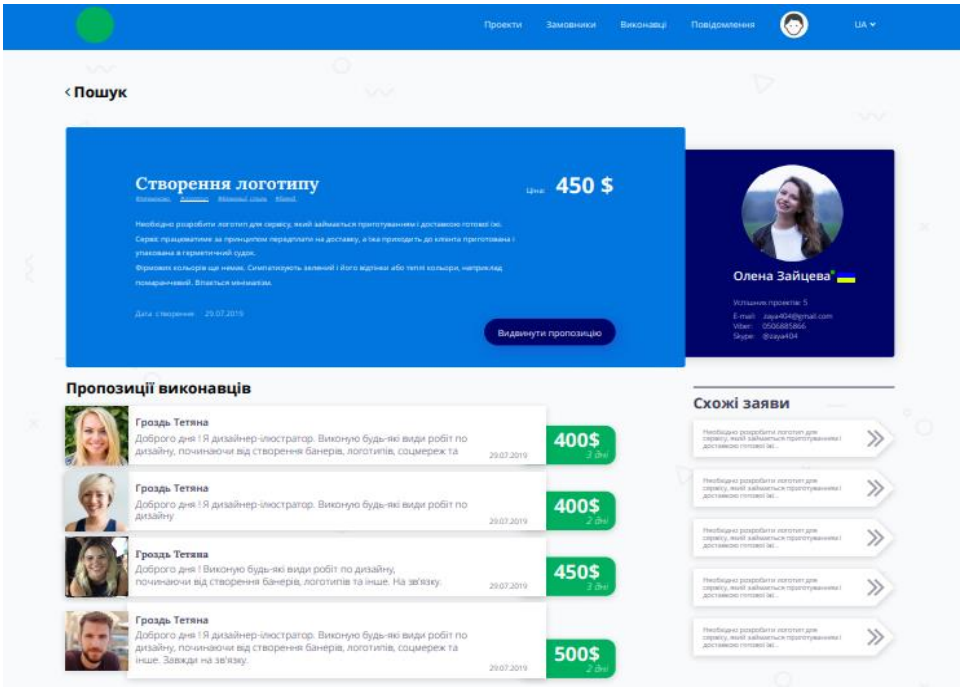

Рисунок Б.5 – Висунення пропозиції

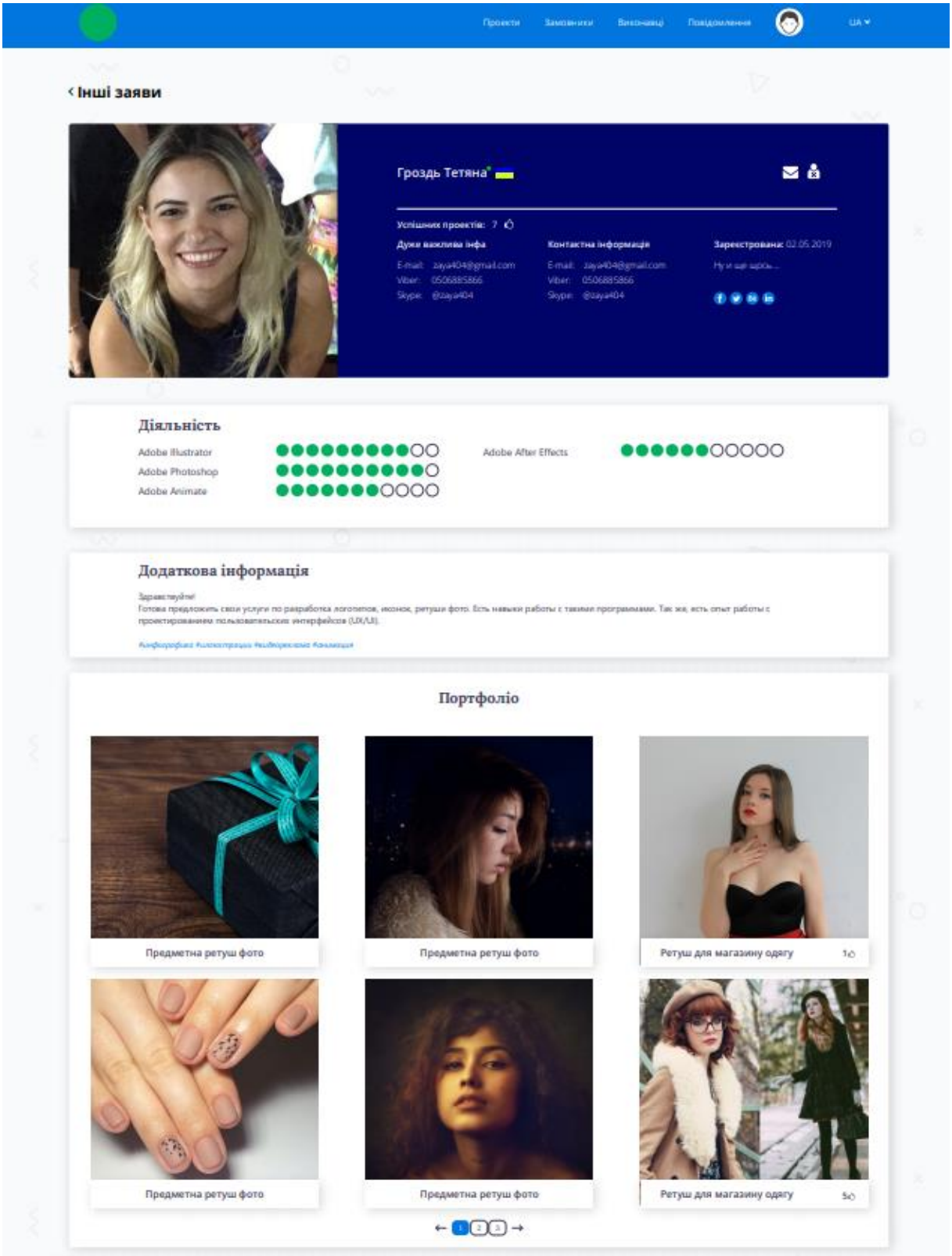

Рисунок Б.6 – Особисті дані виконавця

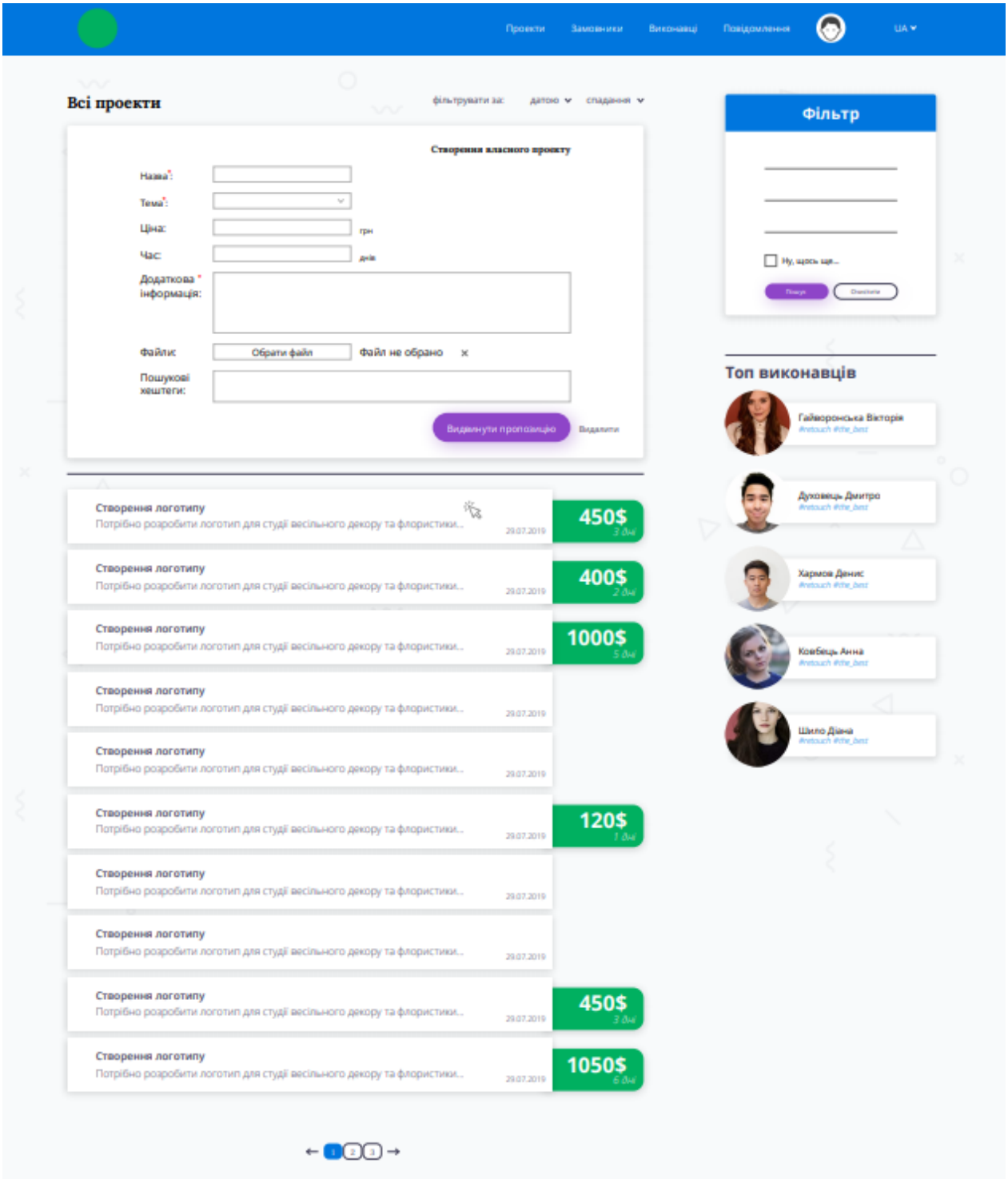

Рисунок Б.7 – Створення власного проекту

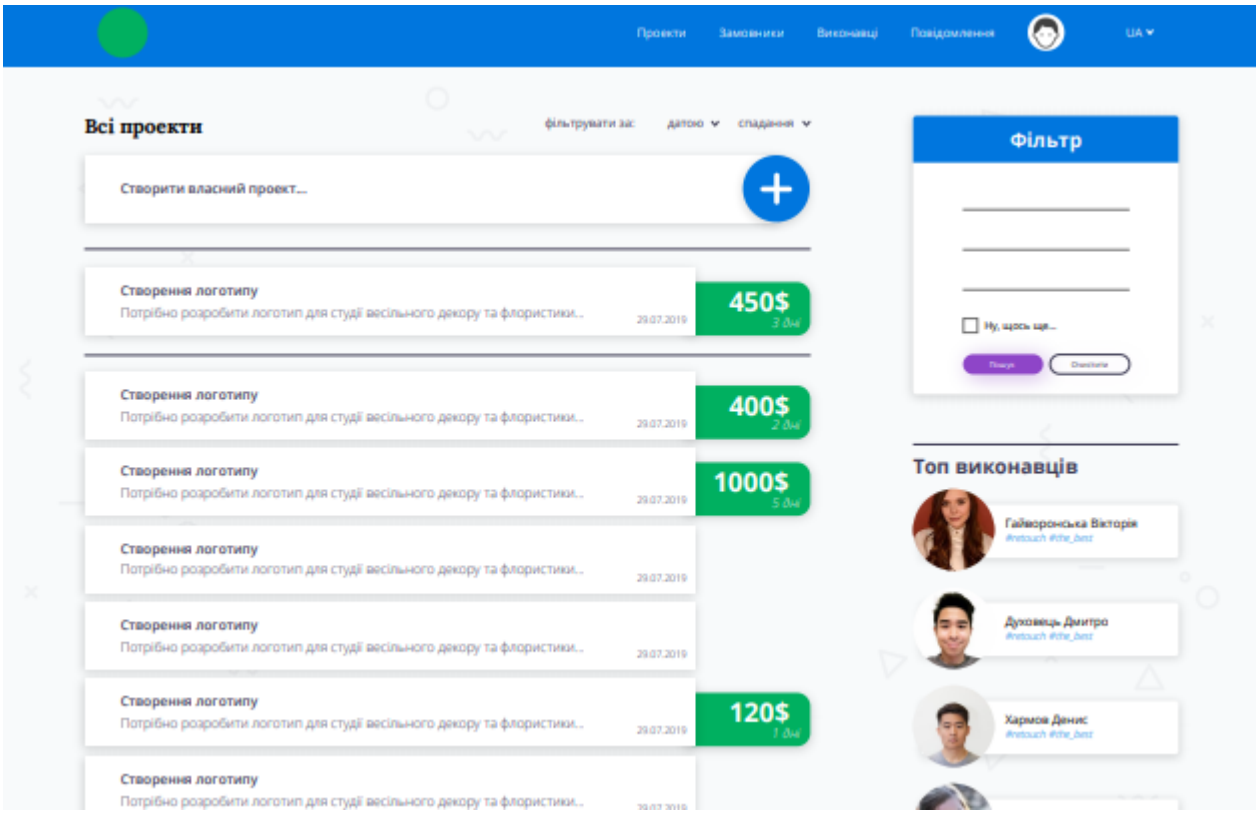

Рисунок Б.8 – Головна сторінка після авторизації

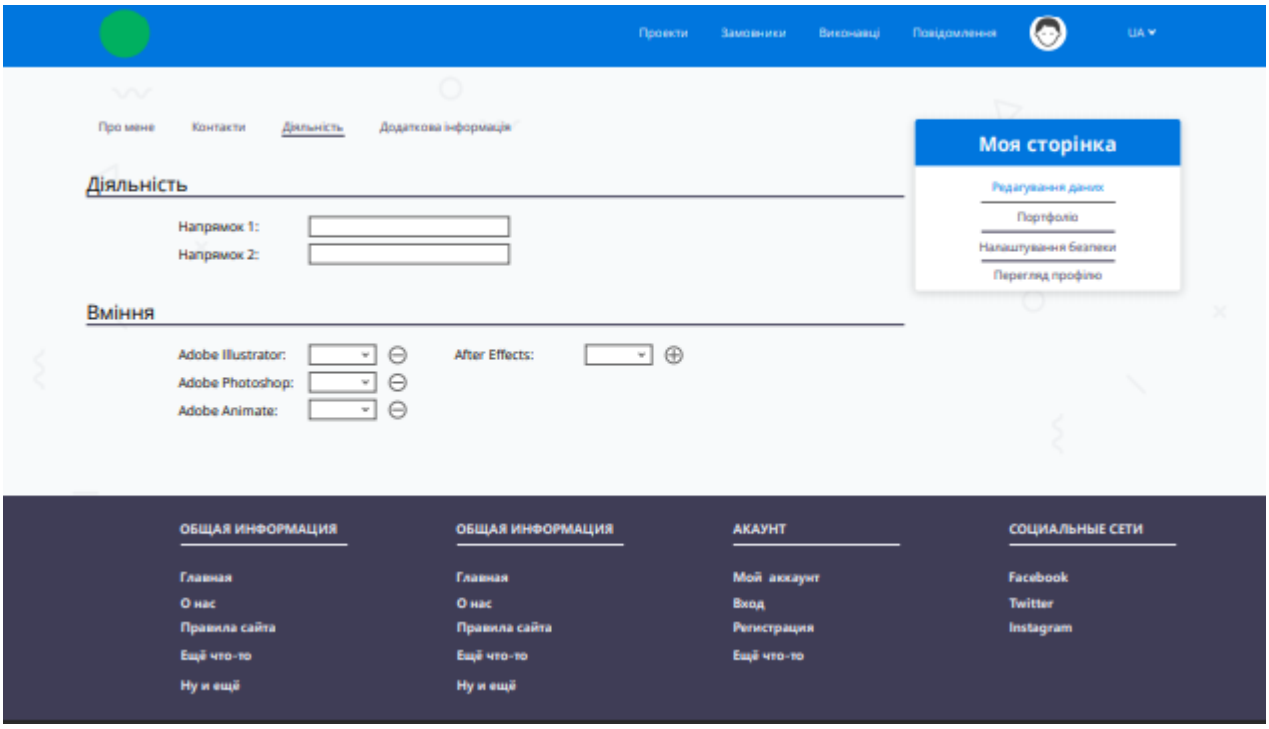

Рисунок Б.9 – Сторінка редагування даних виконавця

Додаток В

Код реалізації інформаційної системи

2020_08_05_083542_create_account_table.php

Міграція створення таблиці аккаута.

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateAccountTable extends Migration {
    public function up()
    {
        Schema::create('account', function (Blueprint $table) {
            $table->id();
            $table->string('email', 100);
            $table->string('password', 100);
            $table->string('phone', 100)->nullable();
            $table->text('description')->nullable();
            $table->string('photo', 100)->default('/img/no-image.png');
            $table->foreignId('account_status_id')->default(1);
            $table->foreignId('country_id');
            $table->foreignId('account_role_id');
            $table->timestamps();
        });
        Schema::table('account', function (Blueprint $table) {
            $table->index('account_status_id');
            $table->foreign('account_status_id')->references('id')->on('account_status');
        });
        Schema::table('account', function (Blueprint $table) {
            $table->index('country_id');
            $table->foreign('country_id')->references('id')->on('country');
        });
        Schema::table('account', function (Blueprint $table) {
            $table->index('account_role_id');
            $table->foreign('account_role_id')->references('id')->on('account_role');
        });
    }
    public function down()
```

```
{Schema::dropIfExists('account');}}
```

2020_08_05_084904_create_person_table.php

Міграція створення таблиці виконавця/замовника.

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreatePersonTable extends Migration {
  public function up()
  {
    Schema::create('person', function (Blueprint $table) {
      $table->id();
      $table->foreignId('account_id');
      $table->string('name', 100);
      $table->string('surname', 100);
      $table->string('patronymic', 100)->nullable();
      $table->date('date_bth')->nullable();
      $table->foreignId('team_id')->nullable();
      $table->foreignId('sex_id');
    });
    Schema::table('person', function (Blueprint $table) {
      $table->index('account_id');
      $table->foreign('account_id')->references('id')->on('account')->onDelete('cascade');
    });
    Schema::table('person', function (Blueprint $table) {
      $table->index('team_id');
      $table->foreign('team_id')->references('id')->on('team');
    });
    Schema::table('person', function (Blueprint $table) {
      $table->index('sex_id');
      $table->foreign('sex_id')->references('id')->on('sex');
    });
  }
  public function down()
  {Schema::dropIfExists('person');}}
```

2020_08_05_083543_create_team_table.php

Міграція створення таблиці команд.

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateTeamTable extends Migration
{
   public function up()
   {
      Schema::create('team', function (Blueprint $table) {
         $table->id();
         $table->foreignId('account_id');
         $table->string('title');
         $table->timestamps();
      });
      Schema::table('team', function (Blueprint $table) {
         $table->index('account_id');
         $table->foreign('account_id')->references('id')->on('account')->onDelete('cascade');
      });
   }
   public function down()
   {Schema::dropIfExists('team');}}
```

2020_08_05_083543_create_team_table.php

Міграція створення таблиці команд.

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateOrderTable extends Migration
{
   public function up()
   {
      Schema::create('order', function (Blueprint $table) {
         $table->id();
         $table->string('title');
         $table->text('description');
         $table->string('file', 100)->nullable();
         $table->integer('price')->nullable();
         $table->string('timing', 45)->nullable();
```

```
            $table->foreignId('account_id');
            $table->foreignId('specialization_id');
            $table->boolean('status')->default(0);
            $table->timestamps();
        });
        Schema::table('order', function (Blueprint $table) {
            $table->index('account_id');
            $table->foreign('account_id')->references('id')->on('account')->onDelete('cascade');
        });
        Schema::table('order', function (Blueprint $table) {
            $table->index('specialization_id');
            $table->foreign('specialization_id')->references('id')->on('specialization');
        });
    }
    public function down()
    {Schema::dropIfExists('order');}}
```

Account.php

Модель аккаунта.

```
namespace App\Models;
use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Laravel\Passport\HasApiTokens;
class Account extends Authenticatable
{
    use Notifiable, HasApiTokens;

    protected $table = 'account';
    protected $fillable = [
        'email',
        'password',
        'phone',
        'description',
        'photo',
        'account_status_id',
        'account_role_id',
        'country_id'
    ];
    protected $hidden = [
```

```
    'password'
  ];

  function country() {
    return $this->belongsTo('App\Models\Country', 'country_id');
  }
  function accountStatus() {
    return $this->belongsTo('App\Models\AccountStatus', 'account_status_id');
  }
  function accountRole() {
    return $this->belongsTo('App\Models\AccountRole', 'account_role_id');
  }
  function person() {
    return $this->hasOne('App\Models\Person', 'account_id');
  }
  function socials() {
    return $this->HasMany('App\Models\AccountSocial', 'account_id');
  }
  function specializations() {
    return $this->HasMany('App\Models\AccountSpecialization', 'account_id');
  }
  function skills() {
    return $this->HasMany('App\Models\AccountSkills', 'account_id');
  }
  function team() {
    return $this->hasOne('App\Models\Team', 'account_id');
  }
  function portfolio() {
    return $this->HasMany('App\Models\Portfolio', 'account_id');
  }
  function recall() {
    return $this->HasMany('App\Models\Recall', 'account_id');
  }}
```

Person.php

Модель замовника/виконавця.

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Person extends Model
```

```
{ protected $table = 'person';
    public $timestamps = false;
    protected $fillable = [
        'name',
        'surname',
        'patronymic',
        'date_bth',
        'team_id',
        'sex_id',
        'account_id'
    ];
    function team() {
        return $this->belongsTo('App\Models\Team', 'team_id');
    }
    function account() {
        return $this->belongsTo('App\Models\Account', 'account_id');
    }
    function specializations() {
        return $this->HasMany('App\Models\AccountSpecialization', 'account_id');
    }
    function skills() {
        return $this->hasMany('App\Models\AccountSkills', 'account_id');
    }}
```

Team.php

Модель команди.

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Team extends Model
{ protected $table = 'team';
    protected $fillable = [
        'title',
        'account_id'
    ];
    function person() {
        return $this->hasMany('App\Models\Person', 'team_id');
    }}
```

Order.php

Модель проектів.

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Order extends Model
{
    protected $table = 'order';
    protected $fillable = [
        'title',
        'description',
        'file',
        'price',
        'timing',
        'account_id',
        'specialization_id',
        'status'
    ];
    function requests() {
        return $this->hasMany('App\Models\Requests', 'order_id');
    }
    function account() {
        return $this->belongsTo('App\Models\Account', 'account_id');
    }
    function specialization() {
        return $this->belongsTo('App\Models\Specialization', 'specialization_id');
    }}
```

AuthController.php

Контроллер авторизації.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Carbon\Carbon;
use App\Models\Account;
use App\Models\Person;
use App\Models\Team;
class AuthController extends Controller
{
    // register
```

```
function register(Request $request) {
    $request->validate([
        'email' => 'required|string|email|unique:account',
        'password' => 'required|string'
    ]);
    $account = new Account();
    $data = $request->all();
    $data["password"] = Hash::make($request->password);
    $responseAccount = $account->create($data);
    if($request->account_role_id == 1) {
        if($request->performerType == 'person') {
            $data = $request->person;
            $data['account_id'] = $responseAccount['id'];
            $person = new Person($data);
            $person->save();
        } else {
            $data = $request->team;
            $data['account_id'] = $responseAccount['id'];
            $team = new Team($data);
            $team->save();
        }
    } else {
        $data = $request->person;
        $data['account_id'] = $responseAccount['id'];
        $person = new Person($data);
        $person->save();
    }
    $credentials = request(['email', 'password']);
    if(!Auth::attempt($credentials)) {
        return response()->json(['message' => 'Unauthorized'], 401);
    }
    $user = Account::with('team', 'person')->find($request->user()['id']);
    $tokenResult = $user->createToken('Personal Access Token');
    $token = $tokenResult->token;
    $token->save();
    return response()->json([
        'access_token' => 'Bearer '.$tokenResult->accessToken,
        'user' => $user
    ]);}
// login
```

```
    function login(Request $request) {
        if(Auth::attempt(['email' => $request->email, 'password' => $request->password, 'account_role_id'
=> 1]) || Auth::attempt(['email' => $request->email, 'password' => $request->password, 'account_role_id' => 2])) {
            $user = Account::with('team', 'person')->find($request->user()['id']);
            $tokenResult = $user->createToken('Personal Access Token');
            $token = $tokenResult->token;
            if($request->remember_me) {
                $token->expires_at = Carbon::now()->addWeeks(1);
            }
            $token->save();
            return response()->json([
                'access_token' => 'Bearer '.$tokenResult->accessToken,
                'user' => $user
            ]);
        } else {
            return response()->json(['message' => 'Unauthorized'], 401);
        }}
    // loginAdmin
    function loginAdmin(Request $request) {
        if(Auth::attempt(['email' => $request->email, 'password' => $request->password, 'account_role_id'
=> 3])) {
            $user = $request->user();
            $tokenResult = $user->createToken('Personal Access Token');
            $token = $tokenResult->token;
            $token->save();
            return response()->json([
                'access_token' => 'Bearer '.$tokenResult->accessToken
            ]);
        } else {
            return response()->json(['message' => 'Unauthorized'], 401);
        }
    }
    // logout
    function logout(Request $request) {
        $request->user()->token()->revoke();
        return response()->json([
            'message' => 'Successfully logged out'
         ]);}}
```

AccountController.php

Контроллер користувачів.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Carbon\Carbon;
use PDF;
use App\Mail\MemberEmail;
use Illuminate\Support\Facades\Mail;
use App\Models\Account;
use App\Models\Person;
use App\Models\Team;
use App\Models\Country;
use App\Models\Specialization;
use App\Models\Skills;
use App\Models\Portfolio;
use App\Models\Socials;
use App\Models\AccountSpecialization;
use App\Models\AccountSkills;
use App\Models\Notifications;
class AccountController extends Controller
{
   protected $fileStorage = "userfiles/";
   // search
   function search(Request $request) {
      $result = [];
      $accounts = Account::with('person', 'team', 'specializations.specialization', 'skills.skill')
         ->where([['account_role_id', '=', 1], ['account_status_id', '!=', 3]])
         ->whereHas('person', function($q) use ($request) {
            if($request->country_id == "") {
               $q->where('name', 'like', '%'.$request->name.'%')
               ->orWhere('surname', 'like', '%'.$request->name.'%');
            } else {
               $q->where([['name', 'like', '%'.$request->name.'%'], ['country_id', $request->country_id]])
               ->orWhere([['surname', 'like', '%'.$request->name.'%'], ['country_id', $request->country_id]]);
            }})
```

```
                ->orWhereHas('team', function($q) use ($request) {
                    if($request->country_id == "") {
                        $q->where('title', 'like', '%'.$request->name.'%');
                    } else {
                        $q->where([['title', 'like', '%'.$request->name.'%'], ['country_id', '=', $request->country_id]]);
                    }
                })
                ->get();
            foreach ($accounts as $key => $account) {
                $account->date = Carbon::parse($account->created_at)->format('d.m.Y H:i');
                foreach ($account->specializations as $key => $accountSpecialization) {
                    foreach ($request->specializations as $key => $specialization) {
                        if($accountSpecialization['specialization']['title'] == $specialization['title']) {
                            array_push($result, $account);
                        }}}}
            if(count($request->specializations) == 0) {
                $result = $accounts;
            }
            return response()->json($result);
        }
        // personToTeam
        function personToTeam(Request $request, $person_id) {
            $model = Person::with('account')->find($person_id);
            $notification = new Notifications();
            $textMessage = 'Ви стали учасником команди <a href="' . $_SERVER['SERVER_NAME'] .
'/artist/' . $request->team['account_id'] . '">' . $request->team['title'] . "</a>";
            $email = $model['account']['email'];
            $notification->create([
                'account_id' => $model['account_id'],
                'text' => 'Ви стали учасником команди <a href="/artist/' . $request->team['account_id'] . '">' .
$request->team['title'] . "</a>"
            ]);
            Mail::send([], [], function($message) use ($textMessage, $email) {
                $message->to($email)
                ->subject('Ви стали учасником команди')
                ->from('freelance@gmail.com', 'Freelance')
                ->setBody($textMessage, 'text/html');
            });
            $model->update([
                "team_id" => $request->team['id']
```

```
            ]);
        }
        // users
        function getUsers() {
            $data = Account::with('person', 'team', 'country', 'accountStatus', 'accountRole')->where('account_role_id', '!=', 3)->get();
            return response()->json($data);}
        // getArtists
        function getArtists($tag = '') {
            $data = [];
            if($tag) {
                $data['artists'] = Account::with('person', 'team', 'specializations', 'skills')
                ->where([['account_role_id', 1], ['account_status_id', '!=', 3]])
                ->whereHas('specializations', function($query) use ($tag) {
                    $query->whereHas('specialization', function($q) use ($tag) {
                        $q->where('title', 'like', '%'.$tag.'%');
                    });
                })
                ->get();
            } else {
                $data['artists'] = Account::with('person', 'team', 'specializations', 'skills')
                ->where([['account_role_id', 1], ['account_status_id', '!=', 3]])
                ->get();
            }
            foreach ($data['artists'] as $key => $value) {
                foreach ($value->specializations as $k => $v) {
                    $v['specialization'] = $v->specialization;
                }
                foreach ($value->skills as $k => $v) {
                    $v['skill'] = $v->skill;
                }
            }
            $data['specializations'] = Specialization::get();
            foreach ($data['specializations'] as $key => $value) {
                $value['selected'] = false;
                $value['countArtist'] = AccountSpecialization::where('specialization_id', $value->id)->count();
            }
            $data['countries'] = Country::get();
            return response()->json($data);
        }
```

```
// profile
function profile() {
    $data = [];
    $data['account'] = Account::with(
        "person",
        "team",
        "country",
        "accountStatus",
        "socials",
        "specializations",
        "skills",
        "portfolio",
        "recall"
    )->find(Auth::id());
    $createdAt = Carbon::parse($data['account']->created_at);
    $data['account']->date = $createdAt->format('d.m.Y');
    if($data['account']->team) {
        $data['account']->team->person;
    }
    foreach ($data['account']->specializations as $key => $value) {
        $value['specialization'] = $value->specialization;
    }
    foreach ($data['account']->skills as $key => $value) {
        $value['skill'] = $value->skill;
    }
    foreach ($data['account']->socials as $key => $value) {
        $value['social'] = $value->social;
    }
    foreach ($data['account']->recall as $key => $value) {
        $value['account'] = $value->account;
        $value['order'] = $value->order;
    }
    $data['specializations'] = Specialization::get();
    $data['skills'] = Skills::get();
    $data['socials'] = Socials::get();
    $data['countries'] = Country::get();
    return response()->json($data);}
// getUser
function getUser($id) {
    if(Account::where('id', $id)->doesntExist()) {
```

```
            return response()->json([
                "message" => "user does not exist"
            ]);
        }
        $data = Account::with(
            "person.team",
            "team",
            "country",
            "accountStatus",
            "socials.social",
            "specializations.specialization",
            "skills.skill",
            "portfolio",
            "recall.account",
            "recall.account.person",
            "recall.account.team",
        )->find($id);
        $createdAt = Carbon::parse($data->created_at);
        $data->date = $createdAt->format('d.m.Y');
        return response()->json($data);}
    // updateUser
    function updateUser(Request $request) {
        $id = Auth::id();
        $data = $request->all();
        if(isset($data['newPassword'])) {
            $user = Auth::user();
            if (Hash::check($data['oldPassword'], $user->password)) {
                $data["password"] = Hash::make($request->newPassword);
            } else {
                return response('error', 401);
            }
        }
        Account::find($id)->update($data);
        if(isset($request->person)) {
            $data = $request->person;
            Person::where('account_id', $id)->update($data);
        } else {
            Team::where('account_id', $id)->update(['title' => $request->team['title']]);
        }
        if(isset($request->specializations)) {
```

```
            AccountSpecialization::where('account_id', $id)->delete();
            foreach ($request->specializations as $key => $value) {
                $model = new AccountSpecialization();
                $model->create([
                    "account_id" => $id,
                    "specialization_id" => $value['specialization_id']
                ]);
            }
        }
        if(isset($request->skills)) {
            AccountSkills::where('account_id', $id)->delete();
            foreach ($request->skills as $key => $value) {
                $model = new AccountSkills();
                $model->create([
                    "account_id" => $id,
                    "skill_id" => $value['skill_id']
                ]);
            }
        }
        return response()->json(Account::find($id));
    }
    // updateUserId
    function updateUserId(Request $request, $id) {
        $data = $request->all();
        Account::find($id)->update($data);
        return response('ok', 200);
    }
    // others
    // img Account
    function img(Request $request) {
        if(isset($request['photo'])) {
            $arr = [];
            if($request['photo']) {
                $file = uniqid().'_photo_min.png';
                $uploadfile = $this->fileStorage . $request['id'] . '/' . $file;
                $img = str_replace('', '', $request['photo']);
                $img = str_replace(' ', '+', $img);
                $fileData = base64_decode($img);
                file_put_contents(public_path().'/'.$uploadfile, $fileData);
```

```
                $this->compressImage(public_path().'/'.$this->fileStorage . $request['id'] . '/'. $file, $this->fileStorage . $request['id'] . '/' . $file, 80);
                $arr['status'] = 'success';
                $arr['path_mini'] = 'http://'.$_SERVER['HTTP_HOST'].'/'.$uploadfile;
                $arr['file_mini'] = $file;
            }
        }
        else {
            if(!file_exists($this->fileStorage.$request['id'])) {
                mkdir($this->fileStorage.$request['id']);
            }
            $uploadfile = $this->fileStorage. $request['id'] . '/' . uniqid().'_photo_original.png';
            $arr = array();
            if (move_uploaded_file($_FILES['file']['tmp_name'], public_path().'/'.$uploadfile)) {
                $arr['status'] = 'success';
                $arr['path_max'] = 'http://'.$_SERVER['HTTP_HOST'].'/'.$uploadfile;
                $arr['file_max'] = $_FILES['file']['name'];
            } else {
              $arr['status'] = 'fail';}}
        header('Content-type: application/json');
        return response()->json($arr);
    }
    // compressImage
    function compressImage($source_url, $destination_url, $quality) {
        $info = getimagesize($source_url);
        if ($info['mime'] == 'image/jpeg') $image = imagecreatefromjpeg($source_url);
        elseif ($info['mime'] == 'image/gif') $image = imagecreatefromgif($source_url);
        elseif ($info['mime'] == 'image/png') $image = imagecreatefrompng($source_url);
        imagejpeg($image, $destination_url, $quality);
    }
    // country
    function country() {
        $data = Country::get();
        return response()->json($data);
    }
    // PDF resume
    function resume($id) {
        $data = Account::with(
            "person",
            "country",
```

```
            "socials",
            "specializations.specialization",
            "skills.skill",
            "portfolio"
        )->find($id);
        $pdf = PDF::loadView('pdf.resume', ['data' => $data]);
        return $pdf->stream('Resume.pdf');
    }
    // Notifications
    function getNotifications() {
        $id = Auth::id();
        $data = Notifications::where('account_id', $id)->orderByDesc('created_at')->get();
        return response()->json($data);
    }
    function putNotifications($id) {
        Notifications::find($id)->update([
            'status' => 1
        ]);
        return response('ok', 200);}}
```

OrderController.php

Контроллер проектів.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Carbon\Carbon;
use App\Mail\MemberEmail;
use Illuminate\Support\Facades\Mail;
use App\Models\Order;
use App\Models\AccountSpecialization;
use App\Models\Notifications;
class OrderController extends Controller
{
    protected $fileStorage = "/userfiles/";
    // sorted
    function sorted(Request $request) {
        $orders = Order::with('specialization')
        ->where('status', 0)
```

```
    ->orderBy($request->column, $request->type)
    ->get();
    foreach ($orders as $key => $order) {
        $order->date = Carbon::parse($order->created_at)->format('d.m.Y H:i');
    }
    return response()->json($orders);
}
// filter
function filter(Request $request) {
    $result = [];
    $orders = Order::with('specialization')
        ->with('specialization')
        ->where('status', 0)
        ->where('title', 'like', '%'.$request->title.'%')
        ->get();
    foreach ($orders as $key => $order) {
        $order->date = Carbon::parse($order->created_at)->format('d.m.Y H:i');
        foreach ($request->specializations as $key => $specialization) {
            if($order['specialization']['title'] == $specialization['title']) {
                array_push($result, $order);
            }}}
    if(count($request->specializations) == 0) {
        $result = $orders;
    }
    return response()->json($result);}
// get
function get($tag = '') {
    $order = Order::with('specialization')
        ->where('status', 0)
        ->whereHas('specialization', function($q) use ($tag) {
            $q->where('title', 'like', '%'.$tag.'%');
        })
        ->orderBy('created_at', 'desc')
        ->get();
    foreach ($order as $key => $value) {
        $value->date = Carbon::parse($value->created_at)->format('d.m.Y H:i');
    }
    return response()->json($order);
}
// getAll
```

```
    function getAll() {
        $order = Order::with('specialization', 'account.person')->get();
        return response()->json($order);
    }
    // post
    function post(Request $request) {
        $order = new Order();
        $data = $request->all();
        $data['account_id'] = Auth::id();
        if(isset($request['file']) && $request['file'] != "null") {
            $puth = $this->fileStorage.Auth::id();
            $name =  $puth."/".uniqid().'.'.$request['file']->getClientOriginalExtension();
            $request['file']->move(public_path().$puth, $name);
            $data['file'] = $name;
        }
        $response = $order->create($data);
        $accounts = AccountSpecialization::with('account')->where('specialization_id', $request->specialization_id)->get();
        foreach ($accounts as $key => $value) {
            $notification = new Notifications();
            $textMessage = 'Опубліковано новий проект за Вашою спеціалізацією <a href="' . $_SERVER['SERVER_NAME'] . '/project/' . $response['id'] . '">' . $request->title . '</a>';
            $email = $value['account']['email'];
            $notification->create([
                'account_id' => $value['account_id'],
                'text' => 'Опубліковано новий проект за Вашою спеціалізацією <a href="/project/' . $response['id'] . '">' . $request->title . '</a>'
            ]);
            Mail::send([], [], function($message) use ($textMessage, $email) {
                $message->to($email)
                ->subject('Опубліковано новий проект за Вашою спеціалізацією')
                ->from('freelance@gmail.com', 'Freelance')
                ->setBody($textMessage, 'text/html');
            });}
        return response('ok', 200);
    }
    // update
    function update(Request $request, $id) {
        $data = $request->all();
        if(isset($request['file']) && $request['file'] != "null") {
```

```
            $puth = $this->fileStorage.Auth::id();
            $name =  $puth."/".uniqid().'.'.$request['file']->getClientOriginalExtension();
            $request['file']->move(public_path().$puth, $name);
            $data['file'] = $name;
        }
        Order::find($id)->update($data);
        return response('ok', 200);
    }
    // getOrderId
    function getOrder($id) {
        $data = Order::with(
            'account.person',
            'account.team',
            'account.country',
            'account.accountStatus',
            'specialization.specialization'
        )->find($id);
        $data->similar = Order::where([['specialization_id', $data['specialization_id']], ['id', '!=', $data['id']]])->limit(5)->get();
        $data->date = Carbon::parse($data->created_at)->format('d.m.Y H:i');
        return response()->json($data);
    }
    // deleteOrder
    function deleteOrder($id) {
        $data = Order::find($id)->delete();
        return response('ok', 200); }}
```

Home.vue

Компонент головної сторінки.

```
<template>
    <b-container>
        <b-row>
            <b-col md="8" lg="8" xl="8" xs="12">
                <div class="mr-5 contentLeft">
                    <b-row>
                        <b-col class="title-1 mb-3" md="5" lg="5" xl="5" xs="12">
                            Всі проекти
                        </b-col>
                        <b-col class="sorted text-right" md="7" lg="7" xl="7" xs="12">
```

```
                    Сортувати за: <span @click="sorted('created_at')">датою {{ sort.created_at ? "&and;" : "&or;" }} </span> <span @click="sorted('price')">ціною {{ sort.price ? "&and;" : "&or;" }}</span>
                  </b-col>
                </b-row>
                <div v-show="userRole == 2">
                  <div class="add" v-if="!form">
                    <span>Створити власний проект...</span>
                    <div class="plus-button" @click="form = true">+</div>
                  </div>

                  <div class="add-form" v-else>
                    <div class="header-1 mb-4">
                      Створення власного проекту
                    </div>
                    <b-row class="mb-3">
                      <b-col cols="2">
                        <label>Назва:</label>
                      </b-col>
                      <b-col cols="10">
                        <b-form-input
                          v-model="order.title"
                          name="title"
                          v-validate="{ required: true}"
                          :class="errors.has('title') ? 'input-has-error' : ""
                        ></b-form-input>
                      </b-col>
                    </b-row>
                    <b-row class="mb-3">
                      <b-col cols="2">
                        <label>Спеціалізація:</label>
                      </b-col>
                      <b-col cols="10">
                        <b-form-select
                          v-model="order.specialization_id"
                          :options="specializations"
                          value-field="id"
                          text-field="title"
                          name="specialization"
                          v-validate="{ required: true}"
                          :class="errors.has('specialization') ? 'input-has-error' : ""
```

```
        ></b-form-select>
    </b-col>
</b-row>
<b-row class="mb-3">
    <b-col cols="2">
        <label>Ціна:</label>
    </b-col>
    <b-col cols="10">
        <b-form-input
            v-model="order.price"
            name="price"
            v-validate="{ required: true}"
            :class="errors.has('price') ? 'input-has-error' : ""
        ></b-form-input> грн.
    </b-col>
</b-row>
<b-row class="mb-3">
    <b-col cols="2">
        <label>Час:</label>
    </b-col>
    <b-col cols="10">
        <b-form-input
            v-model="order.timing"
            name="timing"
            v-validate="{ required: true}"
            :class="errors.has('timing') ? 'input-has-error' : ""
        ></b-form-input> днів
    </b-col>
</b-row>
<b-row class="mb-3">
    <b-col cols="2">
        <label>Додаткова інформація:</label>
    </b-col>
    <b-col cols="10">
        <b-form-textarea
            v-model="order.description"
            rows="5"
            name="description"
            v-validate="{ required: true}"
            :class="errors.has('description') ? 'input-has-error' : ""
```

```
            ></b-form-textarea>
          </b-col>
        </b-row>
        <b-row class="mb-3">
          <b-col cols="2">
            <label>Файл:</label>
          </b-col>
          <b-col cols="10">
            <b-form-file
              ref="file"
              placeholder="Оберіть файл"
              v-model="order.file"
            ></b-form-file>
          </b-col>
        </b-row>
        <div class="text-right">
          <ActiveButton
            @click.native="submit"
            :loading="loading"
            value="Опублікувати"
          ></ActiveButton>

          <button class="button-default px-5 mt-3" type="button" @click="form = false">
            Видалити
          </button>
        </div>
      </div>
      <hr class="my-4">
    </div>

    <CardProject
      v-for="(item, index) in paginatedList"
      :key="index"
      :item="item"
      @remove="removeFromList"
    ></CardProject>

    <div class="d-flex justify-content-center" v-if="preloader">
      <b-spinner style="width: 2rem; height: 2rem;" label="Large Spinner"></b-spinner>
    </div>
```

```
                        <div class="d-flex justify-content-center mt-4">
                            <b-pagination v-model="currentPage" pills :total-rows="data.length" :per-page="prePage"></b-pagination>
                        </div>
                    </div>
                </b-col>
                <b-col md="4" lg="4" xl="4" xs="12" class="px-1 text-center">
                    <FilterComponent
                        :specializations="specializations"
                        @update="filteredOrders($event)"
                    ></FilterComponent>
                    <hr class="mt-5">
                    <BestPerformers></BestPerformers>
                </b-col>
            </b-row>
        </b-container>
    </template>
    <script>
        import CardProject from '../../components/site/CardProject';
        import FilterComponent from '../../components/site/FilterComponent';
        import BestPerformers from '../../components/site/BestPerformers';
        import ActiveButton from '../../components/site/buttons/ActiveButton';
        export default {
            components: {
                CardProject,
                FilterComponent,
                BestPerformers,
                ActiveButton
            },
            data() {
                return {
                    currentPage: 1,
                    prePage: 6,
                    preloader: true,
                    loading: false,
                    form: false,
                    data: [],
                    specializations: [],
                    sort: {
```

```
            price: false,
            created_at: false,
        },
        order: {
            title: "",
            description: "",
            price: "",
            timing: "",
            specialization_id: null,
            file: null},};},
created() {
    this.fetchData();
    this.getSpecializations();
},
computed: {
    userRole() {
        return this.$store.getters.authUser.account_role_id
    },
    paginatedList() {
        let start = (this.currentPage - 1) * this.prePage;
        let end = start + this.prePage;
        return this.data.slice(start, end);
    }
},
methods: {
    fetchData() {
        axios.get(this.$route.params.tag ? '/api/orders/tag/'+this.$route.params.tag : '/api/orders')
        .then((response) => {
            this.preloader = false;
            this.data = response.data;
        })
        .catch((err) => {
            this.preloader = false;
        })
    },
    getSpecializations() {
        axios.get('/api/specialization')
        .then((response) => {
            this.specializations = response.data;
        })
```

```
},
submit() {
    this.$validator.validateAll().then((result) => {
        if (!result) {
            return;
        } else {
            this.loading = true;
            let formData = new FormData();
            formData.append('file', this.order.file);
            formData.append('title', this.order.title);
            formData.append('description', this.order.description);
            formData.append('price', this.order.price);
            formData.append('timing', this.order.timing);
            formData.append('specialization_id', this.order.specialization_id);
            axios.post('/api/orders', formData, {
                headers: {
                    'Content-Type': 'multipart/form-data'
                }
            })
            .then((response)=> {
                this.fetchData();
                this.form = false;
                this.loading = false;
                this.$validator.reset();
                this.order.title = "";
                this.order.description = "";
                this.order.price = "";
                this.order.timing = "";
                this.order.specialization_id = null;
            })
            .catch((err)=> {
                this.loading = false;
            })
        }
    })
},
removeFromList(item) {
    const index = this.data.indexOf(item)
    this.data.splice(index, 1)
},
```

```
            sorted(column) {
                this.sort[column] = !this.sort[column];
                this.preloader = true;
                this.data = [];
                axios.post('/api/orders/sorted', {
                    type: this.sort[column] ? 'asc' : 'desc',
                    column
                })
                .then((response)=> {
                    this.data = response.data;
                    this.preloader = false;
                })
                .catch((err)=> {
                    alert("error")
                    this.preloader = false;
                })
            },
            filteredOrders(data) {
                this.data = data;
            }
        },
        watch: {
            $route(to, from) {
                this.fetchData();
            }
        }
    }
</script>
<style lang="css" scoped>
    .add {
        background: #ffffff;
        box-shadow: 0 0 10px rgba(0,0,0,0.2);
        border-top-right-radius: 50px;
        border-bottom-right-radius: 50px;
    }
    .add span {
        padding: 22px 0 20px 50px;
        display: inline-block;
        width: 70%;
    }
```

```
.button-default {
    color: #000 !important;
}
.add .plus-button {
    border-radius: 50%;
    background: #0076DE;
    float: right;
    color: #ffffff;
    padding: 15px;
    font-size: 60px;
    text-align: center;
    line-height: 35px;
    cursor: pointer;
}
.add-form {
    background: #ffffff;
    box-shadow: 0 0 10px rgba(0,0,0,0.2);
    padding: 30px 60px;
}
.add-form input[type=text], .add-form select {
    display: inline-block;
    width: 80%;
}
.add-form textarea {
    width: 100%;
}
.sorted {
    padding-top: 15px;
}
.sorted span {
    margin-left: 30px;
    word-spacing: 5px;
    cursor: pointer;
}
@media screen and (max-width: 600px) {
    .contentLeft {
        margin: 0 !important;
    }
    .sorted {
        margin-bottom: 10px;
```

```
    }
  }
</style>
```

ArtistView.vue

Компонент сторінки виконавця.

```
<template>
  <div>
    <b-container>
      <LinkBack link="/feed" text="Інші заяви"></LinkBack>

      <PersonInfo colsLeft="4" :data="data" @addToTeam="fetchData()"></PersonInfo>

      <AdditionalInfo
        :description="data.description"
        :skills="data.skills"
        :data="data"
      ></AdditionalInfo>

      <div v-if="data.team && data.team.person.length > 0">
        <div class="title-1 mt-3">
          Наша команда
        </div>
        <b-row class="mt-4">
          <b-col v-for="artist in data.team.person" :key="artist.id" md="3" lg="3" xl="3" cols="6"
class="mb-4">
            <CardArtistTeam :item="artist"></CardArtistTeam>
          </b-col>
        </b-row>
      </div>

      <Portfolio class="mt-5" :data="data.portfolio"></Portfolio>

      <Cooperation class="mt-5" :data="data.recall" v-if="data.recall.length > 0"></Cooperation>
    </b-container>
  </div>
</template>
<script>
import LinkBack from '../../components/site/LinkBack';
```

```
import PersonInfo from '../../components/site/PersonInfo';
import Portfolio from '../../components/site/Portfolio';
import Cooperation from '../../components/site/Cooperation';
import AdditionalInfo from '../../components/site/AdditionalInfo';
import CardArtistTeam from '../../components/site/CardArtistTeam';

export default {
  components: {
    LinkBack,
    PersonInfo,
    Portfolio,
    Cooperation,
    AdditionalInfo,
    CardArtistTeam
  },
  data() {
    return {
      data: {
        id: null,
        email: "",
        phone: "",
        description: "",
        photo: "",
        account_status: {
          title: ""
        },
        country: {
          title: ""
        },
        person: {
          name: "",
          surname: "",
          date_bth: ""
        },
        team: {
          title: "",
          person: []
        },
        portfolio: [],
        recall: []
```

```
            }
        }
    },
    created() {
        this.fetchData();
    },
    methods: {
        fetchData() {
            axios.get('/api/user/'+this.$route.params.id)
            .then((response) => {
                this.data = response.data;
            })
            .catch((err) => {
                alert("error")
            })
        }
    },
    watch: {
        $route(to, from) {
            this.fetchData();
        }
    }
}
</script>
<style lang="css" scoped>

</style>
```

ProjectView.vue

Компонент сторінки проекту

```
<template>
    <b-container>
        <b-modal v-model="editModal" hide-footer>
            <b-row class="mb-3">
                <b-col cols="4">Назва:</b-col>
                <b-col>
                    <b-form-input v-model="editData.title"></b-form-input>
                </b-col>
            </b-row>
            <b-row class="mb-3">
```

```
    <b-col cols="4">Спеціалізація:</b-col>
    <b-col>
      <b-form-select
        v-model="editData.specialization_id"
        :options="specializations"
        value-field="id"
        text-field="title"
      ></b-form-select>
    </b-col>
  </b-row>
  <b-row class="mb-3">
    <b-col cols="4">Ціна:</b-col>
    <b-col>
      <b-form-input v-model="editData.price"></b-form-input>
    </b-col>
  </b-row>
  <b-row class="mb-3">
    <b-col cols="4">Час:</b-col>
    <b-col>
      <b-form-input v-model="editData.timing"></b-form-input>
    </b-col>
  </b-row>
  <b-row class="mb-3">
    <b-col cols="4">Додаткова інформація:</b-col>
    <b-col>
      <b-form-textarea
        v-model="editData.description"
        rows="5"
      ></b-form-textarea>
    </b-col>
  </b-row>
  <b-row class="mb-3">
    <b-col cols="4">Файл:</b-col>
    <b-col>
      <b-form-file
        ref="file"
        placeholder="Оберіть файл"
        v-model="file"
      ></b-form-file>
    </b-col>
```

```
            </b-row>
            <b-button @click="editProject">Зберегти</b-button>
        </b-modal>

        <LinkBack link="/feed" text="Пошук"></LinkBack>
        <b-row>
            <b-col md="8" lg="8" xl="8" xs="12" class="project-info">
                <b-row>
                    <b-col cols="7">
                        <h1 class="title-1">{{ data.title }}</h1>
                        <div class="tags">
                            <router-link :to="'/projects/'+data.specialization.title">#{{ data.specialization.title }}</router-link>
                        </div>
                    </b-col>
                    <b-col cols="5" class="text-right">
                        Ціна: <span class="price">{{ data.price }} грн.</span>
                    </b-col>
                </b-row>
                <p>{{ data.description }}</p>
                <p>Час виконання: {{ data.timing }} днів.</p>
                <div v-if="data.file != 'null'">
                    <hr style="background: #fff">
                    <p class="file">
                        <a :href="data.file">
                            <b-icon icon="file-earmark-plus" font-scale="2"></b-icon>
                            Прикріплений файл
                        </a>
                    </p>
                </div>
                <b-row>
                    <b-col>
                        Дата створення: {{ data.date }}
                    </b-col>
                    <b-col class="text-right">
                        <input
                            type="button"
                            value="Видвинути пропозицію"
                            @click="createOffer = true"
```

```
                        v-show="userHasRequest && userRole == 1 && !data.requests.find(item =>
item.status)"
                    >
                    <b-icon
                        icon="pencil"
                        font-scale="3"
                        class="edit-buttom cursor"
                        v-show="userId == data.account_id"
                        v-b-modal.edit-modal
                        @click="openModal"
                    ></b-icon>
                </b-col>
            </b-row>
        </b-col>
        <b-col md="4" lg="4" xl="4" xs="12" class="person-info">
            <div class="photo">
                <img :src="data.account.photo">
            </div>
            <div class="name">
                {{ data.account.name }}
            </div>

            <!-- Успішних проектів: 0 -->

            <b-row>
                <b-col cols="4">Країна:</b-col>
                <b-col>{{ data.account.country.title }}</b-col>
            </b-row>
            <b-row>
                <b-col cols="4">E-mail:</b-col>
                <b-col style="text-overflow: ellipsis;overflow: hidden" :title="data.account.email">{{
data.account.email }}</b-col>
            </b-row>
            <b-row>
                <b-col cols="4">Телефон:</b-col>
                <b-col>{{ data.account.phone }}</b-col>
            </b-row>
            <b-row>
                <b-col cols="4">Статус:</b-col>
                <b-col>{{ data.account.account_status.title }}</b-col>
```

```
      </b-row>
    </b-col>
  </b-row>
  <b-row>
    <b-col md="7" lg="7" xl="7" xs="12">
      <div v-show="createOffer">
        <div class="header-1 mt-5">
          Створення власної пропозиції
        </div>
        <div class="create-offer">
          <b-row class="mb-3">
            <b-col cols="2">
              <label for="">Ціна:</label>
            </b-col>
            <b-col cols="10">
              <b-form-input
                v-model="newRequest.price"
                name="price"
                v-validate="{required: true}"
                :class="errors.has('price') ? 'input-has-error' : ""
              ></b-form-input> грн.
            </b-col>
          </b-row>
          <b-row class="mb-3">
            <b-col cols="2">
              <label for="">Час:</label>
            </b-col>
            <b-col cols="10">
              <b-form-input
                v-model="newRequest.timing"
                name="timing"
                v-validate="{required: true}"
                :class="errors.has('timing') ? 'input-has-error' : ""
              ></b-form-input> днів
            </b-col>
          </b-row>
          <b-row class="mb-3">
            <b-col cols="2">
              <label for="">Додаткова інформація:</label>
            </b-col>
```

```
            <b-col cols="10">
              <b-form-textarea
                v-model="newRequest.description"
                rows="2"
                name="description"
                v-validate="{required: true}"
                :class="errors.has('description') ? 'input-has-error' : ""
              ></b-form-textarea>
            </b-col>
          </b-row>
          <div class="text-right">
            <ActiveButton
              @click.native="sendRequest"
              :loading="loading"
              value="Видвинути пропозицію"
            ></ActiveButton>
            <input
              type="button"
              class="no-active ml-3"
              value="Видалити"
              @click="createOffer = false"
            >
          </div>
        </div>
      </div>
      <div class="header-1 mt-5">
        Пропозиції виконавців
      </div>
      <div v-show="data.requests.find(item => item.status)" class="mt-3">
        Подачу пропозицій завершено
      </div>
      <CardOffer
        v-for="(item, index) in data.requests"
        :key="index"
        :item="item"
        :orderId="data.id"
        @remove="removeRequest"
        @selectOffer="fetchData()"
      ></CardOffer>
      <div v-show="data.requests.length == 0" class="mt-3">
```

```
                Пропозиції виконавців відсутні
              </div>
            </b-col>
            <b-col md="5" lg="5" xl="5" xs="12" class="pl-5">
              <div class="header-1 mt-5">
                Схожі заяви
              </div>
              <SimilarStatements
                v-for="(item, index) in data.similar"
                :key="index"
                :item="item"
              ></SimilarStatements>
              <div v-show="data.similar.length == 0" class="mt-3">
                Схожих проектів не знайдено
              </div>
            </b-col>
          </b-row>
        </b-container>
</template>
<script>
import CardOffer from '../../components/site/CardOffer';
import SimilarStatements from '../../components/site/SimilarStatements';
import LinkBack from '../../components/site/LinkBack';
import ActiveButton from '../../components/site/buttons/ActiveButton';
export default {
    components: {
        CardOffer,
        SimilarStatements,
        LinkBack,
        ActiveButton
    },
    data() {
        return {
            createOffer: false,
            loading: false,
            editData: {},
            editModal: false,
            specializations: [],
            file: null,
            data: {
```

```
                title: "",
                price: "",
                file: null,
                date: "",
                account: {
                    photo: "",
                    name: "",
                    email: "",
                    phone: "",
                    country: {
                        title: ""
                    },
                    account_status: {
                        title: ""
                    }
                },
                specialization: {
                    title: ""
                },
                requests: [],
                similar: []
            },
            newRequest: {
                price: "",
                timing: "",
                description: ""
            }
        }
    },
    created() {
        this.fetchData();
        this.getSpecializations();
    },
    computed: {
        userHasRequest() {
            return this.data.requests.find(item => item.account_id == this.$store.getters.authUser.id) ? false :
true
        },
        userRole() {
            return this.$store.getters.authUser.account_role_id
```

```
    },
    userId() {
        return this.$store.getters.authUser.id
    }
},
methods: {
    fetchData() {
        axios.get('/api/orders/'+this.$route.params.id)
        .then((response) => {
            this.data = response.data;
        })
        .catch((err) => {
            alert("error")
        })
    },
    sendRequest() {
        this.$validator.validateAll().then((result) => {
            if (!result) {
                return;
            } else {
                this.loading = true;
                this.newRequest.account_customer = this.data.account;
                this.newRequest.order_id = this.$route.params.id;
                this.newRequest.order_title = this.data.title;
                axios.post('/api/request', this.newRequest)
                .then((response) => {
                    this.loading = false;
                    this.createOffer = false;
                    this.$validator.reset();
                    this.fetchData();
                })
                .catch((err)=> {
                    this.loading = false;
                    alert("error")
                })
            }
        })
    },
    getSpecializations() {
        axios.get('/api/specialization')
```

```
        .then((response) => {
            this.specializations = response.data;
        })
        .catch((err) => {
            alert("error")
        })
    },
    openModal() {
        this.editData = Object.assign({}, this.data);
        this.editModal = true;
    },
    editProject() {
        let formData = new FormData();
        formData.append('file', this.file);
        formData.append('title', this.editData.title);
        formData.append('description', this.editData.description);
        formData.append('price', this.editData.price);
        formData.append('timing', this.editData.timing);
        formData.append('specialization_id', this.editData.specialization_id);
        axios.post('/api/orders/'+this.data.id, formData, {
            headers: {
                'Content-Type': 'multipart/form-data'
            }
        })
        .then(()=> {
            this.editModal = false;
            this.fetchData();
        })
        .catch((err)=> {
            alert("error")
        })
    },
    removeRequest(item) {
        const index = this.data.requests.indexOf(item)
        this.data.requests.splice(index, 1)
    },
},
watch: {
    $route(to, from) {
        this.fetchData();
```

```
        }
    }
}
</script>
<style lang="css" scoped>
    .edit-buttom {
        background: #000367;
        color: #fff;
        padding: 10px;
        border-radius: 50%;
    }
    .create-offer {
        margin-top: 20px;
        box-shadow: 0 0 10px rgba(0,0,0,0.2);
        padding: 40px 70px;
        background: #ffffff;
    }
    .create-offer input[type=text] {
        display: inline-block;
        width: 80%;
    }
    .create-offer textarea {
        width: 100%;
    }
    .person-info {
        background: #000367;
        border-radius: 5px;
        color: #fff;
        padding: 20px 30px;
    }
    .person-info img {
        height: 100%;
    }
    .person-info .photo {
        width: 200px;
        height: 200px;
        border-radius: 50%;
        margin: 0 auto;
        overflow: hidden;
    }
```

```
        .person-info .name {
            font-size: 20px;
            font-weight: bold;
            text-align: center;
            padding: 10px;
        }
        .project-info {
            background: #0076DE;
            border-radius: 5px;
            padding: 70px;
            color: #ffffff;
        }
        .project-info .tags {
            margin-top: -5px;
        }
        .project-info .tags a {
            color: #ffffff;
            margin-right: 10px;
        }
        .project-info h1 {
            color: #fff;
        }
        .project-info .price {
            font-size: 40px;
            font-weight: bold;
        }
        .project-info p {
            margin-top: 20px;
        }
        .project-info input[type=button] {
            background: #000367;
            color: #fff;
            padding: 10px 25px;
        }
        .project-info .file a {
            color: #ffffff;
            text-decoration: underline;
        }
</style>
```

Profile.vue

Компонент профілю

```
<template>
  <div>
    <b-container>
      <b-row>
        <b-col md="8" lg="8" xl="8" xs="12">
          <PersonInfo colsLeft="5" :data="data"></PersonInfo>

          <AdditionalInfo
            :description="data.description"
            :specializations="data.specializations"
            :data="data"
          ></AdditionalInfo>

          <div v-if="data.team && data.team.person.length > 0">
            <div class="title-1 mt-3">
              Наша команда
            </div>
            <b-row class="mt-4">
              <b-col v-for="artist in data.team.person" :key="artist.id" md="3" lg="3" xl="3" cols="6"
class="mb-4">
                <CardArtistTeam :item="artist"></CardArtistTeam>
              </b-col>
            </b-row>
          </div>

          <Portfolio v-show="userRole == 1" class="my-3" :data="data.portfolio"></Portfolio>
          <Cooperation class="mt-5" :data="data.recall" v-if="data.recall.length > 0"></Cooperation>
        </b-col>
        <b-col md="4" lg="4" xl="4" xs="12" class="menu">
          <Menu></Menu>
        </b-col>
      </b-row>
    </b-container>
  </div>
</template>
<script>
import crud from '../../../mixins/crud';
```

```
import PersonInfo from '../../../components/site/PersonInfo';
import Portfolio from '../../../components/site/Portfolio';
import Cooperation from '../../../components/site/Cooperation';
import CardArtistTeam from '../../../components/site/CardArtistTeam';
import Menu from '../../../components/site/Menu';
import AdditionalInfo from '../../../components/site/AdditionalInfo';
export default {
    mixins: [crud],
    components: {
        PersonInfo,
        Portfolio,
        Cooperation,
        CardArtistTeam,
        AdditionalInfo,
        Menu},
    data() {
        return {
            apiUrl: 'user/'+JSON.parse(localStorage.getItem('user')).id,
            data: {
                id: null,
                email: "",
                phone: "",
                description: "",
                photo: "",
                created_at: "",
                account_status: {
                    title: ""
                },
                country: {
                    title: ""
                },
                person: {
                    name: "",
                    surname: "",
                    date_bth: ""
                },
                team: null,
                portfolio: [],
                recall: []
            }
```

```
            }
        },
        created() {
            this.fetchData();
        },
        computed: {
            userRole() {
                return this.$store.getters.authUser.account_role_id
            }}}
    </script>
    <style lang="css" scoped>
        @media screen and (max-width: 600px) {
            .menu {
                margin-top: 20px !important;}}
</style>
```