Галицький фаховий коледж імені В’ячеслава Чорновола
відділення комп’ютерних та видавничих технологій
циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач відділенням
комп’ютерних та видавничих
технологій
Чубей О.О. /_____/
підпис
«___»_____2022 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту
освітньо-кваліфікаційного рівня «молодший спеціаліст»
зі спеціальності 122 «Комп’ютерні науки»
на тему : «Вебсервіс керування завданнями для фрілансу»

Студент групи КН-41 Занько Є.А _____
(підпис)

Керівник проекту Сиротюк Н.С. _____
(підпис)

Консультанти:

з техніко-економічного
обґрунтування Меленчук Л.І. _____
(підпис)

нормоконтролер Гавришків Н.Г. _____
(підпис)

Тернопіль – 2022

Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп`ютерних дисциплін

ЗАТВЕРДЖУЮ
Завідувач відділенням
комп’ютерних та видавничих
технологій

Чубей О.О. /_____/
підпис
«___»_____2021 р.

ЗАВДАННЯ

на дипломне проєктування

на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»

студенту _____
(прізвище, ім’я та по-батькові студента)

1. Тема проєкту _____
_____
_____
_____

затверджена наказом по коледжу від “01” “10” 2021 р., № 178-н

2. Термін здачі студентом завершеного проекту “_____” _____2022 р

3. Вихідні дані до проекту _____
_____
_____
_____

4. Перелік питань, які повинні бути розроблені в проекті: _____

а) основна частина _____
_____
_____
_____

б) техніко-економічного обґрунтування _____
_____
_____
_____
_____

5. Перелік графічного матеріалу_____
_____
_____

6. Консультанти проекту: _____

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	_____ (вчена ступень, звання П.І.Б. _____ консультанта)		

КАЛЕНДАРНИЙ ПЛАН
дипломного проектування

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми, ознайомлення з вимогами до дипломного проєктування.	29.09.21 р.	01.10.21 р.
2.	Огляд типових рішень та написання відповідного розділу ПЗ	02.12.21 р.	28.01.22 р.
3.	Дослідження технологій реалізації та написання відповідного розділу ПЗ	28.01.21 р.	17.02.22 р.
4.	Розробка функціональних вимог до проєкту та робота над структурою програмного продукту. Написаннявідповідного розділу ПЗ	17.02.22 р.	04.03.22 р.
5.	Встановлення на налаштування середовища реалізації та написання відповідного розділу ПЗ	04.03.22 р.	16.03.22 р.
6.	Проєктування програмного засобу (функціоналу, інтерфейсу, бази даних продукту) та написання відповідного розділу ПЗ	16.03.22 р.	15.04.22 р.
7.	Реалізація та налаштування програмного засобу та написання відповідного розділу ПЗ	15.04.22 р.	04.05.22 р.
8.	Доопрацювання модулів	05.05.22 р.	18.05.22 р.
9.	Тестування на налагодження програмного продукту та написання відповідного розділу ПЗ	18.05.22 р.	01.06.22 р.
10.	Опрацювання економічного розділу дипломного проєкту та оформлення спеціального розділу	01.06.22 р.	05.06.22 р.
11.	Робота над оформленням пояснювальної записки	05.06.22 р.	12.06.22 р.
12.	Попередній захист дипломного проєкту, доопрацювання	15.06.22 р.	15.06.22
13.	Підготовка до захисту дипломного проєкту	16.06.22 р.	24.06.22 р.
14.	Захист дипломного проєкту	25.06.22 р.	26.06.22 р.

7. Дата видачі завдання ”____” _____ 2021 р.

Керівник _____ / _____

Завдання прийняв до виконання_____ / _____

Реферат

Дипломний проєкт. Тема: «Вебсервіс керування завданнями для фрілансу». 60 сторінок, 45 рисунків, 1 таблиця, 9 джерел, 1 додаток.

Мета дипломного проєкту полягає в тому, щоб спроектувати та розробити сучасний повноцінний веб-додаток для керування завданнями для фрілансу, використовуючи сучасні інструменти розробки зі зручним дизайном та інтерфейсом.

Для реалізації було обрано середовище розробки WebStorm, яке спеціалізується на розробці веб-додатків. Було застосовано сучасні інструменти в веб-розробці. Для клієнтської частини було використано сучасну JavaScript бібліотеку – ReactJS, яка була створена компанією Meta. Для виконання серверної частини було обрано JavaScript платформу – Node JS з використанням побіжних бібліотек до цього інструменту.

Призначенням даної розробки є допомога початковим розробникам реалізувати свої навички та вміння, здобувши за це певну грошову нагороду, використовуючи створений зручний інтерфейс та усі функції для взаємодії з веб-сервісом.

REACTJS JS, MATERIAL UI, NODE JS, EXPRESS JS, MONGOOSE, MONGODB, JAVASCRIPT.

Abstract

Diploma project. Topic: "Web service management tasks for freelancers." 60 pages, 45 figures, 1 table, 9 sources, 1 appendix.

The main goal of the diploma project is to design and develop a modern full-fledged web application for managing freelance tasks, using modern development tools with a user-friendly design and interface.

The WebStorm development environment, which specializes in web application development, was chosen for implementation. Modern web development tools have been used. For the client part, a modern JavaScript library was used - ReactJS, which was created by Meta. To implement the server part, a JavaScript platform was chosen - Node JS using short libraries for this tool.

The purpose of this development is to help novice developers to realize their skills and abilities, earning a cash prize, using a user-friendly interface and all the features to interact with the web service.

REACTJS JS, MATERIAL UI, NODE JS, EXPRESS JS, MONGOOSE, MONGODB, JAVASCRIPT.

ЗМІСТ

ВСТУП

На сьогоднішній день будь-якому сервісу, будь то навіть «офлайн», необхідний веб-додаток. Він потрібен для збільшення кількості потенційних клієнтів та користувачів.

Веб-додаток – це повноцінний інтернет додаток, з клієнтською та серверною частиною, з базами даних в яких зберігається інформація.

Кожен веб-ресурс має мати свій інтерфейс, з яким буде взаємодіяти потенційний користувач. Для цього навіть є окрема сфера в IT – веб-дизайн. Це працівники, які створюють приємний та зручний макет або дизайн для додатку, який включає в себе необхідні шрифти, кольори, зображення та все інше що може зацікавити користувача.

Варто не забувати, що веб-сервіс не має бути повільним, тому що це може відлякувати користувачів, коли додаток витрачає багато часу на завантаження елементів. Для створення сучасних та швидких веб-додатків використовується мова JavaScript та її фреймворки. На даний момент – це найсучасніші інструменти розробки.

Основна мета дипломного проєкту полягає в тому, щоб спроектувати та розробити сучасний повноцінний веб-додаток для керування завданнями для фрілансу, використовуючи сучасні інструменти розробки зі зручним дизайном та інтерфейсом.

Завданням дипломного проєктування є дослідження та аналіз аналогічних за функціональністю веб-сервісів, визначення їх переваг і недоліків, проектування і розробка власного веб-додатку фріланс-біржі зі зручним та відлагодженим функціоналом.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ

1.1 Обґрунтування доцільності створення веб-сервісу

У теперішні часи, у все більше людей з’являється комп’ютер, та все більше людей цікавляться комп’ютерними технологіями. Розвиток комп’ютерних технологій та ІТ-сфери відкрили багато нових можливостей для пошуку роботи та самої роботи в режимі «Онлайн».

В 2019 році після появи інфекційної хвороби Covid-19 і її поширенню на весь світ, більшість компаній перейшли онлайновий режим. Через це з’явилось багато нової роботи для «фрілансерів».

Біржа фрілансу – це сервіс, де зібрана велика кількість роботодавців та виконавців. Перші можуть залишити завдання другим, другі – відгукнутися і запропонувати свої послуги. Сучасні сервіси вийшли за рамки «дошки оголошень». Це багатофункціональні сервіси, які роблять пошук і співпрацю максимально комфортними та продуктивними.

Фріланс-біржі, не є основним доходом для фахівців високого рівня, але вони прекрасно підходять для новачків, щоб спробувати фріланс «на смак». Якщо у деяких людей немає досвіду, або вони думають що ще не готові до роботи в офісі, то перші кроки до замовлень допоможуть визначитись та зібрати своє перше портфоліо для майбутнього пошуку роботи на постійній основі. Біржа це ідеальне місце для перевірки своїх навичок та довершення їх, та зрозуміти яка сфера тобі підходить. Також це ідеальне місце для підробітку у вільний час.

Робота на біржі зазвичай підходить усім, хто може працювати за комп’ютером дистанційно. Програмісти, дизайнери, верстальники, копірайтери, перекладачі – це не далеко весь список професій, праця яких оцінюється на цих веб-сервісах.

Причиною створення даного сервісу є забезпечення необхідним середовищем людей, які планують продовжити свою роботу в режимі пандемії. Також, створення цього сервісу дасть змогу більшій кількості людей

здобувати практичні навички у різних сферах ІТ-напрямку, а також отримувати за свою працю грошову нагороду. Для замовників з’явиться більше сервісів, щоб як найшвидше знайти виконавця і отримати бажаний результат, за найменший період часу.

Доцільність проєкту полягає в тому, що даний продукт буде містити усі необхідні інструменти для замовників та виконавців, комфортне середовище для роботи, та дружелюбний інтерфейс. Цей веб-сервіс підходить для тих, хто вже працює і є вільний час для додаткового заробітку, або для новачків, які ще не мають достатнього досвіду для влаштування на постійну роботу. В проєкті буде можливість вільно обирати свою спеціалізацію та здійснювати пошук по доступних замовленнях в обраній категорії.

1.2 Огляди існуючих рішень

В мережі Інтернет вже існують подібні веб-сервіси, з подібним функціоналом та можливостями. Для створення власного проєкту, необхідно проаналізувати вже існуючі рішення.

В цій роботі буде розглянуто три подібні сервіси: Fleelancer.com, Upwork, Freelance.ua.

Першим прикладом буде іноземний найпопулярніший сервіс Upwork (рисунок.1.1), який дає змогу працювати тим, хто готовий співпрацювати з замовниками з інших країн. У цьому сервісі виконавці можуть шукати замовлення безкоштовно, але також існує платна підписка, з якою віздривається більше можливостей: можна відстежувати пропозиції по цікавим проєктам, також замовники зможуть зв’язатись з вами беспосередньо.

Це багатосторінковий сервіс, в якому реалізований достатньо простий та зручний інтерфейс для користувача. Та реалізовані усі функції, які цей проєкт має виконувати.

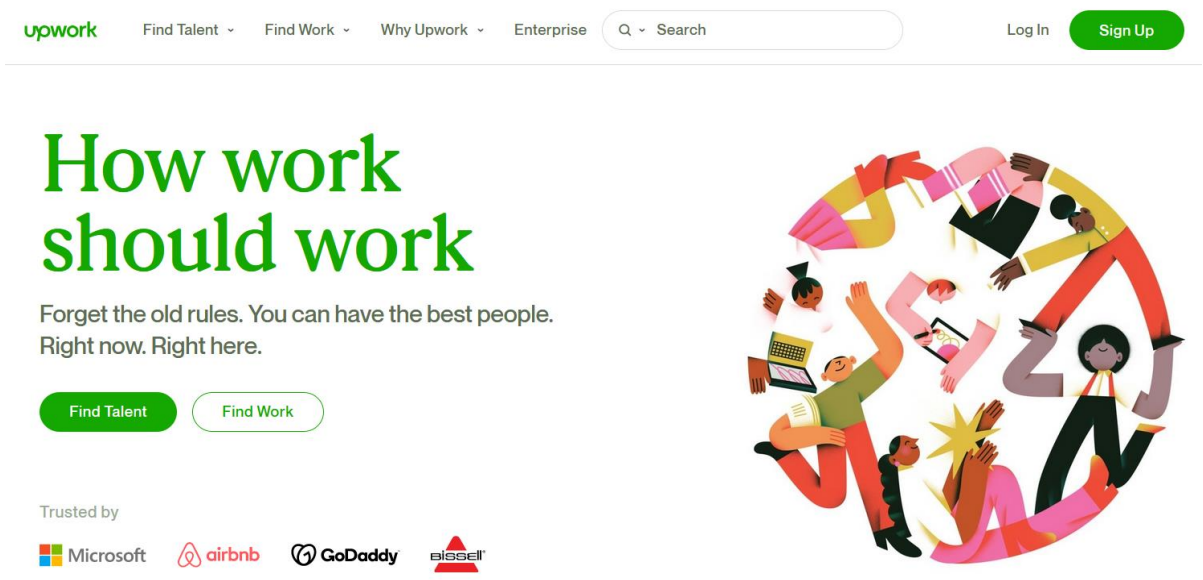

Рисунок 1.1 – Головна сторінка веб-сервісу “Upwork”

Для роботи в цьому сервісі необхідно зареєструватись, для цього необхідно заповними форму реєстрації (рисунок 1.2), в якій необхідно обрати, чи ви замовник, чи ви виконавень.

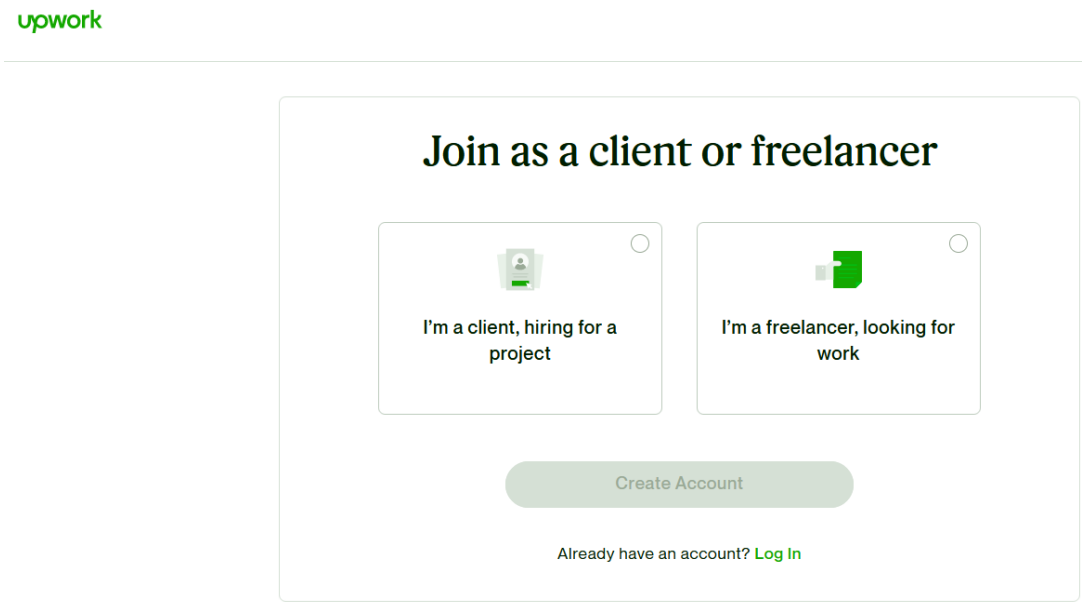

Рисунок 1.2 – Форма реєстрації у сервісі «Upwork»

Перевагами цього сервісу для користувача є те, що у особистому профілі залишається попередня робота і зберігається в портфоліо, і коли замовник буде

обирати виконавця, то він буде бачити чи ви підходите йому чи ні, також виконавці виконуючи завдання, піднімають свій рейтинг та відгудки, за допомогою яких, вони можуть отримувати частіше нові пропозиції. Одною з найбільших переваг в роботі з іноземними замовниками, те що оплата виконується в іноземній валюті. Також на сайті є величезний вибір напрямків роботи, так щоб кожен міг знайти себе.

Перевагами для замовників в тому, що у цьому сервісі є дуже великий вибір виконавців, та можна фільтрувати виконавців по рейтингу, щоб обрати найкращого. Усі виконавці є верифікованні і підтвердженні, тому роботодавцям не варто переживати за безвідповідальність з боку виконавця.

Основний недолік цього сервісу в тому, щоб працювати з замовниками, необхідно володіти англійською мовою, також присутня висока комісія.

Другим прикладом буде український веб-сервіс Freelance.ua (рисунок 1.3). Це найпопулярніша фріланс-біржа в Україні. Яка також є багатосторінковим сайтом, простим в користуванні.

Як ми бачимо на головній сторінці є оформити проект, або стати тим, хто той проект буде виконувати. Також є статистика: скільки проектів взагалом було опубліковано, та скільки користувачів використовують цей сервіс.

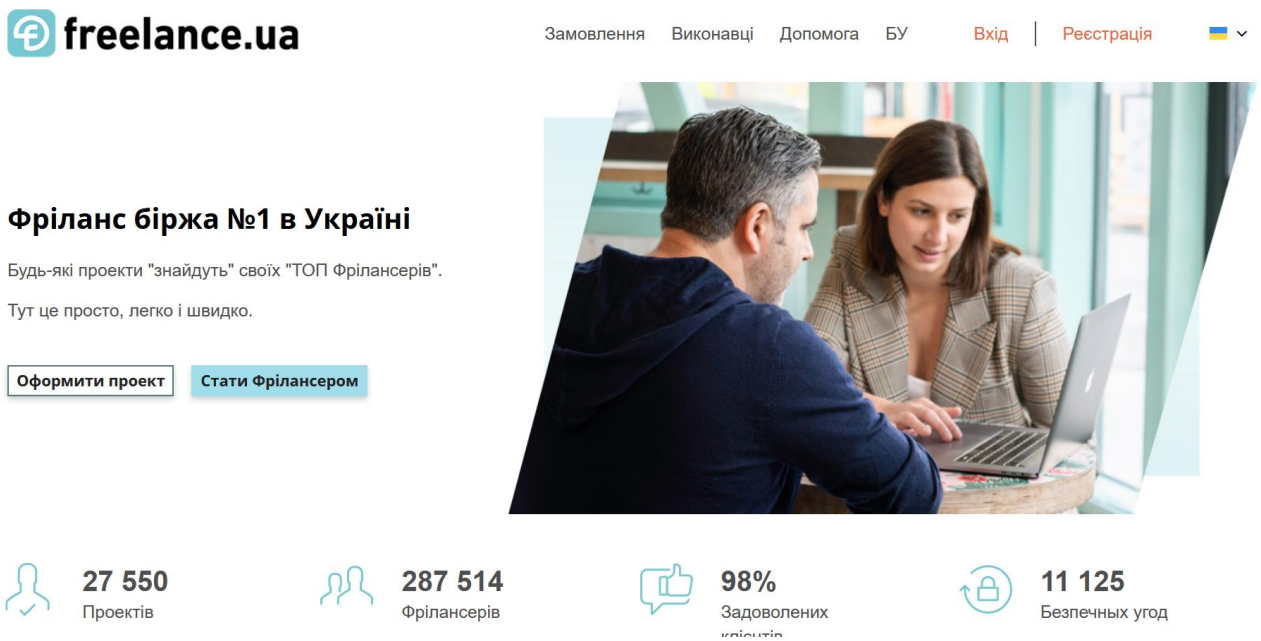

Рисунок 1.3 – Головна сторінка веб-сервісу «Freelance.ua»

На рисунку 1.4 зображено усі напрямки в яких, в яких можна знайти собі замовлення.

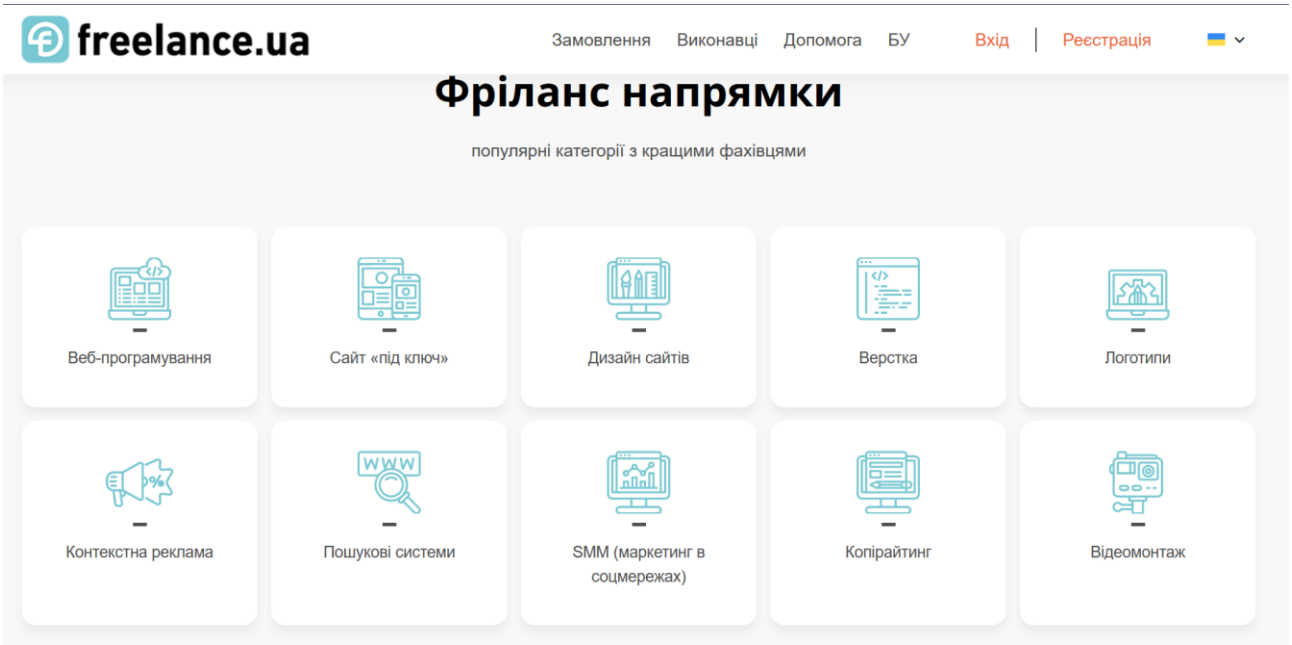

Рисунок 1.4 – Фріланс категорії у «Freelance.ua»

Якщо перейдемо на сторінку «Замовлення» (рисунок 1.5), то побачимо фільтри замовлення, де ми можемо обрати категорію яку бажаємо та усі опублікованні замовлення за останній час.

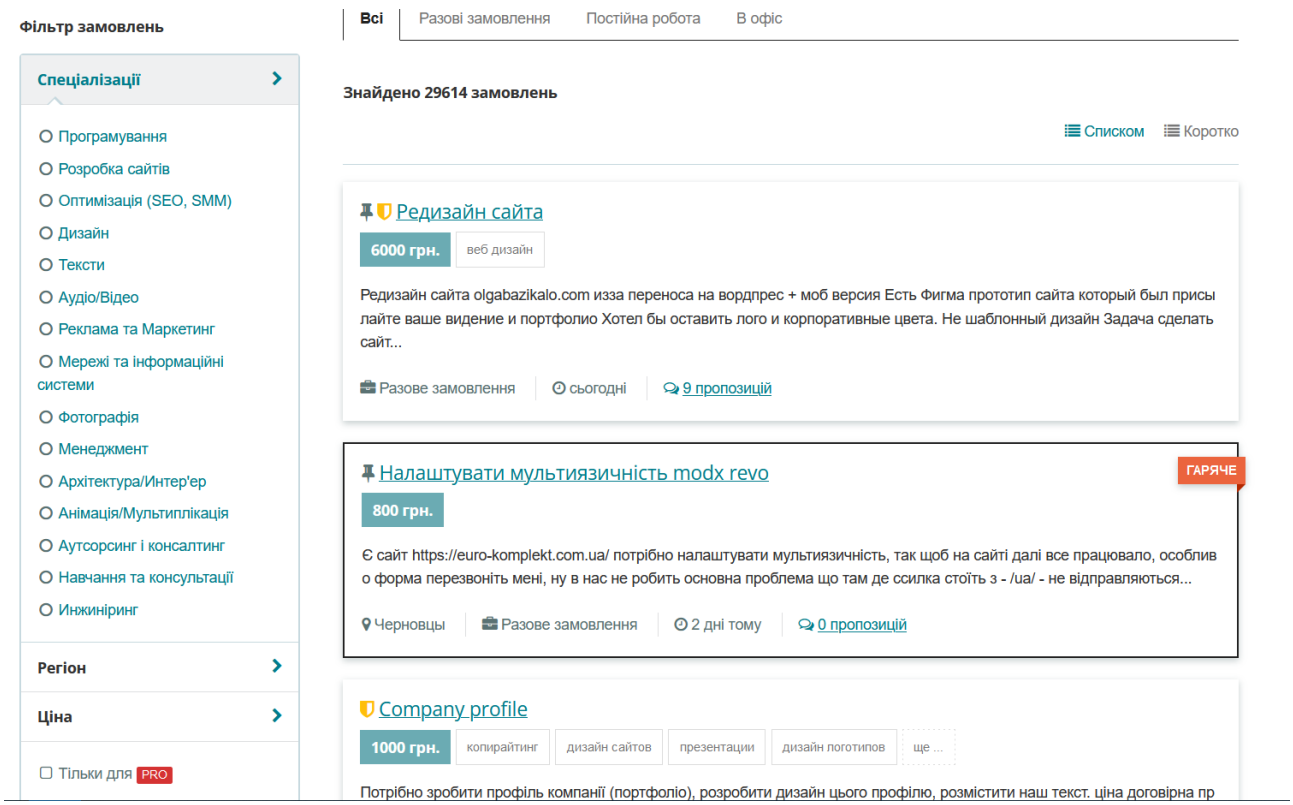

Рисунок 1.5 – Сторінка «Замовлення»

Переваги «Freelance.ua»:

– простий інтерфейс;
– пожливість укласти безпечну угоду;
– зручні інструменти для створення портфоліо;
– бесперервний потік завдань, через популярність самого сервісу;
– виплати: банківська картка (Visa, Mastercard);
– після виконання завдання, замовник залишає відгук;
– сервіс є українським, тому англійська мова тут не обов’язкова.

Найбільшим недоліком є те, що тут існує платна підписка, без якої у виконавця немає можливості відгукатись на замовлення дорожче 400грн.

Третім прикладом буде веб-сервіс Freelancer.com(рисунок 1.6).

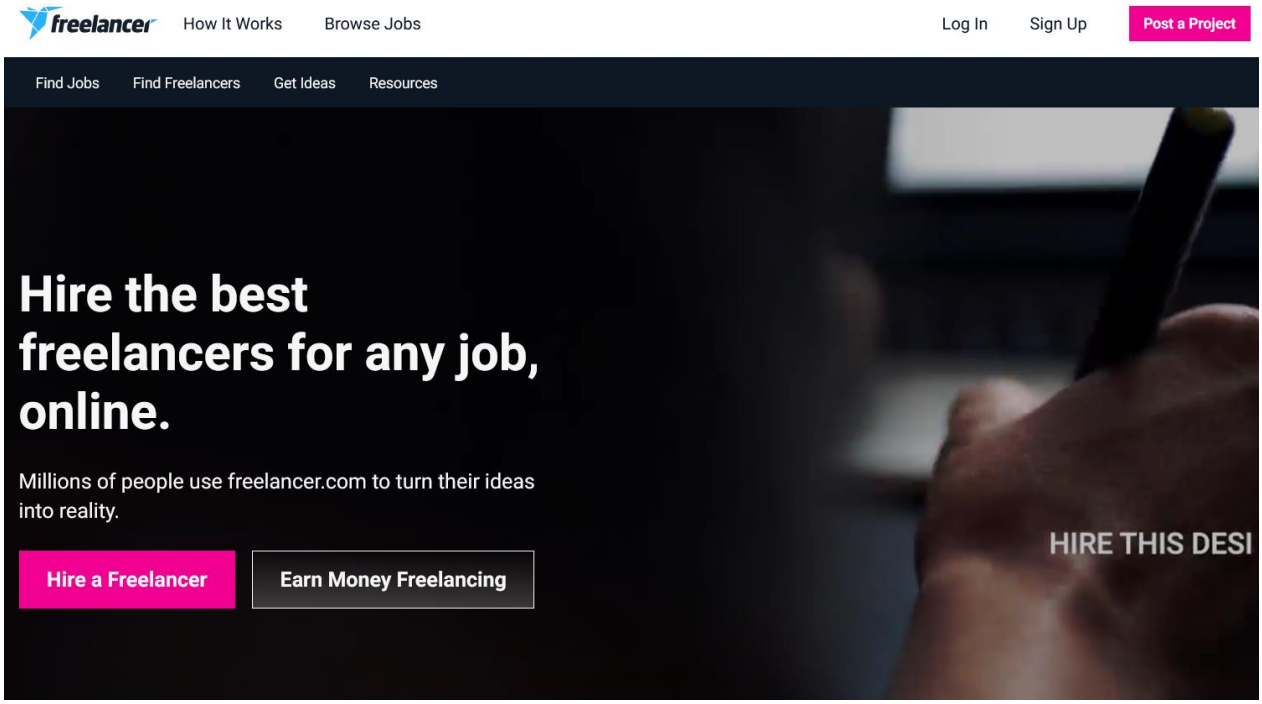

Рисунок 1.6 – головна сторінка «Freelancer.com»

Це ще один іноземний сервіс, який пропонує величезний список напрямків для роботи – 1800 категорій (рисунок 1.7).

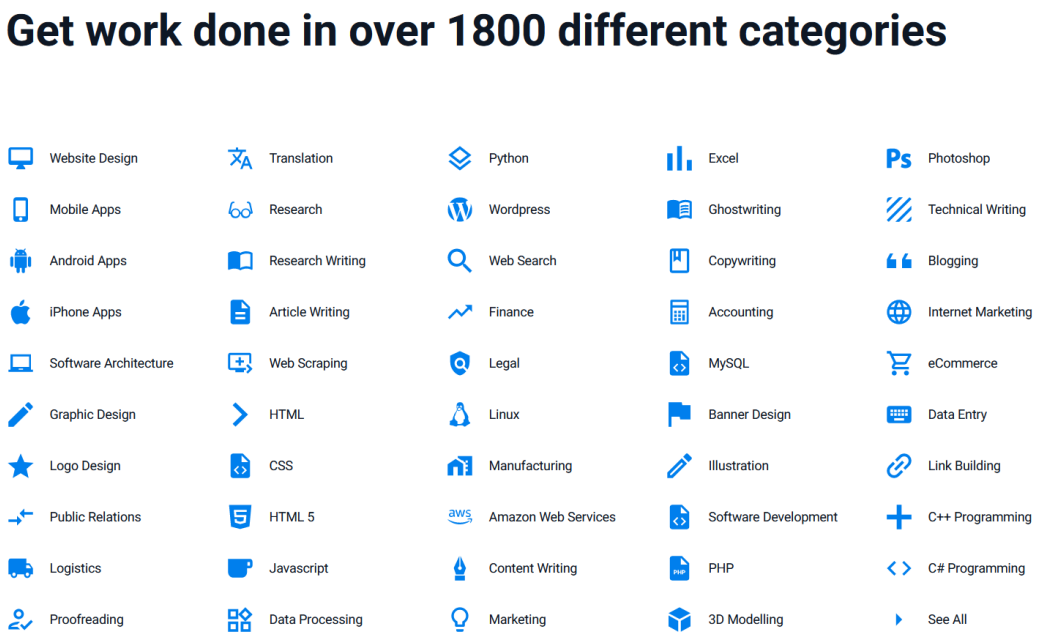

Рисунок 1.7 – напрямки у сервісі «Freelancer.com»

Перевагами цього сервісу є:

– велика кількість категорій та напрямків;
– багато короткострокових проектів;
– менша комісія, в порівнянні з іншими сервісами.

Найбільшим недоліком є те, що при реєстрації не потрібно підтверджувати свою особистість. І через це з'являється багато безвідповідальних осіб, які можуть взятись за якусь роботу і зникнути без причини, завдаючи проблем замовникам.

Отже, фріланс-біржі дуже корисні для новачків, тому що там не потрібен досвід роботи, і ти сам можеш обрати свій графік та завдання, які тобі подобаються, також тобі не прийдеться терпіти свого начальника, тому що ти працюєш сам на себе. Такі сервіси допомагають покращити свої практичні навички, та підзаробити на цьому. Але не варто забувати, що у таких сервісах є не тільки переваги, но і недоліки: основний недолік в тому, що робота для фрілансерів не є стабільною, також немає соціальних пакетів, тобто ніяких відпусток та лікарняних. За часту тобі самому приходиться постійно шукати роботу. Варто не забувати, що на таких сервісах дуже велика конкуренція і на одне завдання може бути 5-10 бажаючих, тому дуже важливе мати прокачені соціальні навички, щоб правильно себе подавати перед роботодавцем.

Можна зробити висновок, що фріланс-сервіси зараз є дуже актуальними через їх доступність та можливість працювати з любої точки світу з різними роботодавцями, не потрібно витрачати час та гроші на поїздки на роботу і тому подібне, самостійно вирішуєш свій графік, коли і скільки працювати на день.

1.3 Постановка завдання

Метою цього дипломного проєкту – створення веб-додатку, який буде схожий за функціоналом з вище згаданими сервісами. Цей сервіс дозволить користувачам знаходити собі завдання та виконувати їх, або публікувати замовлення.

В цій роботі було поставлене завдання розробити full-stack додаток зі зручним користувацьким інтерфейсом, адаптивним дизайном для будь-якого

браузера, та необхідним функціоналом для користувачів, який буде задовільняти їхні потреби.

Веб-сервіс повинен забезпечити авторизацію для різних видів користувачів:

- виконавець;
- замовник.

Даний проект повинен мати такі функціональні можливості:

- реєстрація;
- авторизація;
- добавлення нових замовлень;
- змога відгукатись на замовлення;
- залишення відгуків та коментарів.

Отже, з постановкою визначились, тепер необхідно розглянути усі можливі інструменти для реалізації

2 ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ

2.1 Інструментарій для реалізації проекту

2.1.1 Середовище розробки

Для написання коду усім програмістам необхідно середовище розробки. Обрання середовища залежить від власних вподобань. Для роботи над цим проектом був поставлений вибір між двома найпопулярнішими середовищами: Visual Studio Code від Microsoft та WebStorm від JetBrains.

Visual Studio Code – представляє собою простий текстовий редактор з можливістю підключення різних плагінів для комфортної роботи, які дають можливістю працювати зі всіма можливими мовами програмування для розробки IT продукту. Воно розповсюджується на безкоштовній основі. Серед підтримуваних мов та технологій є: TypeScript, Python, C#, JavaScript та багато інших. На даний час VS Code є однією з найпопулярніших програм для написання коду, але не дивлячись на її високу популярність її функціонал залишається не таким очевидним, через що багато користувачів віддають перевагу іншим продуктам.

Особливості VS Code:

– великим плюсом редактора являється підтримка великої кількості мов програмування;

– проста в засвоюванні;

– додаток підримує розробку на усіх можливих існуючих платформах.

Переваги Visual Studio Code:

– простота і гнучкість;

– велика бібліотека доповнень та готових рішень;

– велика кількість налаштувань інтерфейсу та програми;

– безкоштовне розповсюдження застосунку.

Atom – текстовий редактор, з можливістю встановлення великої кількості розширень для роботи, які дозволяють пристосувати його до будь-яких задач. Створений компанією Github на основі їхньої платформи Electron.

Головна перевага Atom – великі можливості по налаштуванню. Редактор можна налаштувати під свої вподобання. Також цей продукт розповсюджується безкоштовно, що дає можливість спробувати його функціональність кожному.

WebStorm – можливо, найпопулярніше інтегроване середовище для розробки клієнтської частини мовою JavaScript та інших пов'язаних з нею технологій. Створений компанією JetBrains, яка є відома тим, що створює інтегровані середовища розробки. Найпопулярніший їхній продукт – це Java IntelliJ IDEA.

Поширюється даний продукт, як і інші продукти JetBrains платно, але є можливість взяти тридцяти денний тестовий період, щоб спробувати цей застосунок і вирішити чи він вам підходить чи ні, також існує ліцензія для студентів, яка дає можливість користуватись їхніми продуктами за умови підтвердження того факту, що дана особа є студентом. Для цього необхідно авторизуватись, написати своє місце навчання та надіслати фотографію свого студентського квитка.

Мови які підтримує WebStorm:

– для створення продукту на стороні сервера: Meteor та NodeJS;

– для веб-розробки підтримує найпопулярніші JavaScript фреймворки: ReactJS, VueJS та AngularJS;

– для мобільної розробки: React Native;

– для розробки декстопних додатків: Electron.

– Переваги та особливості WebStorm:

– швидкий старт: в цьому середовищі вже вбудована підтримка технологій React, TypeScript, Angular, Vue та багато інших. Це дає можливість

відразу приступити до написання коду, не витрачаючи час на встановлення та налаштування плагінів;

– розумний редактор: працює автозаповнення коду, що дає можливість швидко писати код, використовуючи ключові слова та символи;

– вбудовані інструменти для розробників: запуск та відкладку коду можна робити прямо в редакторі коду, потрібно лише ставити точки для зупинки.

Отже, вибір є великим, кожне середовище має свої переваги та недоліки, які варто враховувати під час вибору інструментарію для розробки.

2.1.2 Розробка клієнтської частини

Для початку необхідно дізнатись з чого складається веб-додаток. На сьогодні веб-застосунок це поєднання клієнтської та серверної частини.

Клієнтська частина – це те що бачить кінцевий користувач на своєму екрані. Це та частина сайту, з якою користувач може взаємодіяти, тобто це інтерфейс за допомогою якого користувач вже здійснює операції з серверною частиною продукту[1].

На сьогоднішній день вже не достатньо зверстати простенький сайт на HTML та CSS. Тепер тільки цими інструментами користувача не зацікавиш. Необхідно створювати красиві анімації, клікабельні кнопки, форми, модальні вікна. Для цього вже необхідний JavaScript. У поєднанні з HTML/CSS і JavaScript вже появляється якийсь функціонал сайту. Уже можна проводити певні обчислення та генерувати новий контент для користувача. Але і цього вже не достатньо, світ Front-end розробки розвивається дуже швидко, появляються нові вирішення проблем, JQuery уже всім набрид, тому появляються нові JavaScript-фреймворки. На початку 2010 років появляється щось середнє між простим сайтом та додатком – Single page application(SPA).

Single Page Application – це архітектура, яка зараз дуже сильно популярна у світі WEB-розробки. SPA – працює так: коли користувач відкриває сторінку, то браузер загружає весь код додатку. Але показує лише ту частину сайту, яка потрібна користувачеві. Коли користувач переходить в

іншу частину додатку, то браузер уже бере загружені дані і показує йому. І якщо потрібно, то динамічно підгружає з сервера потрібний контент без перезавантаження сторінки.

З одної сторони, такі додатки працюють швидко та менше нагружають сервер, але з другої потребують більшої загрузки на початку.

Такі сервіси знижують інтернет-трафік користувача, що є дуже необхідним в даний час.

Односторінкові додатки часто зустрічаються на тих сервісах, де користувач проводить багато часу на одній сторінці та здійснює з нею якісь дії, наприклад:

– передивляється пошту;
– пише пости або коментує чужі;
– дивиться фільми, серіали і тому подібне.

Переваги SPA:

– вони швидкі: перехід між сторінка відбувається швидко, тому що усі дані вже завантаженні при першому попадані на сторінку, потрібно просто підставляти дані, які користувач захотів побачити;
– SPA працюють всюди: все, що потрібно для односторінкових додатків – це підтримка JavaScript;

Недоліки SPA:

– проблеми з SEO;
– залежність від інтернету;
– Single Page Application не для новачків: створити такий додаток на простому HTML/CSS не вийде, необхідно володіти знаннями JavaScript. Для таких додатків використовують сучасні фреймворки: VueJS, ReactJS, AngularJS, EmberJS.

Отже, для створення сучасних додатків необхідно використовувати JavaScript фреймворки, розглянемо найпопулярніші з них.

AngularJS – це JavaScript фреймворк, від компанії Google. Призначений для створення Single page application. Який для написання коду використовує об’єктно-орієнтований підхід.

Переваги Angular:

– детальна документація, яка дозволяє розробникам дізнатись всю необхідну інформацію;

– архітектура та структура додатку, спеціально створена для великих за масштабами проектами;

– Angular може використовуватись разом з TypeScript. Для цього у нього є спеціальна підтримка від розробників.

Недоліки Angular:

– різноманітність різних структур (Components, Pipes, Modules тому подібних), що ускладнює вивчення порівняно з ReactJS та VueJS, у яких лише “Components”;

– повільніша продуктивність в порівнянні з іншими фреймворками.

Організації які використовують Angular: Microsoft, Upwork, Freelancer, Udemy, Youtube, Paypal та багато інших.

VueJS – це ще один JavaScript фреймворк, який потрібен для створення користувацьких інтерфейсів.

Переваги VueJS:

– адаптивність: до цього фреймворку дуже легко пристосуватись, якщо у тебе вже є знання і навички з інних фреймворків;

– детальна документація, яка є представлена різними мовами для розуміння, що дозволяє розробнику прискорити процес поглиблення в нову технологію;

– модульність: легко налаштовується та інтегровується в інші проекти;

– низький поріг входження в порівнянні з іншими фреймворками;

Недоліки VueJS:

– невелика кількість користувачів у порівнянні з іншими гігантами, це означає що при виникненні проблеми буде трішки важче знайти вирішення.

ReactJS[3] – це по суті JavaScript бібліотека, але «в народі» його прийнято називати фреймворком. Використовується для створення користувацьких інтерфейсів, яка використовує функціональний підхід. Це ефективна та гнучка бібліотека від компанії Facebook. На сьогоднішній день це найпопулярніший інструмент для Front-end розробників.

Переваги ReactJS:

– легке засвоєння завдяки детальній та простій документації;

– спеціальне розширення JSX: це розширення до синтаксису мови JS. Дозволяє створювати компоненти використовуючи синтаксис подібний до HTML. React використовує це розширення для написання компонентів;

– React має свій Virtual DOM: він дозволяє взаємодіяти з простим DOM без негативних впливів на продуктивність додатку;

– створення багаторазових компонентів: це дозволяє розробнику не створювати постійно нові компоненти, а використовувати вже створені в різних цілях, що ефективно вплинуть на час роботи розробника та продуктивність додатку;

– велика кількість бібліотек, з якими React реалізовує себе на всі 100%.

Недоліки ReactJS:

– багато «legacy» коду, так як до 16-тої версії React існував зовсім інший підхід до роботи з цим фреймворком – классовий, і з новою версії з’явились так звані «Хуки», що значно спростили роботу, але підтримувати старий код все рівно потрібно.

Angular – хороший вибір для компанії з великою командою розробників, які використовують TypeScript. Його складність зумовлена його спрямованістю на проекти з великими масштабами.

VueJS – хороший конкурент для Angular, який дуже стрімко розвивається. Добре підходить для високоадаптивних інтерфейсів і складних SPA.

ReactJS – вартий уваги, якщо розробник має на меті створити односторінковий додаток, а також зробити його зручним та швидким для кінцевого користувача. Його життєвий цикл, компонентний підхід, роблять React легким у вивченні та створенні професійних додатків та їх підтримці.

2.1.3 Розробка серверної частини

Серверна частина додатку[2] – це та частина застосунку, яку користувач не бачить на власні очі, але взаємодіє з нею через клієнтський інтерфейс. Для програмування серверної логіки розробники використовують такі мови програмування як: Ruby, Golang, Python, Java, NodeJS та багато інших. Обрання мови для проекту залежить від самого завдання та побажань замовника.

Серверну частину пишуть «Бек-енд розробники». Їхня робота складається з:

- організації та роботи з базами даних;
- створення API;
- інтеграції з зовнішніми сервісами;
- розробки базової логіки і алгоритмів роботи додатку.

Розглянемо найпопулярніші інструменти для створення серверної частини. Один з яких є Node JS.

NodeJS[4] – написаний на JavaScript, тому не варто вважати його окремою мовою програмування. Це кросс-платформенне JavaScript середовище, основний фокус якого ставиться на написання серверної логіки та підключення до бази даних. Node JS доступний на усіх найпопулярніших платформах(Windows, MacOS, Linux). Node.js дозволяє розробникам створювати швидкі, масштабовані мережеві програми за допомогою легкого для розуміння коду

Переваги NodeJS:

– при одночасному підключенні до серверу тисяч користувачів NodeJS працює асинхронно, тобто ставить пріорітети і грамотно розподіляє ресурси;

– простий у засвоєнні;

– постійний розвиток;

– наявність NPM (node package manager – це величезна кількість бібліотек, які легко встановлюються).

Тепер розглянемо мову програмування Python, яка на даний момент здобуває велику популярність між розробниками. Python – це інтерпретована мова, що дозволяє заощадити значну кількість часу, що зазвичай витрачається на компіляцію. Вона не є важкою та має низький поріг входження, що збільшує кількість розробників саме цією мобою. У порівнянні з іншими мовами Python має багато важливих відмінностей, нижче подані основні з них:

– керування пам’ятю здійснюється автоматично, тобто розробниками непотрібно хвилюватись щодо його розподілу або звільнення пам’яті;

– операції звичайно виконуються в більш високому рівні абстракції. Це частково результат того, як написана мова, і частково результат розширеної стандартної бібліотеки кодів, що поставляється разом з Python.

Отже, для написання серверної частини мовою програмування Python використовуються його фреймворки. Найпопулярніші з них:

– Django.

– Flask.

– FastAPI.

Кожен з них має свої плюси на мінуси, на які варто зауважити при виборі фреймворку.

2.1.4 Огляд баз даних

Кожен сучасний інтернет-продукт має мати базу даних, в яку зберігаються дані для їх подальшої обробки. Баз даних на даний момент є багато, розглянемо найпопулярніші з них:

MySQL – це реляційна система керування базами даних, розробку якої здійснює компанія Oracle, яка реалізована на основі мови структурованих запитів - SQL. Її найбільше використовують для зберігання даних для веб-сайтів.

Основні переваги використання MySQL:

– висока продуктивність: MySQL – має чітку структуру системи зберігання даних, яка полегшує роботу системних адміністраторам;

– безпека даних: MySQL – відома у світі як найбезпечніша система керування базами даних, яка використовується в найпопулярніших веб-додатках створених найпопулярнішими CMS-системами – Joomla, Wordpress, Drupal;

– цілодобовий час роботи: MySQL забезпечує безперебійну роботу 24/7 і пропонує широкий спектр рішень високої доступності.

Варто зауважити, що ця база даних є безкоштовною, що збільшує її популярність на ринку баз даних.

MongoDB[5] - документоорієнтована система керування базами даних, в якій не потрібно описувати схеми таблиць. MongoDB є NoSQL і використовує JSON-подібні документи.

JSON – це спеціальний формат даних, який дуже зручний у використанні, тому що використовує формат «Ключ - значення», що дозволяє розробникам легко взаємодіяти з даними, які дістаються з бази даних.

Основні переваги MongoDB:

– висока швидкість: завдяки моделі документів, що використовується в MongoDB, інформація може бути вбудована в один

документ, а не покладатися на дорогі операції з’єднання з традиційних реляційних баз даних.;

– простий у використанні;

– гнучкі схеми документів: Модель документа MongoDB дозволяє легко моделювати і маніпулювати практично будь-якою структурою даних.

Отже, кожна база має свої переваги, які варто враховувати при виборі для розробки власного продукту.

2.2 Проектування інтерфейсу

Кожен веб-додаток або веб-сайт має мати свій інтерфейс, який необхідний користувачеві для користування сайту. Варто пам’ятати що інтерфейс має бути зрозумілим та зручним. Додаток має бути приємним на вигляд, щоб не відлякувати потенційного користувача.

Якщо це не простий сайт-лендинг для перегляду інформації, яка розміщена лише на одній сторінці, а великий функціональний додаток з кількома сторінками, то обов’язково необхідна навігація. Навігація – це один з основних елементів в сервісі, який допомагає користувачеві переходити між сторінками та вкладками, щоб дізнаватись весь контент додатку.

На головній сторінці користувач має бачити все що йому потрібно, щоб залишитись на сайті, тобто його має зацікавити зовнішній вигляд сайту та інформація яка подається на головній сторінці.

Найбільш популярний інтерфейс головної сторінки (рисунок 2.1) складається з верхньої частини – «header», де користувач може побачити логотип та навігацію. Найбільша частина інтерфейсу – «main» - де користувач бачить весь контент сайту. Остання частина, та не менш важлива це - підвал сайту – «footer», де користувач часто може побачити усі необхідні посилання.

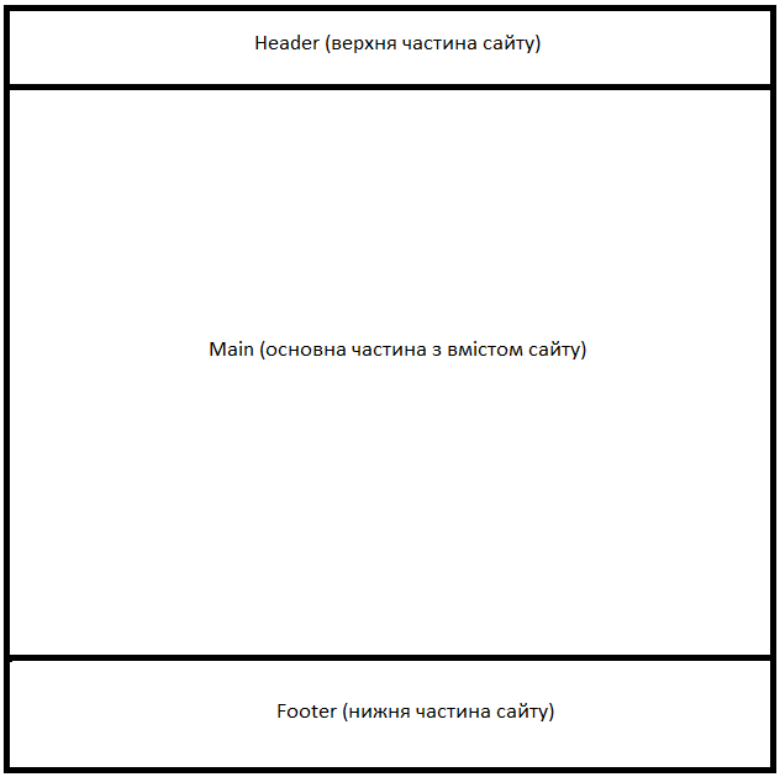

Рисунок 2.1 – Макет інтерфейсу

Такий вид інтерфейсу є найбільш розповсюдженим, звичним та приємним для користувача.

2.3 Проєктування структури веб-сервісу

В даному проєкті існують два види користувачів – гість (рисунок 2.2), який не авторизований на сайті та авторизований користувач – клієнт сайту (рисунок 2.3).

У кожного з них свої права доступу та можливості в користуванні.

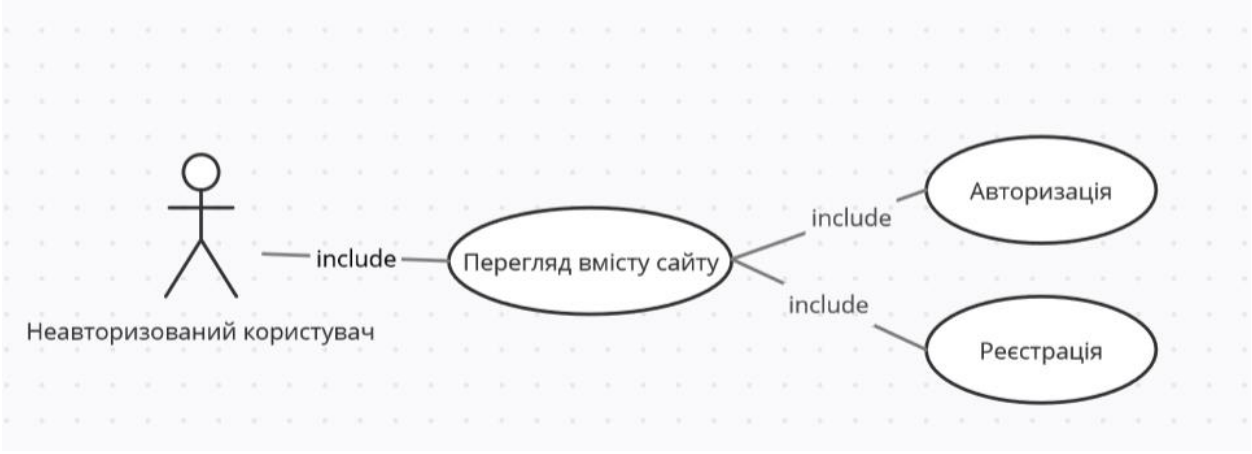

Рисунок 2.2 – Діаграма варіантів використання для неавторизованого користувача

Варіанти усіх можливих дій неавторизованого користувача зображенко на рисунку 2.2

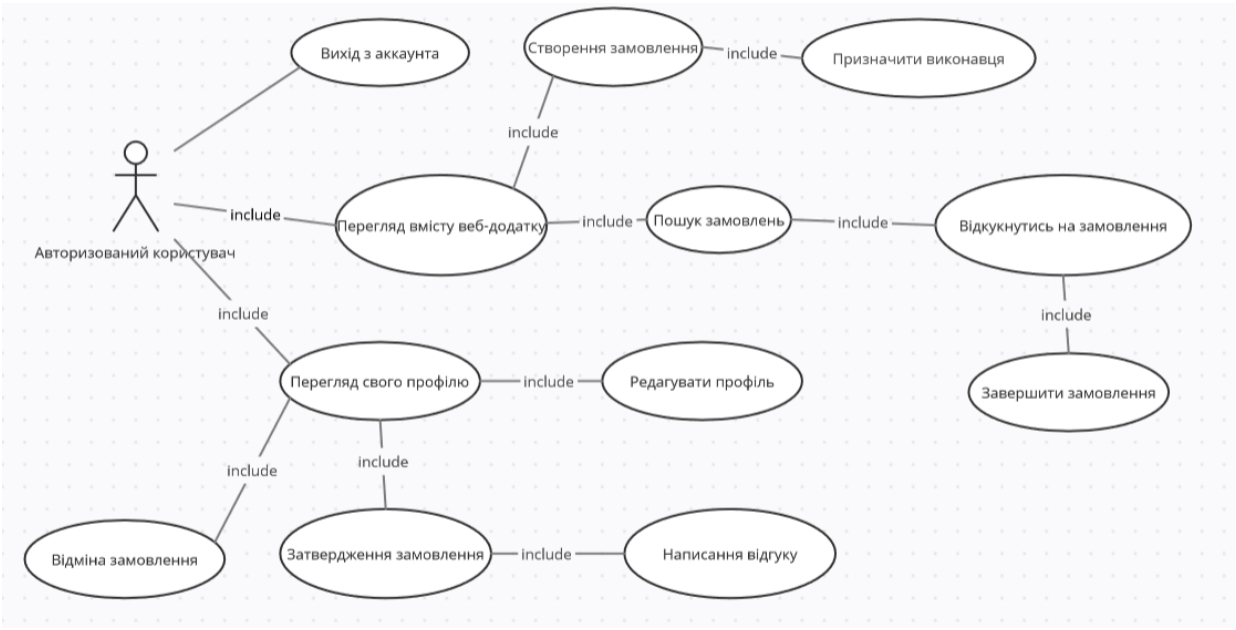

Рисунок 2.3 – Діаграма варіантів використання для авторизованого користувача

Варіанти всіх можливих дій авторизованого користувача зображено на рисунку 2.3.

Отже, коли визначились з правами доступу для користувачів, можна приступати то реалізації проєкту.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

3.1 Опис засобів реалізації

Для розробки проєкту було обрано середовище розробки WebStorm.

Для реалізації клієнтської частини було обрано JavaScript бібліотеку – ReactJS. Який на даний момент є найпопулярнішим інструментом для розробки front-end частини.

React JS за популярністю на голову вище за інші фреймворки, такі як Angular або VueJS. Основні переваги цієї бібліотеки:

– багаторазові компоненти: компонент це частина візуального інтерфейсу з певним функціоналом. В ReactJS розробник має можливість створювати компоненту і налаштовувати її так, щоб можна було використовувати одну ту ж саму компоненту кілька разів, що значно зменшить час роботи для розробника;

– ReactJS є легким в навчанні: причиною того, що ця бібліотека є найпопулярнішою серед інших це її простота та детальна і проста документація, яка дозволяє розробникам швидко оволодіти даним інструментом;

– ця бібліотека використовується не лише для веб-додатків, а і для мобільних застосунків завдяки фреймворку – React Native, який створений на основі React JS;

– існування зручних додаткових інструментів для налагодження в браузерах Google Chrome та Mozilla Firefox.

ReactJS був створений компанією Meta, що говорить нам про якість продукту. Варто зауважити, що ReactJS швидко розвивається і випускаються нові версії, для зручного користування.

Не варто забувати, що для структури інтерфейсу потрібна ще його стилізація. Для стилізації компонентів було обрано CSS Framework – MaterialUI.

Material UI[7] – це найпопулярніший інструмент стилізації для ReactJS. Він дозволяє використовувати вже готові компоненти, які необхідно лише налаштувати під свої потреби. Також він дозволяє швидко та без великих зуссиль адаптувати компоненти для різнних видів пристроїв.

Для збереження даних на клієнті, які приходять з бази даних було обрано найпопулярніший так званий стейт-менеджер – Redux. Стейт-менеджери - це інструменти для винесення усієї логіки за межі компонент та збереження усіх даних в загальному сховищі даних.

Сховище даних, так званий state, дозволяє розробникам діставати усю необхідну інформацію в любому місці та в любій компоненті.

Для відправлення запитів з клієнтської частини до серверної частини було обрано бібліотеку “axios js”. Яка дозволяє швидко та легко налаштовувати тип запиту (POST, GET, PUT, PATCH, DELETE), внесення даних в запит та усі необхідні операції для відправлення запиту.

Для реалізації навігації було обрано бібліотеку “react-router-dom”. Ця бібліотека дозволить добавити динамічну навігацію, щоб переходити між сторінками в додатку без перезавантаження сторінки після зміни URL-стрічки, що є одним з найбільших переваг React JS.

Для реалізації серверної частини було обрано інструмент Node JS.

Для роботи з NodeJS було обрано Node фреймворк Express JS. Це найпопулярніший фреймворк для розробки веб-додатків на Node JS. Цей інструмент полегшує організацію функціональності вашої програми за допомогою проміжного програмного забезпечення та маршрутизації. Він додає корисні утиліти до об’єктів HTTP Node.js і полегшує відтворення динамічних об’єктів HTTP.

Переваги ExpressJS:

- легко налаштовується;
- дозволяє визначати маршрути програми за допомогою методів HTTP та URL-адрес;
- швидко та легко розробляє веб-додатки Node.js.

Express дозволяє відслідковувати який тип запиту був відправлений з клієнтської частини та який шаблон використати в результаті.

Базою даних було обрано MongoDB. MongoDB - це нереляційна система управління базами даних (СУБД) з відкритим вихідним кодом, яка використовує гнучкі документи замість таблиць і рядків для обробки та зберігання різних форм даних. Документи MongoDB або колекції документів є основними одиницями даних. Оскільки MongoDB використовує динамічний дизайн схеми, користувачі мають неперевершену гнучкість під час створення записів даних, запитів до колекцій документів за допомогою агрегації MongoDB та аналізу великих обсягів інформації.

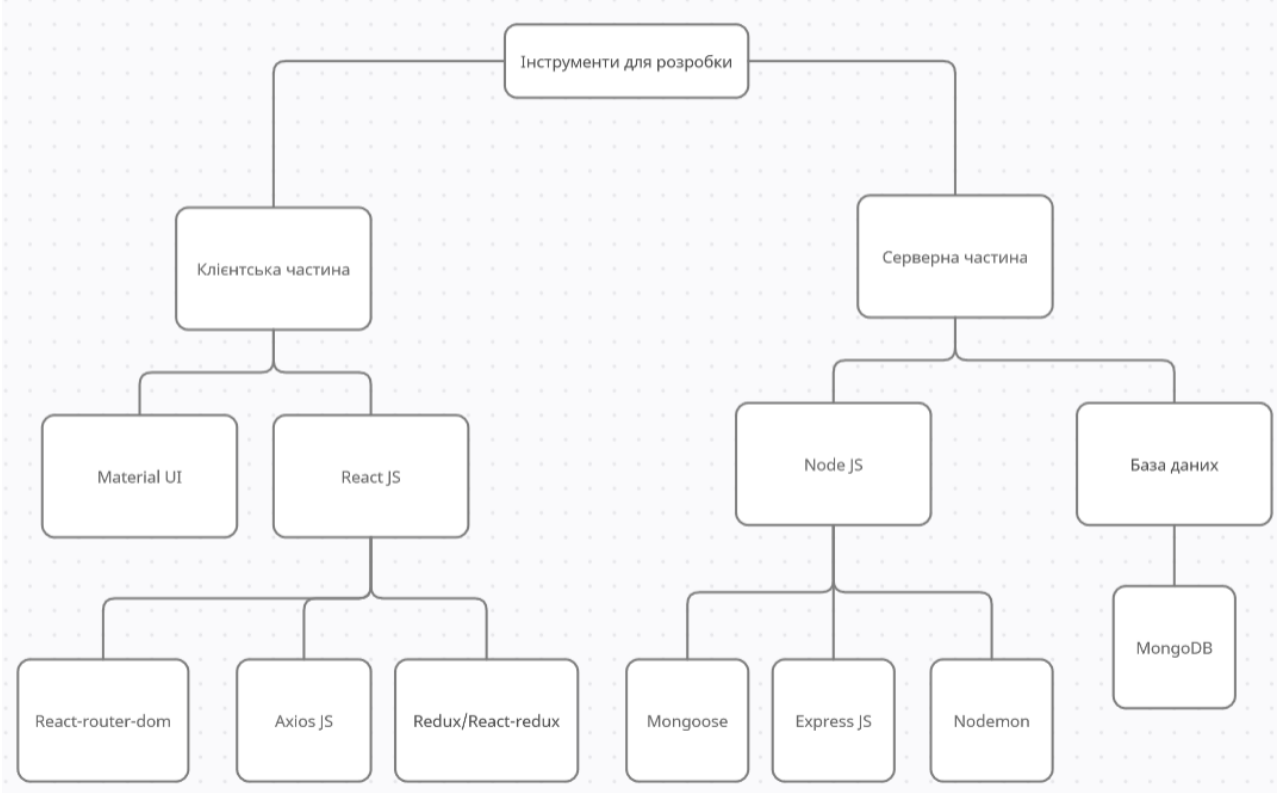

Рисунок 3.1 – Використаний інструментарій

Отже, після вибору необхідного інструментарію (рисунок 3.1), можна приступати до роботи над веб-додатком.

3.2 Створення бази даних

Для створення бази даних необхідно перейти на головну сторінку офіційного сайту MongoDB (рисунок 3.2).

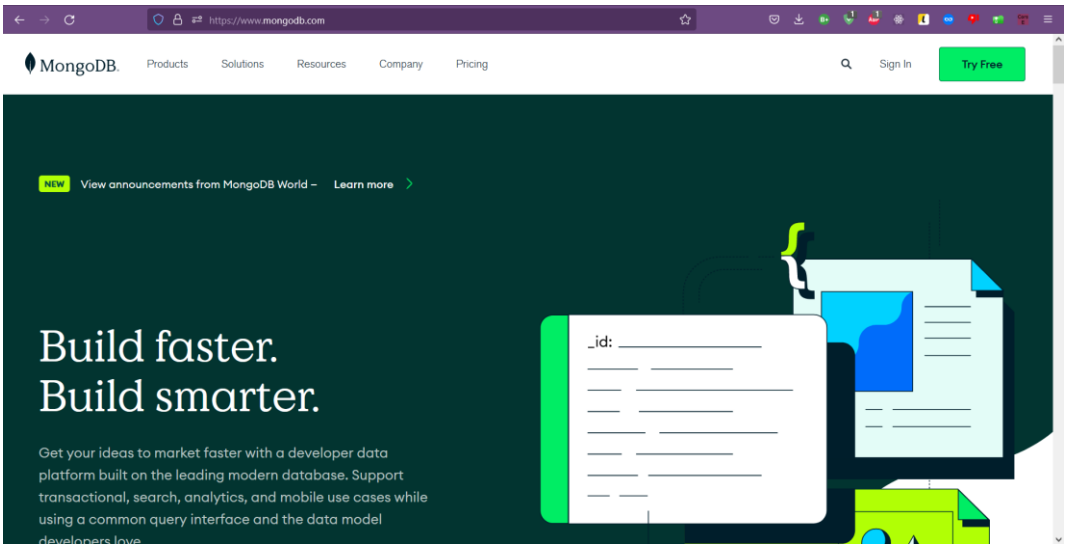

Рисунок 3.2 – Офіційна сторінка MongoDB

Після цього необхідно створити аккаунт в даній системі або авторизуватись за допомогою власної електронної скриньки.

Після авторизації, необхідно створити власний кластер, де будуть зберігатись ваші бази даних. У сервісі MongoDB є три види кластерів (рисунок 3.3) – Serverless, Dedicated та Shared. Перші два видаються на платній основі.

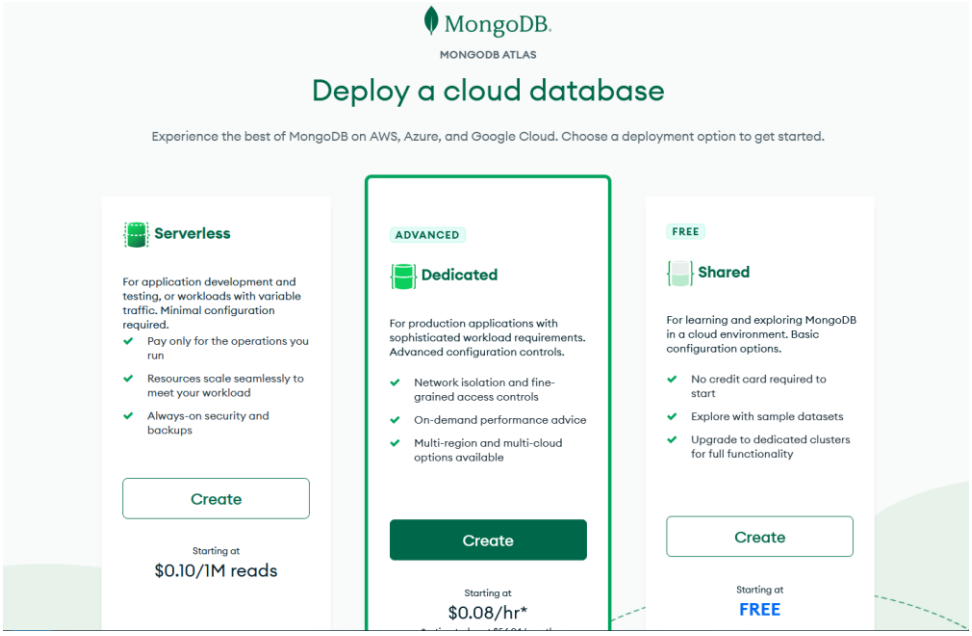

Рисунок 3.3 – Типи кластерів

Для даного проєкту достатньо безкоштовної версії. Після обрання типу кластера, далі дають можливість обрати регіон (рисунок 3.4) в якому ця база даних буде розташована:

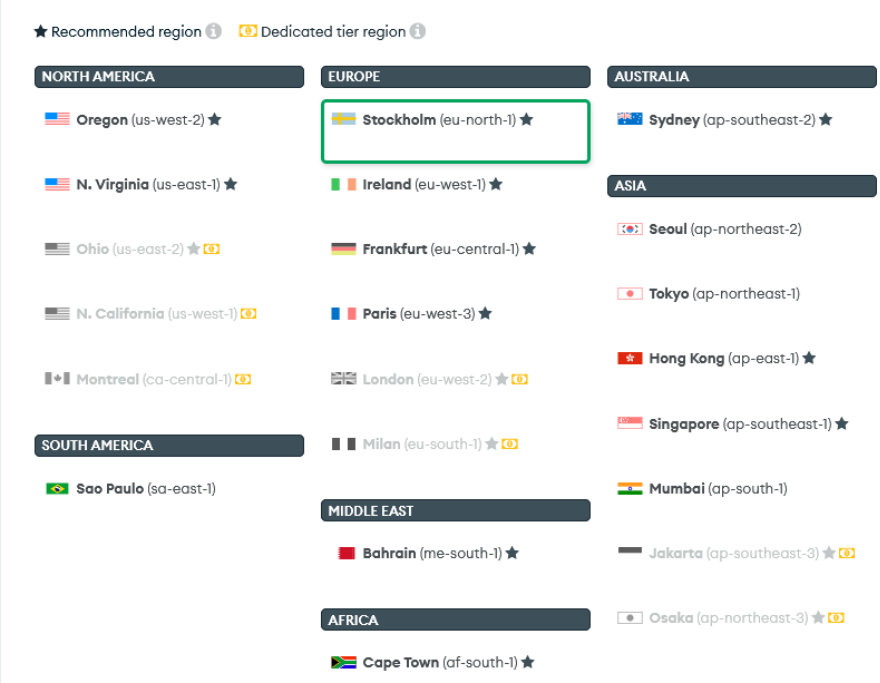

Рисунок 3.4 – Регіони для кластерів

Бажано обирати найближчий до вашого місцезнаходження, щоб не було проблем з інтернет-з’єднанням.

Після створення кластера, появиться можливість здійснити з’єднання до бази даних (рисунок 3.5).

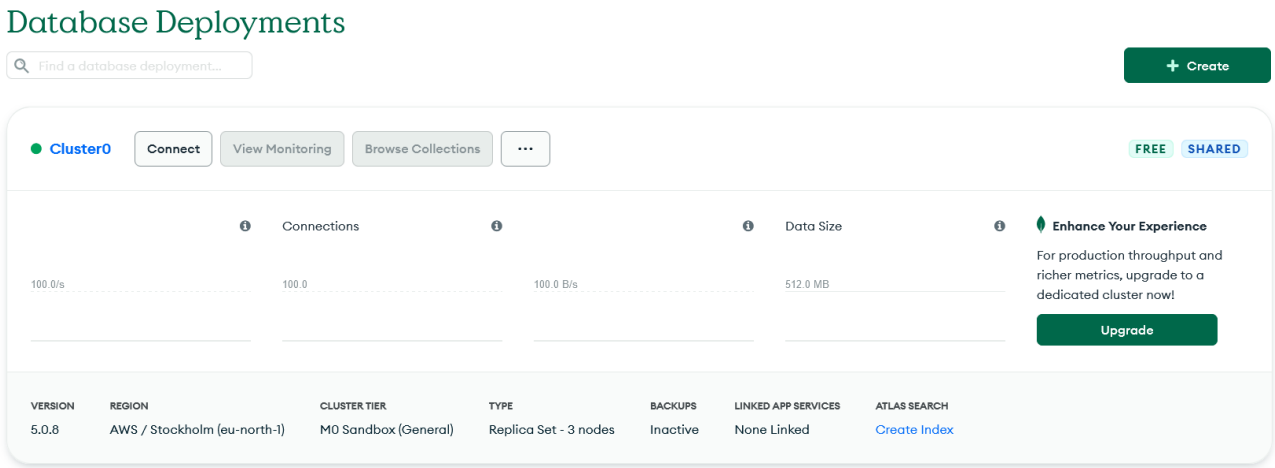

Рисунок 3.5 – Створений кластер

З’єднання до бази даних буде реалізоване інструментом Node JS та побічними застосунками до нього.

3.3 Написання серверної частини веб-додатку

Перед написанням коду, необхідно встановити Node JS на персональний комп’ютер, це можна зробити з офіційного сайту NodeJS.

Найголовніше в написанні серверної частини – це підключення до бази даних. Для цього необхідно встановити додаткову бібліотеку «mongoose», завдяки якій зможемо здійснити підключення до MongoDB.

Також, для зручної роботи необхідно інсталювати додатковий пакет «nodemon» (рисунок 3.6), який потрібен для того, щоб серверна частина автоматично оновлювалась при будь-яких змінах в коді, без необхідності оновлювання вручну.

```
PS G:\diploma-project\diploma-backend> npm i nodemon

up to date, audited 221 packages in 1s

27 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Рисунок 3.6 – Встановлення “nodemon”

Після встановлення усіх необхідних пакетів, можна приступати до під’єднання до бази даних.

Для цього необхідно перейти в свій профіль на офіційній сторінці MongoDB та натиснути кнопку «Connect» (рисунок 3.7).

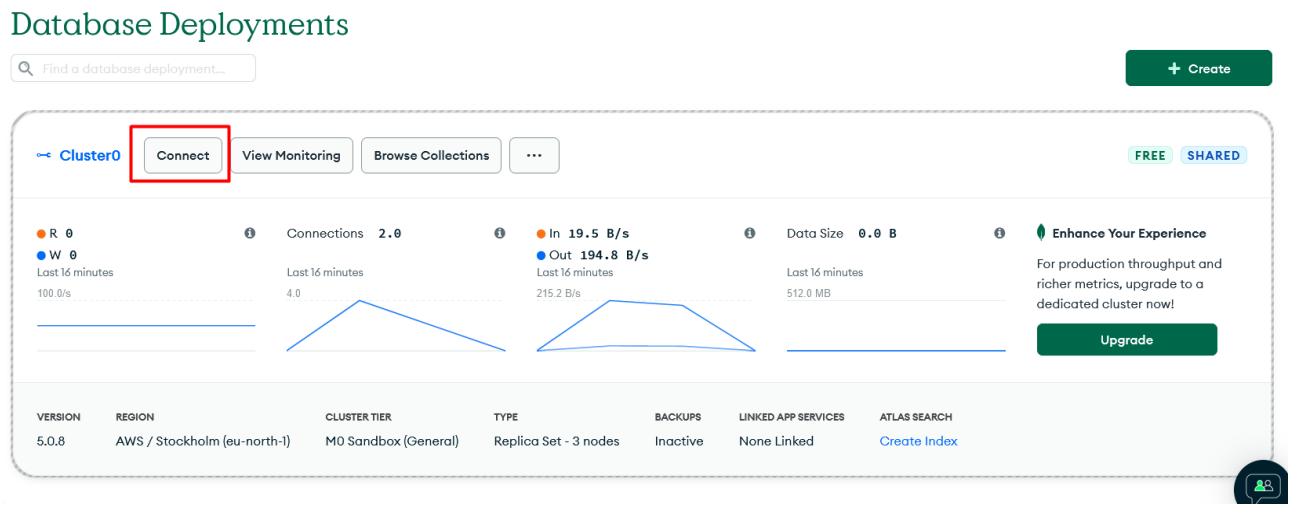

Рисунок 3.7 – Кнопка для під’єднання до бази даних

Після якої з’являться усі можливі способи підключення до бази даних (рисунок 3.8):

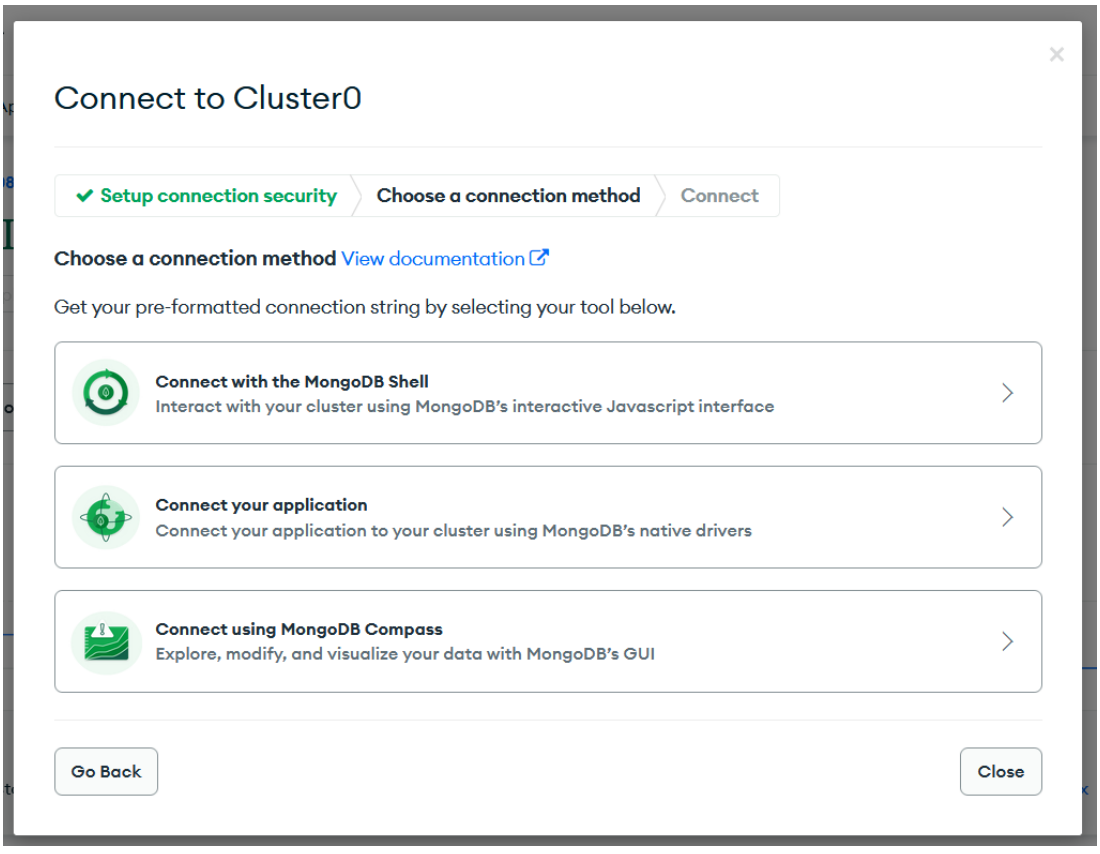

Рисунок 3.8 – Види підключень до бази даних

В модальному вікні можна обрати спосіб, в цьому проекті був обраний спосіб «Connect your application», що дозволить під’єднатись до бази даних за допомогою коду.

Після цього, уже можна починати писати серверну частину для веб-сервісу.

За допомогою бібліотеки «mongoose» здійснюється під’єднання до бази даних. Після написання відповідного коду, необхідно запустити код в терміналі за допомогою команди «npm start» (рисунок 3.9), після чого можна побачити чи з’єднання вдалось чи ні:

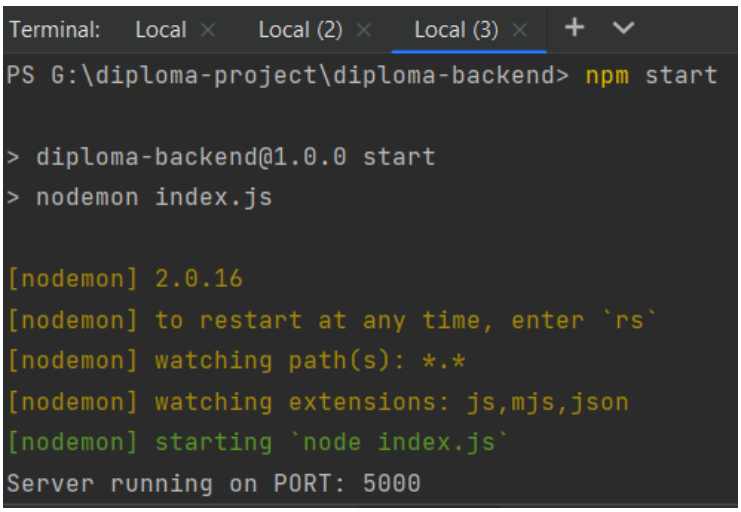

Рисунок 3.9 – Запуск сервера

З’єднання пройшло успішно, після цього можна створювати логіку для серверної частини.

3.4 Опис та реалізація власного дизайну

Для початку необхідно ініціалізувати React JS проєкт, раніше для успішного запуску необхідно було налаштовувати проєт вручну за допомогою пакувальника Webpack, але для розробників це забирало багато часу, тому компанія «Meta» (засновник React JS) створила коротку команду, яка дозволяє ініціалізувати проєкт вже з налаштованим Webpack для цього необхідно ввести команду «npx create-react-app folderName» в термінал (рисунок 3.10):

```
PS G:\test> npx create-react-app client

Creating a new React app in G:\test\client.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

[                    ] - idealTree:client: sill idealTree buildDeps
```

Рисунок 3.10 – Ініціалізація проекту

Цей процес не є швидким та займе кілька хвилин. Після того як все встановиться, папка має виглядати ось так (рисунок 3.11):

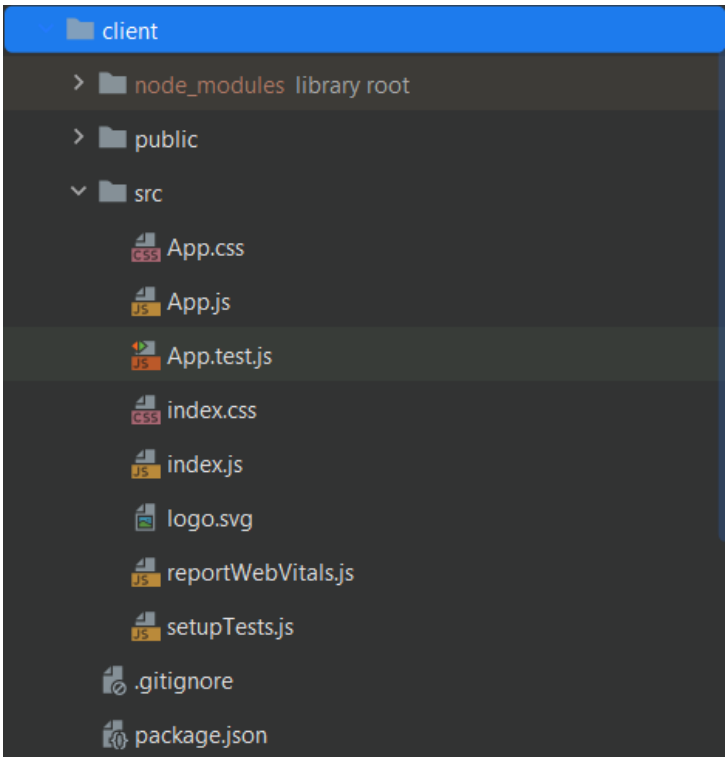

Рисунок 3.11 – Структура проекту

Після ініціалізації проекту з’явиться можливість запустити його командою «npm start». Якщо все пройшло успішно, то з’явиться повідомлення про успішно запущений проєкт в терміналі. Проєкт запущений на локальному сервері із портом 3000 (рисунок 3.12):

```
Compiled successfully!

You can now view client in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Рисунок 3.12 – Запущений проект

Для подальшої роботи необхідно встановити усі додаткові пакети (axios, redux, react-redux, react-router-dom, material ui) (рисунок 3.13):

```
PS G:\test\client> npm i axios redux react-redux @mui/material react-router-dom
[...................] \ reify: timing arborist:ctor Completed in 1ms
```

Рисунок 3.13 – Встановлення додаткових пакетів

Клієнтська частина розроблена за допомогою ReactJS та його стилізація Material UI, зображення головної сторінки на рисунку 3.14, 3.15.

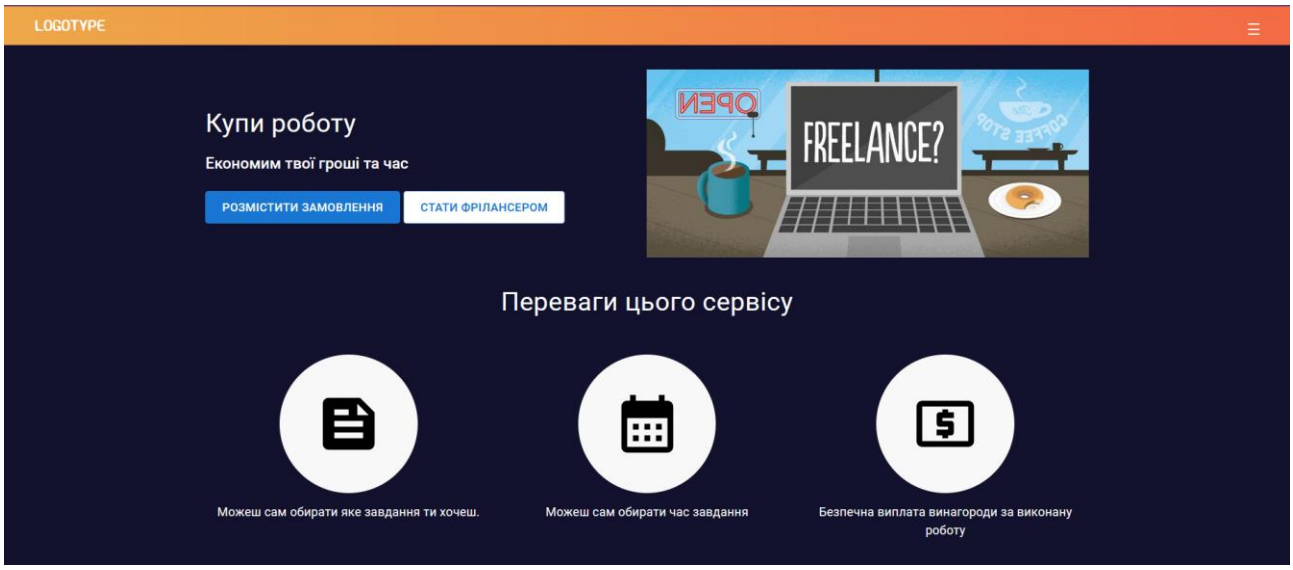

Рисунок 3.14 – Головна сторінка

На рисунку 3.14 зображена частина головної сторінки з навігаційною панелю та основною інформацією щодо сервісу.

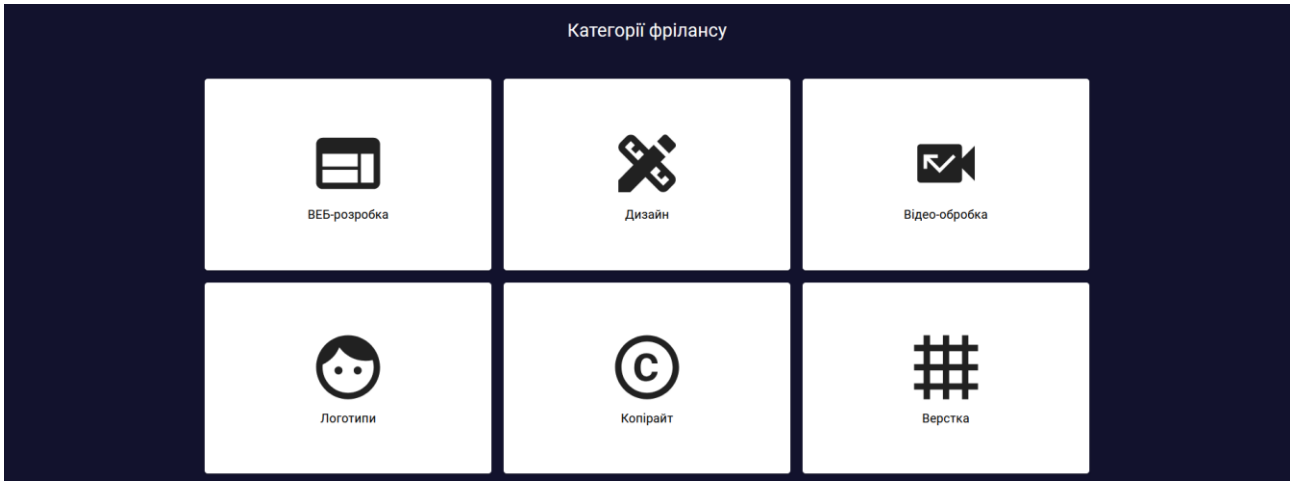

Рисунок 3.15 – Головна сторінка

Головна сторінка – це лице веб-сервісу, вона має зацікавити користувача відразу. На ній не має бути лишньої інформації, лише те що необхідно потенційному користувачеві.

Окрім головної сторінки, також присутня сторінка авторизації, яка необхідна для цього сервісу. Сторінка авторизації являє собою просту форму для внесення даних (рисунок 3.16).

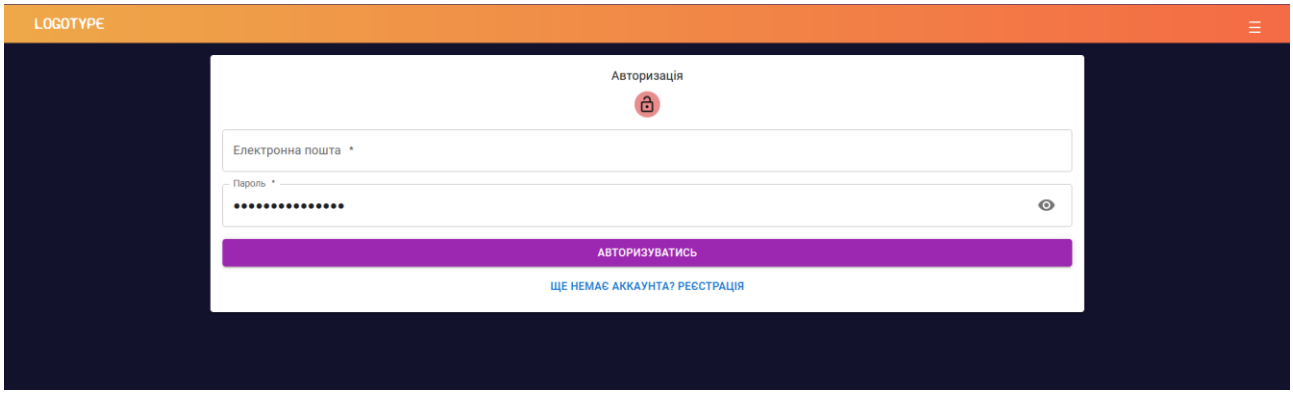

Рисунок 3.16 – Сторінка авторизації

Також необхідна сторінка для реєстрації користувача, вона є подібною до авторизації. На ній користувач вводить усі необхідні дані для реєстрації (рисунок 3.17).

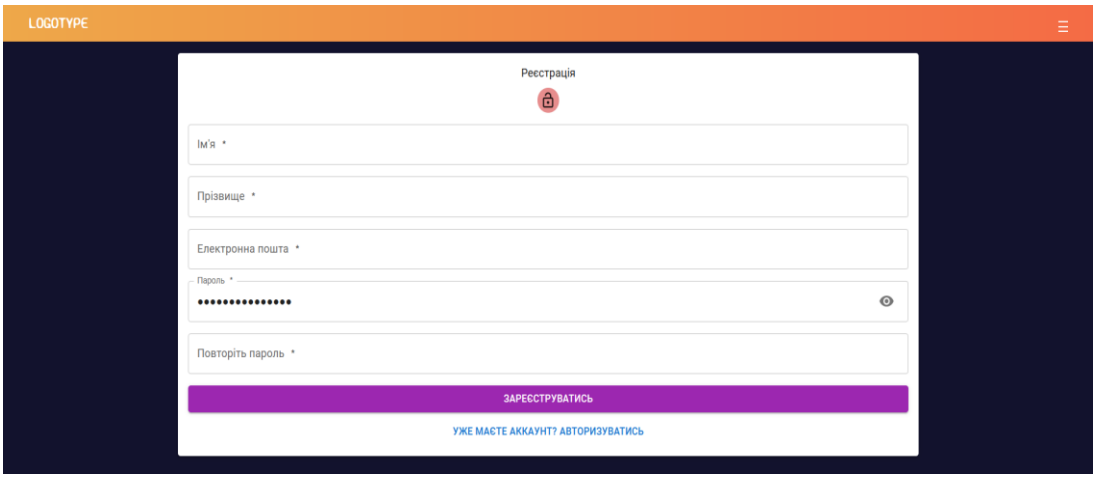

Рисунок 3.17 – Сторінка реєстрації

Була реалізована сторінка профіля, яка містить всю інформацію яка потрібна користувачеві: замовлення та відгуки (рисунок 3.18).

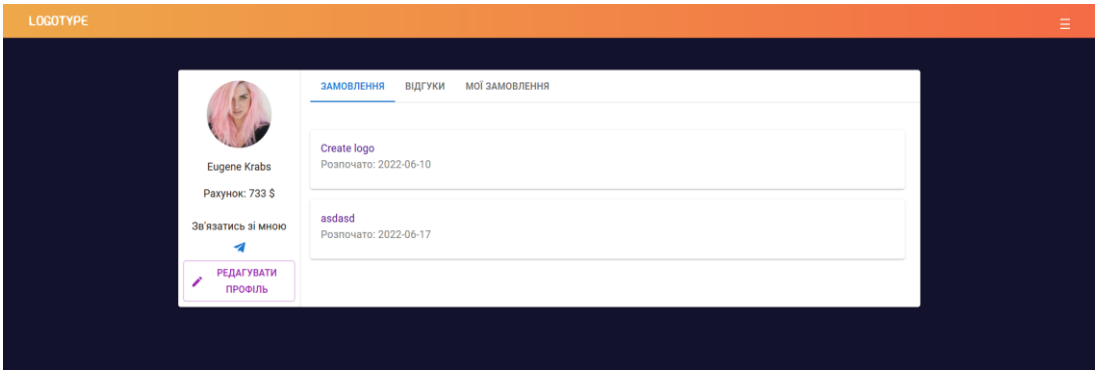

Рисунок 3.18 – Сторінка профілю

Сторінка замовлень виглядає ось так (рисунок 3.19)

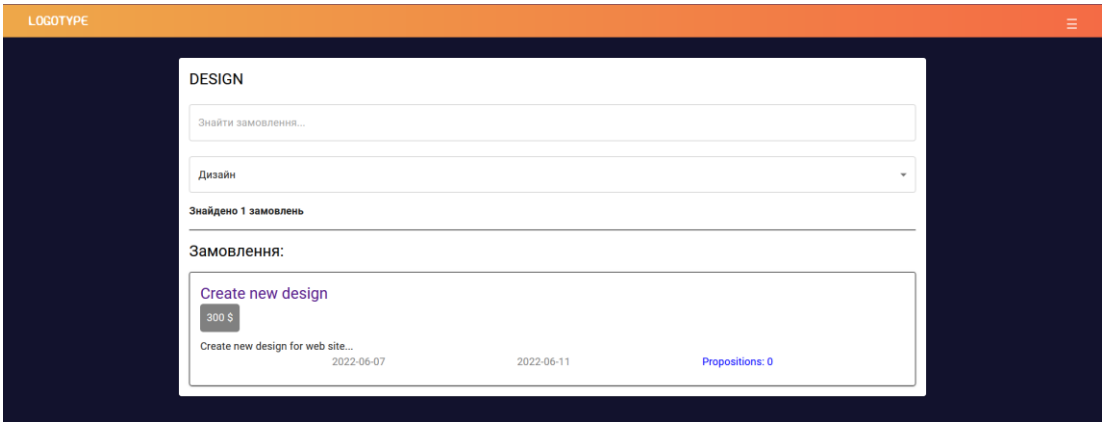

Рисунок 3.19 – Сторінка замовлень

Перейдемо до наступого етапу – адаптивність.

Варто не забувати, що кожен сайт має бути адаптивним. Адаптивний сайт – це сайт з дизайном, який виглядає приємно на будь яких пристроях (телефони, ноутбуки або планшети).

Якщо сервіс є адаптивним, то він охопить більшу аудиторію людей, що є дуже важливим для популярності додатку.

Отже, розглянем як виглядає цей додаток на різних пристроях. Головна сторінка в розширенні смартфона виглядає ось так (рисунок 3.20).

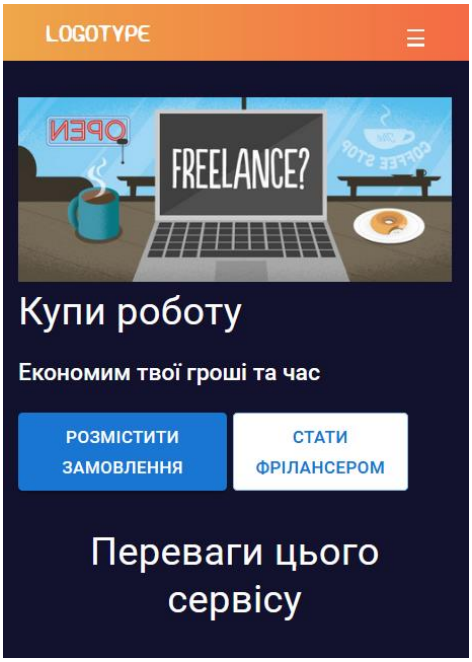

Рисунок 3.20 – Адаптивна головна сторінка

Сторінка реєстрації в розширенні смартфона має такий вигляд (рисунок 3.21).

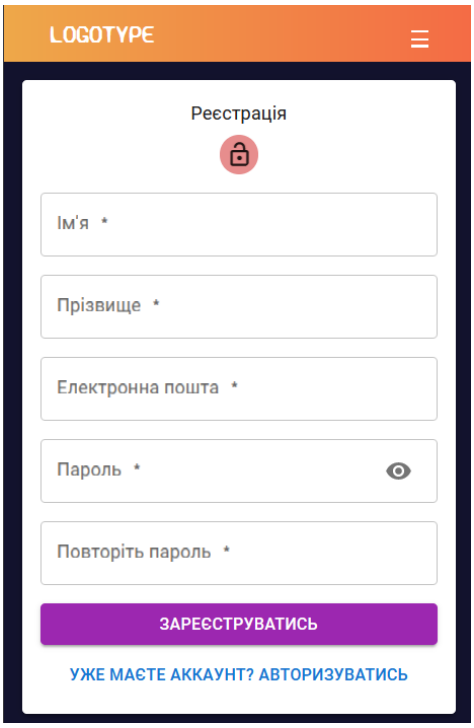

Рисунок 3.21 – Адаптивна сторінка реєстрації

Отже, написання серверної частини та клієнтської завершено, тепер можна приступити до тестування веб-сервісу.

3.5 Тестування веб-сервісу

Для початку спробуємо створити новий аккаунт, для цього переходимо на сторінку реєстрації. Навмисно пишемо пароль маленької довжини і очікуємо на реакцію додатку (рисунок 3.22).

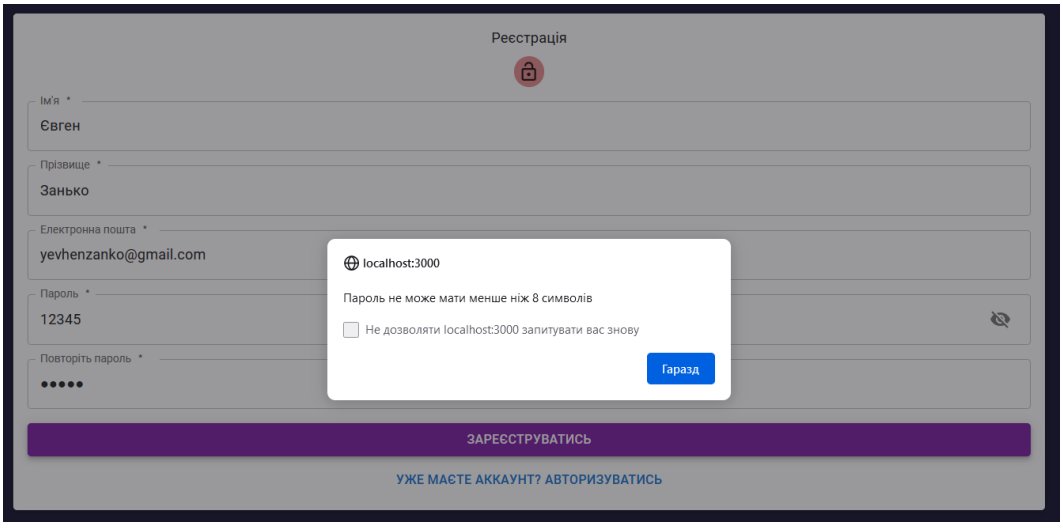

Рисунок 3.22 – Помилка про меншу кількість символів

Бачимо що спрацювала перевірка на кількість символів в паролі. Тепер спробуємо ввести відповідний пароль (рисунок 3.23).

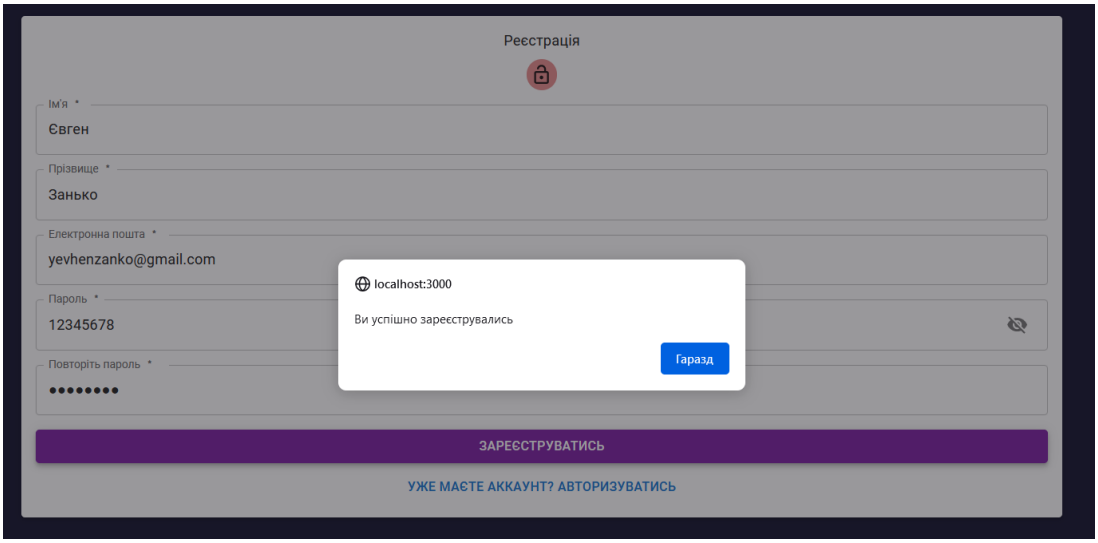

Рисунок 3.23– Успішна реєстрація

Після успішної реєстрації, ви залишаєтесь авторизованими на сайті та повертаєтесь на головну сторінку, де у вас вже є права доступу для створення замовлень.

Тепер спробуємо створити нове замовлення для інших користувачів (рисунок 3.24).

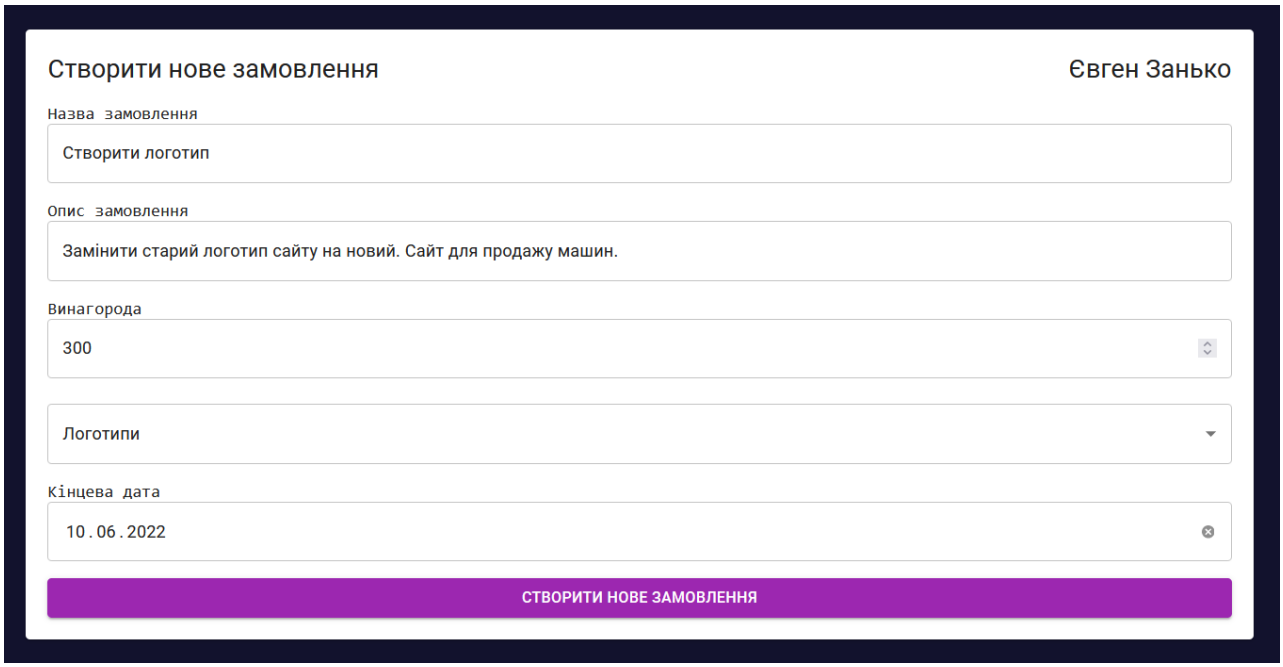

Рисунок 3.24 – Створення замовлення

Усе пройшло добре і замовлення створилось, після чого користувача автоматично відправило на сторінку замовлення (рисунок 3.25).

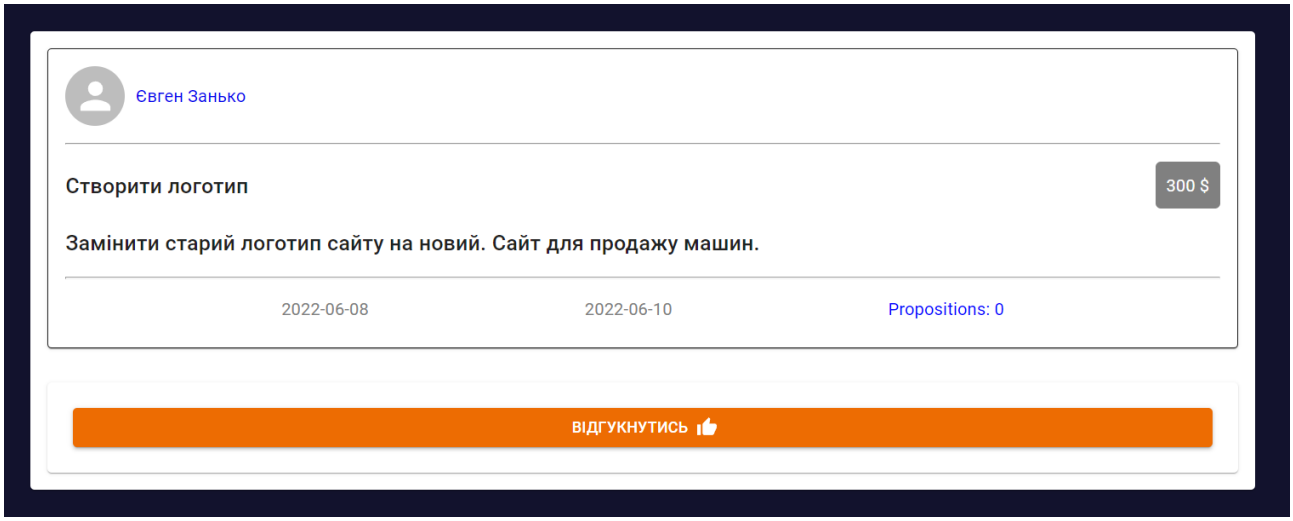

Рисунок 3.25 – Сторінка замовлення

Цю сторінку замовлення можуть бачити інші користувачі, а авторизовані користувачі можуть залишати свою пропозицію на виконання (рисунок 3.26).

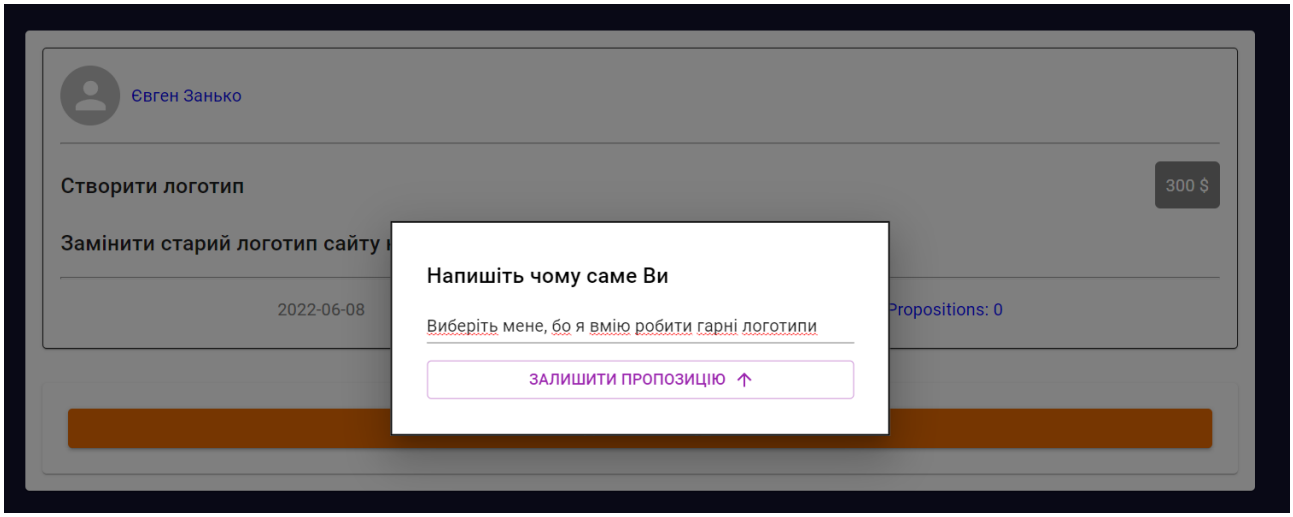

Рисунок 3.26 – Форма для пропозицій

Після того як користувач залишив свою пропозицію, замовник який створив це замовлення, може переглянути усіх бажаючих на це замовлення (рисунок 3.27)

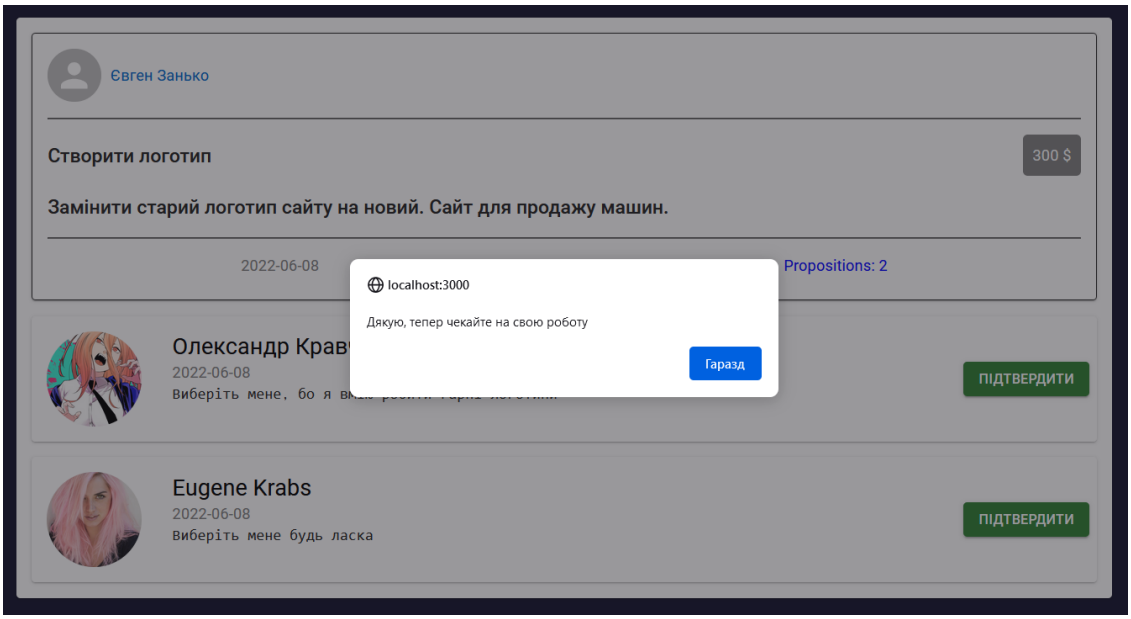

Рисунок 3.27 – Перегляд пропозицій

Після того, як замовник обрав виконавця для свого замовника, замовлення перейшло в статус «inProccess» і замовнику залишається лише зачекати на виконання.

Натомість виконавець, який взявся за це завдання, коли воно буде готове, має можливість здати його та перевірку, залишишки посилання на Github-репозиторій (рисунок 3.28).

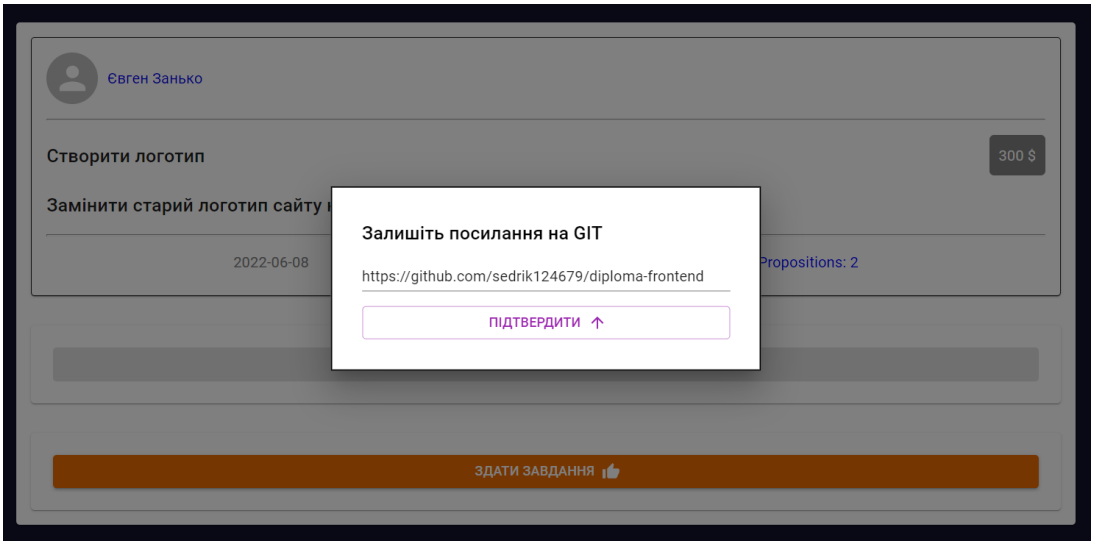

Рисунок 3.28 – Форма для здачі замовлення

Після чого замовник може перейти в свій профіль і переглянути свої замовлення та їх стан виконання (рисунок 3.29).

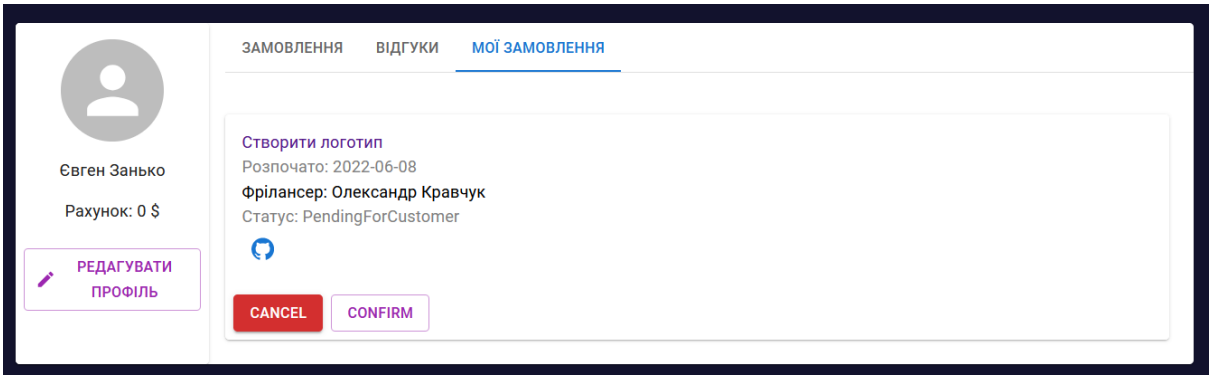

Рисунок 3.29 – Сторінка профілю

Як бачимо замовлення вже в статус «PendingForCustomer» - що значить, що воно здане на перевірку. Тут замовник може перейти по лінку на GitHub. Після перегляду роботи він обирає або відмінити замовлення і воно повернеться до початкового стану «В очікуванні» або прийняти завдання натиснувши кнопку «Confirm» після чого замовника перекине на сторінку для написання рецензії про фрілансера (рисунок 3.30).

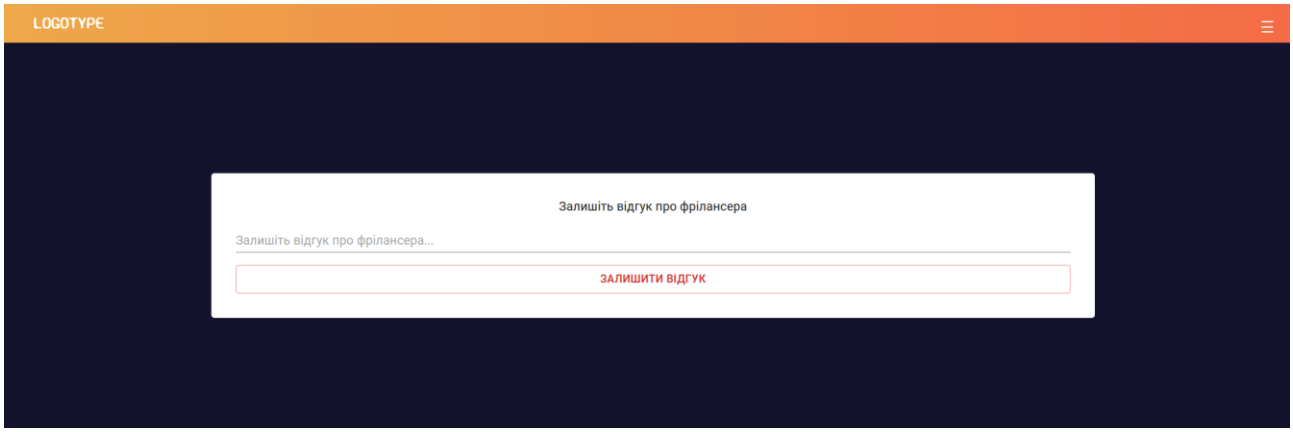

Рисунок 3.30 – Сторінка для відгуку

На сторінці напису рецензії за бажанням замовник може написати кілька слів про фрілансера (рисунок 3.31).

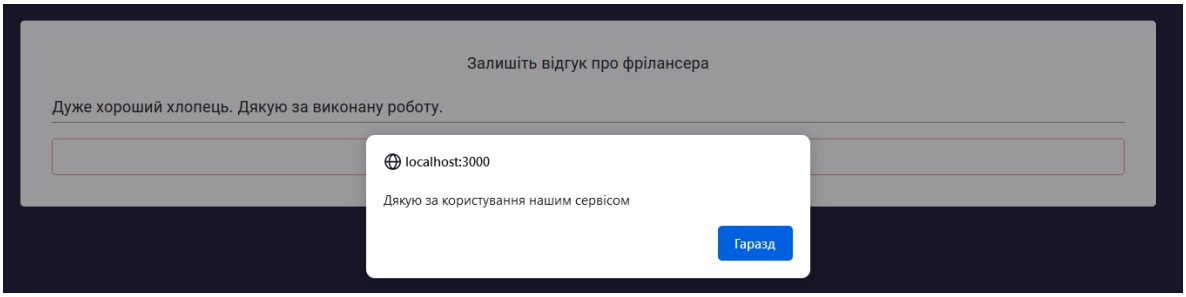

Рисунок 3.32 – Форма для відгуку

Після написання рецензії та здачі роботи, грошова винагорода начисляється фрілансеру.

Після чого фрілансер може переглянути свої відгуки та рахунок в своєму профілі (рисунок 3.33).

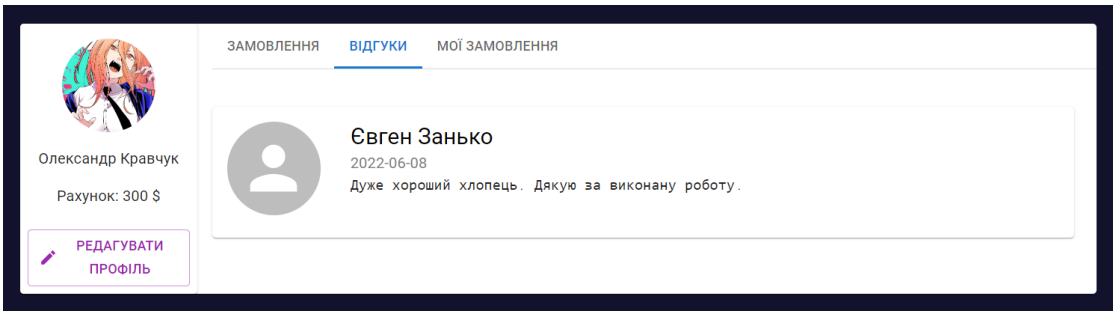

Рисунок 3.33 – Перегляд відгуків

Після реєстрації аккаунта користувач записується в базу даних (рисунок 3.34).

Рисунок 3.34 – Користувач в базі даних

Після створення завдання та внесення його в базу даних MongoDB, наш документ «Orders» виглядає ось так (рисунок 3.35).

Рисунок 3.35 – Замовлення в базі даних

Отже, тестування завершене, веб-додаток працює справно без помилок. Усі перевірки завершились успішно.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Аналіз ринку

Основним завданням роботи було створення веб-додатку для керування завданнями для фрілансу. Створений за допомогою сучасних технологій розробки. Цей сервіс не є єдиним на ринку, він лише повторює основний функціонал таких гігантів як: Upwork.com, Freelancer.com, Freelance.ua. Сервіс є доступним на всіх можливих браузерах та систем.

Веб-сервіс – це повноцінний веб-ресурс з клієнтською та серверними частинами та з базами даних з інтерфейсом і відповідним функціоналом до його тематики.

Потенційними користувачами веб-сервісу є фрілансери, які бажаються здобути необхідний досвід роботи та отримати перші гроші за їхню працю. Також, додатком можуть користуватись замовники для розміщення власних замовлень з метою їх виконання.

Реалізований проєкт надає користувачам необхідний функціонал, який відповідає тематиці сайту, для взаємодії з сервісом:

- реєстрація;
- авторизація;
- створення замовлень;
- відгуки на замовлення;
- написання відгуків та коментарів для фрілансерів.

Кожен створений продукт, будь то фізичний або нефізичний, потребує необхідного обслуговування. Цей проєкт не є вийнятком. З сервісом без відповідного нагляду, може статись багато неприємного, що може порушити його роботу, це неправильна робота певного функціоналу, різні помилки, які можуть виникнути в процесі користування. За цим необхідно слідкувати та швидко виправляти, щоб не втратити потенційних користувачів.

4.2 Розрахунок витрат на проєктування

В Україні розмір мінімальної заробітної плати з 1 грудня 2022 року стоновить 6500 гривень в місяць, тобто розмір не може бути меншим за повний місяць роботи.

В даному проєкті було задіяно три працівника: web-програміст, дизайнер та тестувальник.

Web-програміст в даному сервісі займався розробкою клієнтської та серверною частинами. Його оклад в місяць становить 7200 гривень

Дизайнер в даному сервісі займався макетом та дизайном додатку. Його оклад в місяць становить 7100 гривень в місяць.

Тестувальник в даному сервісі займався тестуванням всього функціоналу та повідомляв головному розробнику про виникнення проблем, для їх вирішення. Його оклад становить 7000 гривень в місяць.

Заробітна плата працівників показана в таблиці 4.1.

Таблиця 4.1 – заробітна плата працівників

№	Посада	Оклад грн./міс.	Відрахування, грн./міс.	Кількість		Сума, грн.
1	Web-програміст	10233	1995,39	1 чол.	3 міс.	24712
2	Дизайнер	9921	1934,59	1 чол.	3 міс.	23959
3	Тестувальник	9283	1810,18	1 чол.	3 міс.	22418
Загальна заробітна платня						71089

Отже, загальна заробітня платня для усіх працівників становить 71089 гривень за три місяці роботи.

4.3 Обґрунтування необхідності розробки

На даний момент IT сфера розвивається з колосальною швидкістю. З’являються багато нових професій в цій сфері. Також, з’являються багато охочих попробувати себе в ній, та здобути власний куток в великому просторі інформаційних технологій. Але дуже часто, щоб здобути першу роботу,

необхідно мати вже якийсь досвід. Для цього ідеально підходять фріланс-біржі, які дозволяють початківцям попробувати свої можливості та отримати перші завдання, досвід та проекти які доповнять їх портфоліо, який необхідний для майбутнього розвитку в сфері.

При правильному керуванні та при відповідному розголошенні, яке необхідне для просунення сервісу, створення такого сервісу зможе приносити користь людям, тим самим приносити прибуток власникам додатку та збільшення охопленої аудиторії.

Отже, при правильному керуванні, такий сервіс має право на існування та найголовніше приносити прибуток, окрім того бути корисним для потенційних користувачів.

ВИСНОВКИ

Під час дипломного проектування було спроектовано та реалізовано повноцінний веб-додаток з клієнтською та серверною частиною з використанням найсучасніших інструментів для веб-розробки. З «нуля» спроектований власний дизайн, який є ергономічним та зручним у використанні.

Протягом розробки клієнтської частини було використано найсучасніший інструмент - React JS, що дозволило здобути досвід та вміння розробки «front end». Для реалізації серверної частини було використано Node JS , що дозволило набути досвіду в написанні «back end», завдяки якому було набуто нових навичок та знань.

В результаті роботи був реалізований та протестований повноцінний веб-сервіс на базі цих інструментів розробки, який задовільняє усі поставлені завдання. Цей веб-додаток є хорошим доповненням до портфоліо розробника, який допоможе в майбутньому в розвитку в сфері веб-розробки.

В подальшому планується розширення функціоналу, впровадження нових сучасних технологій розробки. Також спланована впровадження банківської системи для веб-додатку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хто такі Frontend і Backend розробники: ItStep: веб-сайт URL: https://te.itstep.org/blog/frontend-and-backend-developers (дата звернення: 16.04.2022).

2. Основи веб-розробки. Основи веб-розробки: веб-сайт URL: https://sites.google.com/view/inf10/дистанційне-навчання/brainbasket-foundation/основи-веб-розробки (дата звернення: 16.04.2022).

3. Документація ReactJS. ReactJS: веб-сайт. URL: https://reactjs.org/docs/getting-started.html (дата звернення 18.04.2022).

4. Документація NodeJS. Node JS: веб-сайт. URL: https://nodejs.org/en/docs/ (дата звернення: 22.04.2022).

5. Сервіс MongoDB. MongoDB: веб-сайт URL: https://www.mongodb.com/ (дата звернення: 26.04.2022).

6. Сучасний посібник JavaScript. JavaScript: веб-сайт URL: https://uk.javascript.info/ (дата звернення 15.04.2022).

7. CSS фреймворк Material UI. Material UI: веб-сайт URL: https://mui.com/ (дата звернення 21.04.2022).

8. Довідник по HTML: w3Schools: веб-сайт URL: https://www.w3schools.com/html/ (дата звернення 22.04.2022).

9. Довідник по CSS: w3schools: веб-сайт URL: https://www.w3schools.com/css/ (дата звернення 22.04.2022).

ДОДАТКИ

Додаток А

Лістинг програмного коду

Файл index.js:

```
import express from 'express';
import bodyParser from 'body-parser';
import mongoose from 'mongoose';
import cors from 'cors';
import dotenv from 'dotenv';

import userRoutes from './Routes/user.js';
import orderRoutes from './Routes/orders.js';

dotenv.config()
const app = express();

app.use(bodyParser.json({limit: '30mb', extended: true}));
app.use(bodyParser.urlencoded({limit: '30mb', extended: true}));
app.use(cors());

app.use('/user', userRoutes)
app.use('/orders', orderRoutes)

const CONNECTION_URL = "mongodb+srv://admin:admin@diploma-
cluster.hvqf8.mongodb.net/myFirstDatabase?retryWrites=true&w=majorit
y";
const PORT = process.env.PORT || 5000;

mongoose.connect(CONNECTION_URL)
    .then(() => app.listen(PORT, (console.log(`Server running on
PORT: ${PORT}`))))
    .catch((error) => console.log(`${error} did not connect`))
```

Файл UserController.jsx:

```
import bcrypt from 'bcryptjs';
import jwt from 'jsonwebtoken';
import User from '../Models/user.js';
import mongoose from "mongoose";

export const secretKey = 'secretKey';
```

```
export const signin = async (req, res) => {
    const {email, password} = req.body;

    try {
        const existingUser = await User.findOne({ email });
        if (!existingUser) return res.status(404).json({message:
'Користувач з такою електронною скринькою не існує.'});

        const isPasswordCorrect = await bcrypt.compare(password,
existingUser.password);

        if(!isPasswordCorrect) return res.status(400).json({message:
'Неправильний пароль'});

        const token = jwt.sign({ email: existingUser.email, id:
existingUser._id }, secretKey, {expiresIn: '1h'});

        res.status(200).json({ result: existingUser, token });
    } catch (e) {
        res.status(500).json({ message: 'Щось пішло не так' })
    }
}

export const signup = async (req, res) => {
    const { email, password, confirmPassword, firstName, lastName }
= req.body;

    try {
        const existingUser = await User.findOne({ email });

        if(existingUser) return res.status(400).json({message:
'Користувач з такою електронною скринькою вже існує.'});

        if(password !== confirmPassword) return
res.status(400).json({message: 'Паролі не співпадають.'});

        const hashedPassword = await bcrypt.hash(password, 12);

        const result = await User.create({ email, password:
hashedPassword, name: `${firstName} ${lastName}`, selectedFile: ''
});

        const token = jwt.sign({ email: result.email, id: result._id
}, secretKey, {expiresIn: '1h'});

        res.status(200).json({ result, token, message: 'Ви успішно
зареєструвались' });
    } catch (e) {
        res.status(500).json({ message: 'Щось пішло не так' });
```

```
    }
}

export const getuser = async (req, res) => {
    try {
        const user = await User.findOne({ name: req.params.username
})
        res.status(200).json({ user })
    } catch (e) {
        res.status(500).json({ message: 'Користувач не існує' })
    }
}

export const updateAvatar = async (req, res) => {
    const {id} = req.params;
    const {selectedFile} = req.body

    if (!mongoose.Types.ObjectId.isValid(id)) return
res.status(404).send(`Немає користувача з таким id: ${id}`);

    const user = await User.findById(id);

    await User.findByIdAndUpdate(id, {selectedFile: selectedFile});

    res.json(user)
}

export const updateInformation = async (req, res) => {
    const id = req.userId;
    const {telegram} = req.body;
    if (!mongoose.Types.ObjectId.isValid(id)) return
res.status(404).send(`No user with id: ${id}`);
    try {
        await User.findByIdAndUpdate(id, {telegram: telegram})
        res.json({message: 'Обновіть сторінку'})
    } catch (e) {
        res.status(201).json({message: 'Щось пішло не так'})
    }
}
```

Файл OrderController.js:

```
import express from 'express';
import mongoose from 'mongoose';

import Order from '../Models/order.js'
import User from "../Models/user.js";

export const getOrders = async (req, res) => {
    const orders = await Order.find()
```

```
    res.status(201).json(orders)
}

export const getOrder = async (req, res) => {
    const {id} = req.params;
    const order = await Order.findById(id)
    const creator = await User.findById(order.creator)
    res.status(201).json({order, creator })
}

export const addOffer = async (req, res) => {
    const {id} = req.params;
    const {offer} = req.body;
    const order = await Order.findById(id)
    const user = await User.findById(req.userId)
    try {
        await order.addToComments(req.userId, user.name,
user.selectedFile, offer)
        res.status(201).json({message: 'Дякую, тепер чекайте
відповіді'})
    } catch (e) {
        res.status(409).json({ message: e.message });
    }
}

export const createOrder = async (req, res) => {
    const {formData} = req.body
    const newOrder = new Order({...formData, creator: req.userId})

    try {
        await newOrder.save();
        res.status(201).json(newOrder)
    } catch (e) {
        res.status(409).json({ message: e.message });
    }
}

export const confirmOffer = async (req, res) => {
    const {id} = req.params
    const {userId} = req.body
    const user = await User.findById(userId)
    const order = await Order.findByIdAndUpdate(id, {status:
'inProccess', freelancer: user.name, freelancerId: userId})

    try {
        await user.addToOrders(id, order.creator, order.startDate,
order.orderName, id)
        res.status(201).json({message: 'Дякую, тепер чекайте на свою
роботу'})
    } catch (e) {
```

```
        res.status(409).json({ message: e.message });
    }

}

export const getUserOrders = async (req, res) => {
    const {id} = req.params
    const order = await Order.find({creator: id})
    res.json(order)
}

export const confirmFromUser = async (req, res) => {
    const {id} = req.params;
    const {gitLink} = req.body
    const order = await Order.findByIdAndUpdate(id, {status:
'PendingForCustomer', gitLink: gitLink})
    res.status(201).json({message: 'Дякую, тепер зачекайте
відповіді'})
}

export const cancelOffer = async (req, res) => {
    const {id} = req.params;
    const {userName} = req.body;
    const order = await Order.findByIdAndUpdate(id, {status:
'Pending', freelancer: '', comments: [], gitLink: '', freelancerId:
''});
    const user = await User.findOne({name: userName});

    try {
        await user.deleteOrder(id)
        res.status(201).json({message: 'Дякую, тепер зачекайте',
orderId: id})
    } catch (e) {
        res.status(409).json({ message: e.message });
    }
}

export const submitTask = async (req, res) => {
    const {id} = req.params;
    const {userName} = req.body;
    try {
        const order = await Order.findByIdAndUpdate(id, {status:
'Completed', comments: []})
        const user = await User.findOne({name: userName})
        user.money+= order.orderPrice
        await user.save()
        res.status(201).json({message: 'Дякую за користування нашим
сервісом', freelancerId: order.freelancerId})
    } catch (e) {
        console.log(e)
```

```
    }
}

export const addReview = async (req,res) => {
    const {id} = req.params;
    const {reviewDescr, reviewDate} = req.body;
    const customer = await User.findById(req.userId);
    const user = await User.findById(id);
    try {
        await user.addToReviews(customer._id, customer.selectedFile,
customer.name, reviewDescr, reviewDate)
        res.status(201).json({message: 'Дякую за користування нашим
сервісом'})
    } catch (e) {
        res.status(409).json({ message: e.message });
    }
}
```

Файл UserRoute.js:

```
import express from "express";
import {signin, signup, getuser, updateAvatar, updateInformation}
from "../Controllers/user.js";
import auth from "../Middleware/auth.js";

const router = express.Router()

router.post('/signin', signin);
router.post('/signup', signup);


router.get('/getuser/:username', getuser);
router.patch('/:id', updateAvatar);
router.post('/updateinformation', auth, updateInformation)

export default router;
```

Файл OrdersRoute.js:

```
import express from "express";

const router = express.Router();

import auth from "../Middleware/auth.js";
import {
    createOrder,
    getOrders,
    getOrder,
    addOffer,
    confirmOffer,
```

```
    getUserOrders,
    confirmFromUser,
    cancelOffer, submitTask,
    addReview
} from "../Controllers/order.js";

router.post('/', auth, createOrder);
router.post('/addoffer/:id', auth, addOffer);
router.get('/getorders', getOrders);
router.get('/getorder/:id', getOrder);
router.post('/confirmoffer/:id', confirmOffer);
router.get('/getuserorders/:id', getUserOrders)
router.put('/updateoffer/:id', confirmFromUser);
router.put('/canceloffer/:id', cancelOffer);
router.put('/submitoffer/:id', submitTask);
router.put('/addreviews/:id', auth, addReview)

export default router;
```

Файл API.js:

```
import axios from "axios";

const API = axios.create({baseURL: 'https://mern-
diploma.herokuapp.com'});

API.interceptors.request.use((req) => {
    if (localStorage.getItem('profile')) {
        req.headers.Authorization = `Bearer
${JSON.parse(localStorage.getItem('profile')).token}`;
    }

    return req;
});

export const signIn = formData => API.post('/user/signin',
formData);
export const signUp = formData => API.post('/user/signup',
formData);

export const getUserProfile = username =>
API.get(`/user/getuser/${username}`);
export const updateUserAvatar = ( id, img ) =>
API.patch(`/user/${id}`, img);
export const updateUserInformation = (telegram) =>
API.post('/user/updateinformation', {telegram});

export const createOrder = formData => API.post('/orders',
{formData});
export const getOrders = () => API.get('/orders/getorders');
```

```
export const getOneOrder = id => API.get(`/orders/getorder/${id}`);
export const addOneOffer = (id, offer) =>
API.post(`/orders/addoffer/${id}`, {offer});
export const confirmOffer = (id, userId) =>
API.post(`/orders/confirmoffer/${id}`, {userId})
export const getUsersOrders = (id) =>
API.get(`/orders/getuserorders/${id}`);
export const updateOffer = (id, gitLink) =>
API.put(`/orders/updateoffer/${id}`, {gitLink});
export const cancelOffer = (id, userName) =>
API.put(`/orders/cancelOffer/${id}`, {userName});
export const submitTask = (id, userName) =>
API.put(`/orders/submitoffer/${id}`, {userName});
export const addNewReview = (id, reviewDescr, reviewDate) =>
API.put(`/orders/addreviews/${id}`, {reviewDescr, reviewDate})
```