**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ**
**Кафедра програмної інженерії**

# КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ РЕКРУТИНГОВОЇ ПЛАТФОРМИ ДЛЯ ІТ КОМПАНІЙ ІЗ ВИКОРИСТАННЯМ .NET FRAMEWORK»**

Виконав: студент 2 курсу, групи 8.1213-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
О.О. Безбородько
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Мильцев О.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент професор кафедри комп'ютерних наук, професор, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

**ЗАТВЕРДЖУЮ**
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

# З А В Д А Н Н Я

## НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Безбородьку Олексію Олександровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка серверної частини рекрутингової платформи для IT компаній із використанням .NET Framework

керівник роботи Мильцев Олександр Михайлович, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 385-с

2. Строк подання студентом роботи 25.11.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Проєктування та реалізація бекенд-частини рекрутингової платформи.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 07.03.2024 р.

## К А Л Е Н Д А Р Н И Й  П Л А Н

| № | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Розробка плану роботи. | 01.04.2024 | |
| | | | |
| 2. | Збір вихідних даних. | 30.04.2024 | |
| | | | |
| 3. | Обробка методичних та теоретичних | 05.06.2024 | |
| | джерел. | | |
| | | | |
| 4. | Розробка першого та другого розділу. | 02.09.2024 | |
| | | | |
| 5. | Розробка третього розділу. | 04.11.2024 | |
| | | | |
| 6. | Оформлення та нормоконтроль | 20.11.2024 | |
| | кваліфікаційної роботи магістра. | | |
| | | | |
| 7. | Захист кваліфікаційної роботи. | 11.12.2024 | |

Студент _____ О.О. Безбородько
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.М. Мильцев
(підпис) (ініціали та прізвище)

## Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

# РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка серверної частини рекрутингової платформи для IT компаній із використанням .NET Framework»: 62 с., 25 рис., 14 джерел.

БАЗИ ДАНИХ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РЕКРУТИНГ, ASP.NET CORE, CRM-СИСТЕМА, ENTITY FRAMEWORK, MS SQL SERVER, .NET, REST API.

Об’єкт дослідження – технології розробки CRM-систем рекрутингу для IT-компаній.

Предмет дослідження – процес розробки та впровадження CRM-системи для рекрутингу для IT-компаній, яка базується на мікросервісній архітектурі та використовує технології ASP.NET Core.

Мета роботи – розробка CRM-системи для медичного закладу, яка забезпечить ефективне управління даними пацієнтів, медичними послугами, персоналом та іншими аспектами функціонування медичного закладу, з використанням сучасних технологій мікросервісної архітектури на платформі ASP.NET Core.

Метод дослідження – моделювання, проєктування, програмний, аналітичний.

На сьогоднішній день проблема підбору та відбору фахівців на підприємствах стає все більш актуальною. Керівники відповідально підходять до питань формування трудового колективу, тому що від цього багато в чому залежить успішність діяльності компанії. Підбір необхідних фахівців є невід’ємною складовою успіху будь-якої компанії. Тому, зважаючи на актуальність цього питання та тенденцію до комп’ютеризації у сучасному світі, корисним рішенням є створення застосунку, який би допомагав його вирішити.

У роботі детально розглянуто вибір технологій, процеси інтеграції та забезпечення безпеки даних, а також проведено тестування та оцінку.

# SUMMARY

Master's qualifying paper "Developing a Backend Part of a Recruiting Platform for IT Companies Using .NET Framework": 62 pages, 25 figures, 14 references.

DATABASES, SOFTWARE, RECRUITMENT, ASP.NET CORE, CRM-SYSTEM, ENTITY FRAMEWORK, MS SQL SERVER, .NET, REST API.

The object of research is the development of CRM recruiting systems for IT companies.

The subject of the study is the process of developing and implementing a CRM system for recruiting for IT companies, which is based on microservice architecture and uses ASP.NET Core technologies.

The goal of the work is to develop a CRM system for a medical institution that will provide effective management of patient data, medical services, personnel and other aspects of the operation of a medical institution, using modern microservice architecture technologies on the ASP.NET Core platform.

The research method is modeling, design, software, analytical.

Today, the problem of recruitment and selection of specialists at enterprises is becoming more and more urgent. Managers approach the issue of workforce formation responsibly, because the success of the company largely depends on it. The selection of the necessary specialists is an integral part of the success of any company. Therefore, considering the relevance of this issue and the trend towards computerization in the modern world, a useful solution is to create an application that would help to solve it. In the work, the choice of technologies, processes of integration and ensuring data security, as well as testing and evaluation were carried out in detail.

**ЗМІСТ**

Завдання на кваліфікаційну роботу……2
Реферат ..... 4
Summary ..... 5
Вступ..... 8
1 Технічне завдання ..... 10
1.1 Терміни та визначення..... 10
1.2 Функціональні вимоги..... 11
1.3 Нефункціональні вимоги..... 12
1.4 Опис предметної області ..... 12
1.5 Опис системи..... 13
1.6 Огляд інструментів розробки..... 14
1.6.1 Структура front-end проєкту ..... 15
1.6.2 Структура back-end проєкту ..... 16
1.6.3 Аналіз конкурентів ..... 17
2 Проєктування..... 19
2.1 Використання UML під час розробки системи..... 19
2.1.1 Діаграма прецедентів..... 20
2.1.2 Діаграма класів..... 24
2.1.3 Діаграма компонентів..... 26
2.1.4 Діаграма розміщення..... 28
2.2 Архітектура серверної частини ..... 30
2.2.1 Серверні статуси ..... 32
3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ..... 33
3.1 Опис інструментів для розробки клієнтської частини..... 33
3.1.1 Мова програмування TypeScript..... 33
3.1.2 Vite..... 33
3.1.3 React..... 34

3.2 Опис інструментів для розробки серверної частини ..... 34
3.3 Процес реалізації серверної частини ..... 39
3.3.1 Процес створення Entity Framework Code First-моделі ..... 39
3.3.2 Процес створення ASP.NET WebAPI-контролера ..... 40
3.4 Тестування системи (backend, frontend) ..... 41
3.5 Unit тести ..... 42
3.6 Керівництво користувача ..... 42
3.6.1 Рівень підготовки користувача ..... 42
3.6.2 Підготовка до роботи ..... 43
3.6.3 Авторизація, вхід в систему ..... 43
3.6.4 Реєстрація нового акаунту ..... 44
3.6.5 Особистий кабінет ..... 45
3.6.6 Пошук вакансій ..... 48
3.6.7 Перегляд вакансії ..... 50
3.6.8 Створення вакансії ..... 52
3.6.9 Перегляд кандидатів ..... 55
3.6.10 Перегляд анкети кандидата ..... 57
3.6.11 Перегляд компаній ..... 58
Висновки ..... 60
Перелік посилань ..... 61

# ВСТУП

Ринок праці в IT-сфері динамічно розвивається, і попит на кваліфікованих фахівців значно перевищує пропозицію. Ця робота описує розробку програми, яка допоможе HR-агенціям ефективніше шукати та підбирати IT-фахівців для замовників.

Основний функціонал системи:

– пошук кандидатів за ключовими словами, навичками, досвідом роботи та іншими параметрами;
– створення та проведення тестів для оцінки знань та навичок кандидатів;
– ведення бази даних резюме кандидатів;
– можливість спілкування з кандидатами через чат.

Актуальність дослідження: пошук та підбір IT-фахівців – це складний процес, який потребує багато часу та зусиль. Традиційні методи, такі як розміщення оголошень на сайтах з пошуку роботи, не завжди дають бажаний результат.

З огляду на це, можна виділити наступні цілі і задачі нашого дослідження.

Мета: розробити систему рекрутингу IT фахівців.

Задачі:

– провести аналіз існуючих програм для пошуку IT-фахівців;
– розробити концепцію та специфікацію програми;
– створити прототип програми;
– провести тестування прототипу програми;
– оцінити ефективність програми.

Методи дослідження:

– аналіз літературних джерел;
– анкетування;
– інтерв’ю;
– проєктування програмного забезпечення;
– тестування програмного забезпечення.

Об’єктом дослідження є процес рекрутингу та відбору ІТ-фахівців у сучасній індустрії інформаційних технологій.

Предметом дослідження є розробка та впровадження системи рекрутингу ІТ-фахівців для оптимізації процесів відбору та прийому на роботу спеціалістів в галузі інформаційних технологій.

Дослідження базується на аналізі літературних джерел, а також на використанні методів аналізу, проєктування, програмування та тестування програмного забезпечення.

# 1 ТЕХНІЧНЕ ЗАВДАННЯ

## 1.1 Терміни та визначення

Бекенд: елемент системи, є частиною вебдодатка, яка відповідає за обробку логіки, збереження даних та забезпечення комунікації з фронтендом.

Фронтенд: це та частина вебдодатка, яка відповідає за відображення інтерфейсу користувача та взаємодію з користувачем.

YML-файл: формат файлу, який використовується для опису даних і їх характеристик у структурованому вигляді.

API: інтерфейс програмування додатків – набір готових класів, процедур, функцій або структур, які використовуються для взаємодії між програмами.

Кросплатформовість: це властивість програмного забезпечення або додатків, яка дозволяє їм працювати на різних платформах або операційних системах з мінімальними змінами в коді або логіці. Це означає, що можна розробити одну версію додатку і використовувати її на різних платформах, таких як iOS, Android, веббраузери та інші, замість написання окремого коду для кожної платформи.

Нативний: це термін, що використовується в програмному забезпеченні для опису компонентів або додатків, розроблених для конкретної платформи або операційної системи. Коли говорять про нативний код, це означає, що код написаний з використанням мови програмування та інструментів, які є стандартними для даної платформи.

Фреймворк: це набір компонентів, бібліотек, інструментів та правил, що надають основу для розробки програмного забезпечення або додатків. Він визначає структуру, організацію та спосіб взаємодії між компонентами програми, сприяючи швидкій розробці та підтримці програмного забезпечення.

Перевикористання: це принцип або практика в програмному забезпеченні, коли певні компоненти, модулі, функції або інші елементи коду можуть бути використані повторно в різних частинах програми або навіть в різних програмах.

HR-агенство: організація, яка спеціалізується на наданні послуг у сфері управління людськими ресурсами, зокрема, на пошуку, відборі та розміщенні робочих місць для кандидатів на роботу.

Кандидат: особа, яка подає заяву на робоче місце та проходить відповідні процедури відбору для здобуття цієї посади.

Сеньйор: досвідчений фахівець у сфері розробки програмного забезпечення, який має високий рівень знань та навичок у мові програмування.

Вимоги до програми: сукупність умов, функціональних властивостей та характеристик, які має відповідати програма для виконання певного завдання.

Квіз: короткий тест, який спрямований на оцінку знань кандидата з основних мов програмування, щоб визначити його рівень компетентності.

Тімлід: фахівець, який відповідає за керування розробкою програмного забезпечення та технічне керівництво в проєкті.

Кабінет кандидата: індивідуальний обліковий запис кожного кандидата в системі, де вони можуть переглядати та редагувати свої особисті дані, завантажувати резюме, взаємодіяти з HR-агентством та отримувати оновлення щодо статусу своїх заявок.

Месенджери: програмні засоби комунікації, такі як чати та платформи для обміну повідомленнями.

Програмний модуль: окрема частина програмного забезпечення, яка виконує певну функцію або завдання, і може бути використана самостійно або разом з іншими модулями для досягнення більшої функціональності.

JSON: легкий формат обміну даними, який легко читається людьми і легко аналізується машинами.

## 1.2 Функціональні вимоги

Створення заявки з необхідними вимогами щодо кандидата.

Інтеграція з ресурсами для пошуку роботи для спілкування та взаємодії з кандидатами.

Зберігання баз даних кандидатів та створення їх індивідуальних кабінетів.

Прив’язка до месенджерів та електронної пошти для зручного спілкування.

Проведення міні тесту на знання основних мов програмування для кандидатів.

Відсіювання кандидатів, які не відповідають вимогам, щоб економити час тимлідів та керівників проєктів.

### 1.3 Нефункціональні вимоги

Мова програмування: розробка системи повинна здійснюватися з використанням мови програмування C# та TypeScript.

Розробка бекенду на платформі .NET: система повинна бути розроблена з використанням платформи .NET.

Використання Visual Studio: розробка повинна проводитися в середовищі Visual Studio.

Розробка фронтенду: система повинна бути розроблена з використанням TypeScript.

Використання ASP .NET Core: для розробки вебзастосунку слід використовувати фреймворк ASP .NET Core.

Використання Visual Studio Code: розробка повинна проводитися в середовищі Visual Studio Code.

Збереження даних: система повинна використовувати базу даних для збереження інформації про вакансії, кандидатів та інше.

### 1.4 Опис предметної області

Програма, яка розробляється, призначена для автоматизації процесу рекрутингу та відбору кандидатів для вакансій в IT компаніях через посередництво HR агенств. Основні функції програми включають в себе

створення та обробку заявок на робочі місця, взаємодію з кандидатами, зберігання інформації про кандидатів та допомогу у проведенні тестів для відбору найкращих претендентів.

Програма буде здатна ефективно керувати всіма аспектами процесу відбору персоналу, починаючи від створення заявок на робочі місця та закінчуючи проведенням тестів для відбору найкращих кандидатів. Вона буде забезпечувати зручний та швидкий доступ до інформації про кандидатів, а також спрощувати процес спілкування з ними через використання месенджерів та електронної пошти.

Однією з головних переваг програми буде можливість зберігання всієї необхідної інформації про кандидатів у централізованій базі даних, що дозволить легко і швидко відстежувати їхній прогрес у процесі відбору. Крім того, програма буде забезпечувати можливість автоматичної фільтрації кандидатів за різними критеріями, що значно спростить процес відбору та скоротить час, необхідний для знаходження найбільш підходящих претендентів на вакансії.

## 1.5 Опис системи

Програма призначена для автоматизації процесу рекрутингу та відбору кандидатів для вакансій в IT компаніях через посередництво HR агенств. Розглянемо основні функції програми.

*Створення та обробка заявок на робочі місця*: програма забезпечує можливість створення та обробки заявок на робочі місця, що дозволяє HR агентствам ефективно управляти потоком кандидатів.

*Взаємодія з кандидатами*: програма надає зручний інтерфейс для взаємодії з кандидатами, включаючи спілкування через різні канали зв’язку, такі як електронна пошта та месенджери.

*Зберігання інформації про кандидатів*: програма автоматично зберігає та організовує інформацію про кандидатів, що допомагає забезпечити доступність та легкість управління даними.

*Допомога у проведенні тестів для відбору найкращих претендентів*: програма надає можливість проведення тестів та оцінки кандидатів, що допомагає відібрати найбільш підходящих кандидатів для вакансій.

Ці функції спрямовані на оптимізацію процесу відбору та забезпечення високої ефективності рекрутингу через HR агентства в IT секторі.

Серверна частина системи розроблена з використанням .NET, ASP .NET Core та MSSQL DB і відповідає за обробку та керування даними. Вона складається з низки сервісів, які забезпечують основні функції системи.

Також, серверна частина включає REST API, яке забезпечує взаємодію між сервером та клієнтською частиною додатка. API надає набір ендпоінтів, які дозволяють клієнтському додатку виконувати різноманітні операції, такі як отримання, оновлення та видалення даних.

Клієнтська частина вебдодатка розроблена з використанням React і представляє собою інтерфейс, через який кандидати або рекрутери взаємодіють з системою. Вона містить набір сторінок, які надають доступ до функціональності системи, такі як перегляд та редагування свого профілю, пошук вакансій або робітників (в залежності від типу акаунту), а також переписку з кандидатом або компанією для узгодження умов роботи на тій чи іншій посаді.

## 1.6 Огляд інструментів розробки

Для розробки системи використовуються наступні інструменти та технології, наведемо їх далі.

Мова програмування: C# – мова програмування, що надає широкі можливості для розробки веб- та десктоп-додатків на платформі .NET [10].

ReactJS (скорочено від React.js) – це бібліотека JavaScript, яку використовують для створення інтерактивних інтерфейсів користувача. Вона допомагає розробникам створювати динамічні та масштабовані вебзастосунки та вебсайти.

TypeScript – це надбудова над JavaScript, розроблена Microsoft, яка додає статичну типізацію та додаткові об’єктно-орієнтовані можливості до мови. TypeScript спрощує написання більш надійного та безпечного коду, полегшує його налагодження та підтримку.

Платформа: .NET – платформа розробки програмного забезпечення, яка надає ряд інструментів та бібліотек для створення програм для Windows, Linux та інших операційних систем.

Розробка вебзастосунків: ASP .NET Core – фреймворк для розробки вебдодатків на мові C#, який надає високу продуктивність, переносимість та масштабованість.

Розробка середовища: Visual Studio – інтегроване середовище розробки (IDE) для платформи .NET, яке надає зручні інструменти для створення, налагодження та відлагодження програмного забезпечення.

Збереження даних: для збереження інформації про вакансії, кандидатів та інше використовується база даних, яка підтримується платформою .NET.

Для забезпечення взаємодії між клієнтською (візуальною) частиною додатка та сервером має використовуватися REST API. REST API дозволяє клієнтському додатку відправляти запити на сервер та отримувати від нього відповіді, що включають необхідні дані або статус операції. Це дає можливість реалізувати високий рівень абстракції, забезпечуючи гнучкість та масштабованість системи.

### 1.6.1 Структура front-end проєкту

Структура front-end проєкту складається з декількох основних папок:

- **index.ts** або **index.js/App.ts**: головний файл, в якому відбувається ініціалізація додатка та рендеринг головного компонента;
- **src/**: директорія, де зберігаються власні компоненти, стилі, роутери та інші файли додатка;

- **node_modules/**: директорія, в якій зберігаються залежності проєкту, установлені за допомогою npm або yarn;
- **package.json**: файл, що містить інформацію про ваш проєкт, включаючи залежності та налаштування скриптів.

### 1.6.2 Структура back-end проєкту

*API*. Проєкт API відповідає за надання точок доступу та обробку вхідних запитів від клієнтів. Він містить контролери, які визначають маршрути та дії для обробки різних HTTP-запитів. Проєкт API взаємодіє з проєктами BLL та DAL для обробки запитів, отримання або оновлення даних та повернення відповідних відповідей клієнтам.

*BLL*. Проєкт BLL містить бізнес-логіку додатку. Він укриває правила та операції, що регулюють поведінку додатку. Цей рівень відповідає за реалізацію бізнес-логіки, виконання перевірок, виконання складних операцій та координацію взаємодії між проєктами API та DAL. Він включає служби, менеджери або інші компоненти, які обробляють бізнес-операції.

*CORE*. Проєкт CORE є загальним проєктом, який містить загальний код та утиліти, які використовуються в усьому рішенні. Він служить центральним місцем для визначення основних функціональностей, загальних компонентів та засобів, які використовуються в проєктах API, BLL та DAL.

*DAL*. Проєкт DAL відповідає за доступ до даних та взаємодію з джерелами даних, такими як база даних або зовнішні сервіси. Він містить класи та компоненти, які забезпечують операції збереження, отримання, оновлення та видалення даних. DAL виконує завдання доступу до даних, а BLL використовує ці операції для реалізації бізнес-логіки.

*SharedModels*. Проєкт SharedModels містить моделі даних або класи, які використовуються як загальний код між різними проєктами в рішенні. Цей проєкт дозволяє спільно використовувати моделі даних та інші класи, що є необхідними для взаємодії між проєктами API, BLL та DAL.

### 1.6.3 Аналіз конкурентів

У кожного IT продукта є своя ніша та свої конкурентні продукти. Нижче ми порівняємо створену з систему з іншими гравцями на ринку.

Розроблена рекрутингова система Marina.io має переваги, розглянемо їх далі.

*Проведення міні-тестів на знання основних мов програмування та технології*. Дуже гнучко налаштовувані тести, які може запустити рекрутер для кандидата згідно скілів, які необхідні в вакансії. Все відбувається в рамках одного чату і дуже гнучко інтегровано в екосистему продукту, що дає додаткову зручність як для рекрутера, так і для кандидата.

*Сучасний та простий інтерфейс*. Сучасний інтерфейс продукту дозволяє зручно орієнтуватись користувачам в системі та дозволяє швидко навчитись користуватись системою. Використовуючи сучасні дизайн-підходи, ми отримуємо зручний продукт, який завдяки своїй простоті та відсутності «перевантаженого» інтерфейсу може бути гарним варіантом для невеликих та середніх рекрутингових IT агенцій.

*Сучасні технології*. Використовуючи останні версії фронтенд та бекенд бібліотек та фреймовркв, ми отримуємо швидкий та безпечний продукт, який завдяки цьому стає зручніше використовувати.

Продукт-конкурент **HURMA**.

Переваги:

– комплексне рішення для HR;
– широкий спектр функцій;
– підходить для великих компаній.

Недоліки:

– дорожче, ніж інші системи;
– може бути складним для налаштування та використання.

Продукт-конкурент **Workable**.

Переваги:

– штучний інтелект для автоматизації процесів;

– гнучкі налаштування;
– підходить для компаній будь-якого розміру.

Недоліки:

– складний для налаштування;
– дорогий щомісячний тарифний план.

# 2 ПРОЄКТУВАННЯ

## 2.1 Використання UML під час розробки системи

Для розробки системи використовуються різні методології. Одним з найпоширеніших мов моделювання є уніфікована мова моделювання UML (Unified Modeling Language), призначена для моделювання, уявлення, проєктування та документування програмних систем, організаційно-економічних систем, технічних систем та інших систем різної природи. UML дозволяє розробникам програмного забезпечення досягти згоди щодо графічних позначень для представлення загальних понять, таких як класи, компоненти, узагальнення, об’єднання і поведінка, і зосередитися на проєктуванні та архітектурі.

UML використовується для моделювання бізнес-процесів, системного проєктування та відображення організаційних структур. Вона не прив’язана до певної платформи або мови програмування, що дозволяє з’єднати мережі різних систем. UML містить стандартний набір діаграм і нотацій різних видів, які допомагають організувати спілкування між замовниками і розробниками.

Нижче наведено список діаграм, які містить UML.

Діаграми структури:

– діаграма класів (Class Diagrams) – моделює статичну структуру класів системи і зв’язків між ними;
– діаграми компонентів (Component Diagrams) – моделюють ієрархію компонентів або підсистем системи;
– діаграми розміщення (Deployment Diagrams) – моделюють фізичну архітектуру системи.

Моделі поведінки: діаграми варіантів використання (Use Case Diagrams) – моделюють функціональні вимоги до системи у вигляді сценаріїв взаємодії користувачів з системою.

### 2.1.1 Діаграма прецедентів

Для опису функціональної структури системи використовується діаграма варіантів використання. Цей тип діаграми описує загальну функціональність системи. Вона відображає взаємодію між варіантами використання, які представляють функції системи, та діючими особами, які представляють людей або системи, що отримують або передають інформацію в дану систему. Кожна функціональність зображується у вигляді «прецедентів використання» (use case) або просто прецедентів. Прецедент – це типова взаємодія користувача з системою, яка може представляти різні рівні деталізації, описувати видиму користувачем функцію, забезпечувати досягнення конкретної мети, важливої для користувача. Прецедент зображується як овал, пов’язаний з типовими користувачами, які називаються акторами (actors). Список всіх прецедентів фактично визначає функціональні вимоги до системи.

Мета даної діаграми полягає в наступному: проєктовані бізнес-процеси представляються у формі так званих варіантів використання, з якими взаємодіють зовнішні сутності або актори. При цьому актором називається будь-який об’єкт, суб’єкт або система, що взаємодіє з модельованою компанією ззовні. Це може бути людина, технічний пристрій, програма або система, яка служить джерелом впливу на модельовані бізнес-процеси системи так, як визначить розробник. Варіант використання служить для опису сервісів і функцій, які компанія надає замовнику. Іншими словами, кожен варіант використання визначає набір дій, які здійснює компанія при взаємодії з замовником – актором. При цьому не наводиться конкретна реалізація взаємодії акторів з компанією та виконання варіантів використання.

Переваги моделі варіантів використання полягають у наступному:

– визначає користувачів і межі системи;
– визначає системний інтерфейс;
– зручна для спілкування користувачів з розробниками;
– використовується для написання тестів;
– служить основою для написання документації користувача.

Між користувачами і варіантами використання можуть існувати різні види зв’язків. Основні види зв’язків наступні:

– включення (include): показує, що варіант використання включається в базову послідовність і завжди виконується;
– розширення (extend): показує, що варіант використання розширює базову послідовність дій і включає власну послідовність (відмінністю від включення є те, що розширена послідовність може виконуватись у залежності від певних умов).

На діаграмі прецедентів зображуються ці взаємодії між акторами та прецедентами.

На рисунку 2.1 представлена діаграма прецедентів проєкту.

На діаграмі представлені два актора «Кандидат» та «Рекрутер». Актор «Кандидат» – це особа, яка шукає роботу в IT-сфері та використовує систему для пошуку вакансій, проходження тестів, взаємодії з рекрутерами тощо. Кандидат може здійснювати пошук вакансій, подавати заявку до них та взаємодіяти з рекрутерами через чат, для обговорення умов роботи. Актор «Рекрутер» – це представник HR-агентства або компанії, який відповідає за пошук та відбір кандидатів для IT-позицій. Рекрутер має доступ до бази даних резюме кандидатів. Він може створювати вакансії та публікувати їх у системі.

Розглянемо прецеденти, наведені на діаграмі 2.1.

**Створити вакансію**: рекрутер створює нову вакансію, вказуючи назву, опис, вимоги та умови. Після створення вакансії вона стає доступною для перегляду кандидатами.

**Редагувати вакансію**: рекрутер може редагувати існуючу вакансію, змінюючи інформацію про неї, таку як опис, вимоги або умови. Це дозволяє підтримувати актуальну інформацію про вакансії.

**Перегляд заявок на вакансію**: рекрутер може переглядати список заявок, які надійшли на конкретну вакансію, для оцінки і відбору потенційних кандидатів. Це допомагає ефективно керувати процесом відбору кандидатів.

**Вихід з акаунту**: користувач може вийти зі свого акаунту у системі, закінчивши поточну сесію роботи.

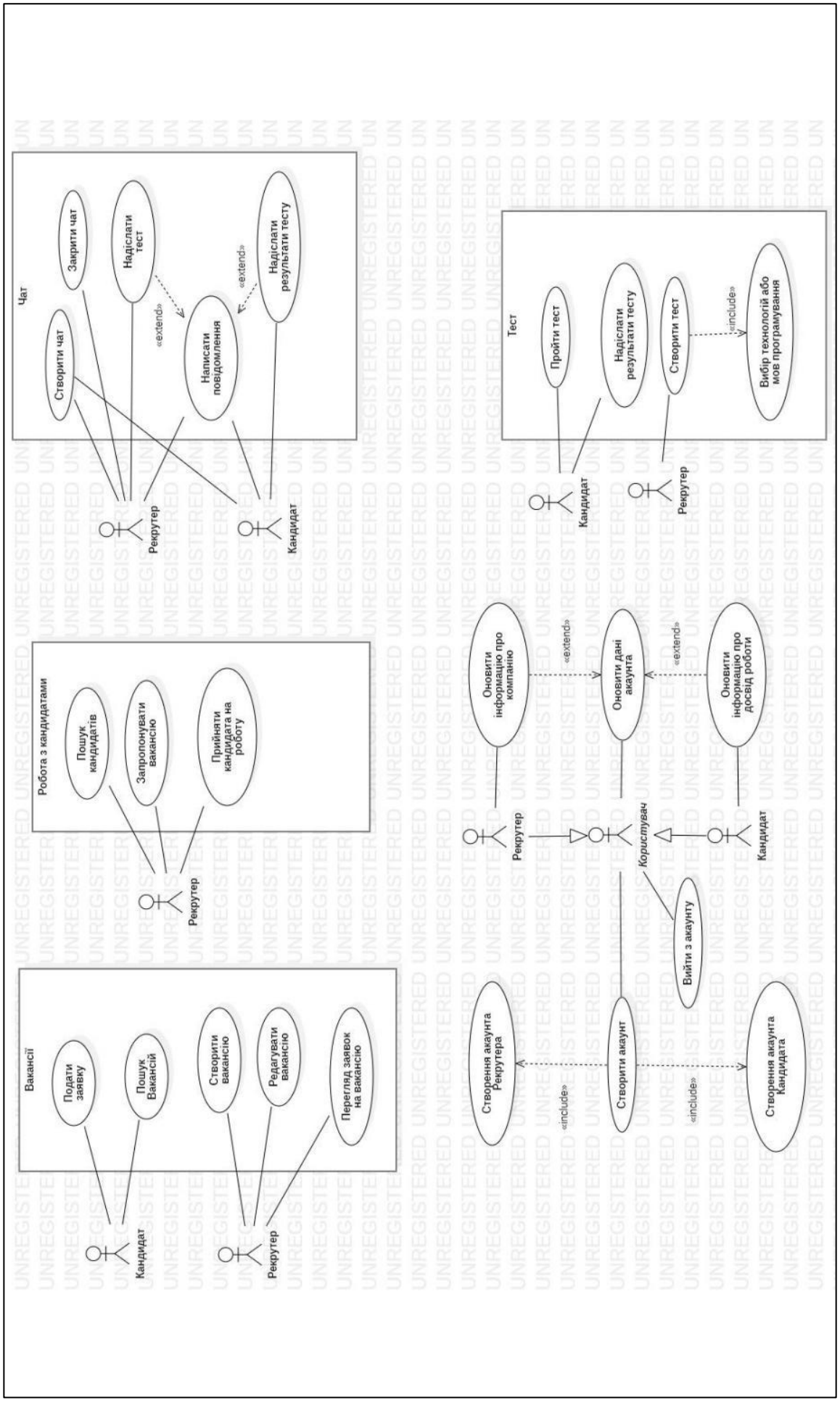



Рисунок 2.1 – Діаграма прецедентів усього проєкту

**Подати заявку**: кандидат подає заявку на вакансію, заповнюючи відповідну форму з обов’язковими даними, такими як резюме та контактна інформація. Після подання заявки кандидат може отримувати сповіщення про статус заявки.

**Пошук вакансій**: кандидат використовує функцію пошуку вакансій для знаходження потрібної позиції, вводячи ключові слова або фільтруючи за категоріями. Після знаходження вакансії кандидат може переглянути деталі та подати заявку на неї.

**Пошук кандидатів**: рекрутер використовує систему для пошуку кандидатів з відповідними навичками та досвідом, використовуючи ключові слова або фільтри. Після знаходження потенційних кандидатів, рекрутер може переглянути їхні резюме та відправити запрошення на співбесіду.

**Запропонувати вакансію**: рекрутер може надіслати запрошення на співбесіду обраному кандидату для певної вакансії, вказавши дату, час та місце співбесіди.

**Прийняти кандидата на роботу**: після проведення співбесіди та вибору найкращого кандидата, рекрутер може здійснити прийняття на роботу, узгодивши умови з кандидатом та відправивши йому офіційну пропозицію про працевлаштування.

**Відмовити від прийняття на роботу**: у випадку, якщо кандидат не відповідає очікуванням або не підходить після співбесіди, рекрутер може відмовити йому в прийнятті на роботу, повідомивши про це та можливо надавши конструктивний фідбек.

**Створити акаунт**: користувач має можливість створити акаунт у системі, під час чого він обирає роль акаунта: Рекрутер або Кандидат. Для Рекрутера він може вказати інформацію про компанію, а для Кандидата – додаткову інформацію про свій досвід роботи.

**Оновити дані акаунту**: користувач може оновити інформацію у своєму акаунті. Рекрутер може змінювати дані про компанію, такі як адреса, контактна інформація тощо. Кандидат може оновити свій досвід роботи, додати нові навички чи виконані проєкти.

**Створити чат**: рекрутер або кандидат можуть створити чат, ініціюючи його взаємодію. Кандидат може створити чат з рекрутером після відправлення заявки на вакансію, або рекрутер може створити чат з кандидатом після надсилання пропозиції з вакансією. Цей чат дозволяє їм обговорювати деталі вакансії, умови праці, а також взаємно задавати питання.

**Надіслати повідомлення**: обидва актори, рекрутер і кандидат, можуть надсилати текстові повідомлення у чаті для обговорення умов роботи та інших важливих питань. Рекрутер може також надсилати тести як частину процесу відбору, які будуть відправлені у вигляді повідомлення. Кандидат, в свою чергу, після проходження тесту, може надіслати результати тесту у вигляді повідомлення.

**Закрити чат**: рекрутер має можливість закрити чат у випадку, якщо кандидат не підходить для вакансії або в результаті інших обставин. Після закриття чату, подальша комунікація між ними в цьому чаті стає недоступною.

### 2.1.2 Діаграма класів

Діаграма класів є графічним зображенням структури класів та взаємозв’язків між ними в об’єктно-орієнтованому програмуванні [9]. Вона допомагає візуалізувати основні компоненти системи, такі як класи, атрибути та методи, а також залежності та взаємодії між класами.

Діаграми класів забезпечують зрозумілу та структуровану інформацію про архітектуру програми, допомагають при аналізі, проєктуванні та розробці програмного забезпечення. Вони можуть використовуватись як комунікаційний засіб між розробниками, дизайнерами та іншими учасниками проєкту.

За допомогою діаграм класів можна визначити ієрархію класів, спадкування, асоціації, агрегації та композиції між об’єктами. Вони також дозволяють відображати різні рівні видимості, модифікатори доступу та залежності між класами.

На зображенні 2.2 представлено діаграму класів проєкту.

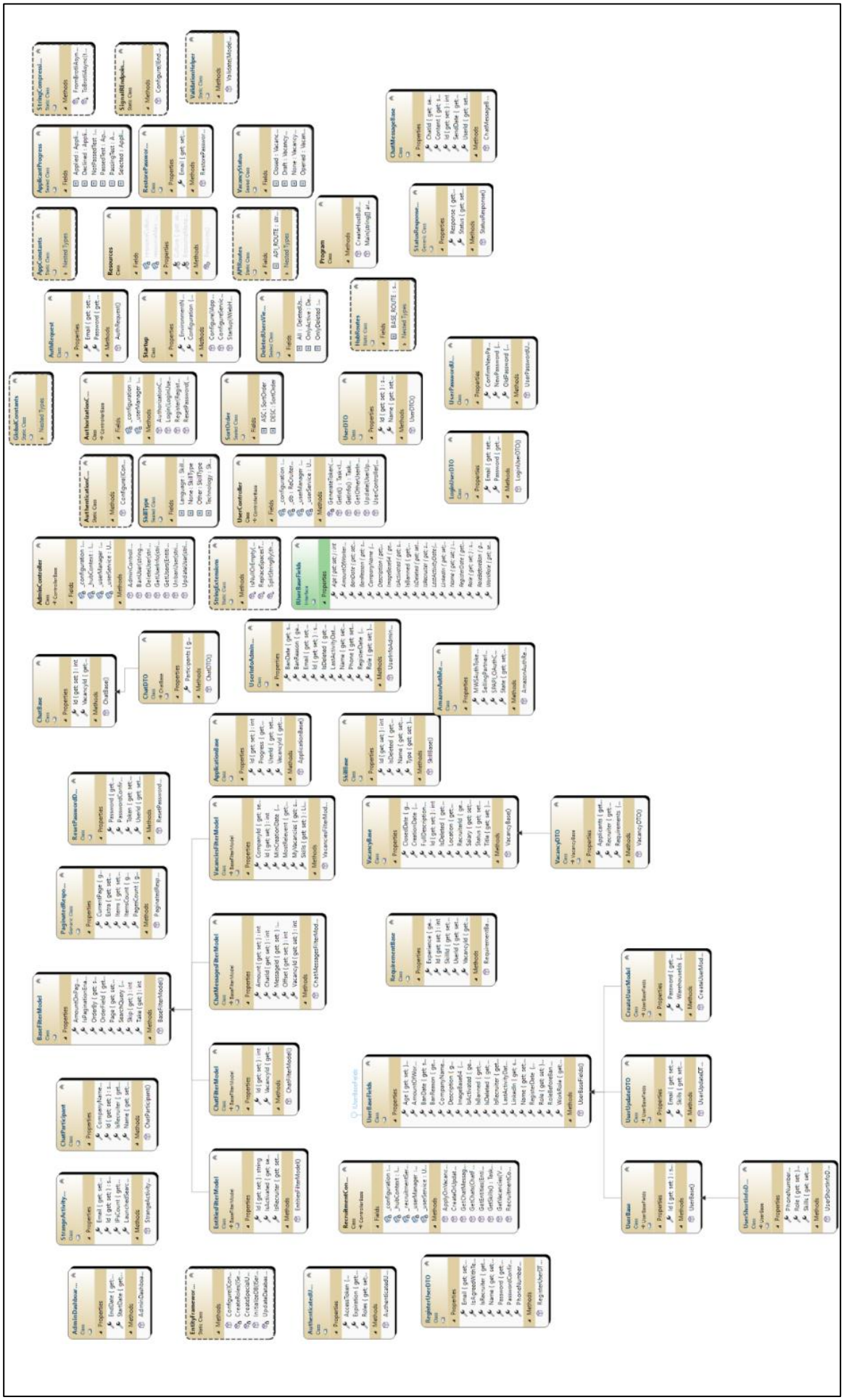


Рисунок 2.2 – Діаграма класів

### 2.1.3 Діаграма компонентів

Діаграма компонентів є важливим інструментом в архітектурі програмного забезпечення для візуалізації компонентної структури системи. Вона надає зрозуміле та зручне представлення взаємодії між компонентами системи, їх розміщення та взаємозв’язок.

Діаграма компонентів відображає фізичну або логічну структуру системи, включаючи компоненти, їх взаємозв’язки та взаємодію зовнішніх систем або користувачів. Вона часто використовується на етапі проєктування системи, де деталізується архітектурний дизайн та взаємодія компонентів.

У діаграмі компонентів компоненти представлені у вигляді прямокутників або куль, які відображають фізичні або логічні модулі системи. Кожен компонент має ім’я та може мати додаткові атрибути, такі як версія, автор або дата створення. Компоненти можуть бути вкладені один в одного для показує структури та організації системи.

Взаємозв’язки між компонентами відображаються за допомогою стрілок або ліній, які показують напрямок та тип взаємодії. Це можуть бути залежності між компонентами, такі як використання одного компонента іншим, або взаємодія за допомогою інтерфейсів, подій або повідомлень.

Діаграма компонентів також може включати інформацію про розміщення компонентів на фізичних або логічних вузлах. Наприклад, компоненти можуть бути розподілені по різних серверах або включені в один процес. Це допомагає уявити фізичну структуру системи та розподілення навантаження.

Діаграма компонентів є корисним інструментом для комунікації між членами команди розробників, архітекторами та зацікавленими сторонами. Вона допомагає уточнити взаємозв’язки між компонентами, визначити границі відповідальності та підтримувати зрозумілість архітектури системи.

Крім того, діаграма компонентів може бути використана як основа для подальшого проєктування системи, визначення розподілення ресурсів та оптимізації взаємодії між компонентами.

Узагальнюючи, діаграма компонентів є потужним інструментом для візуалізації структури та взаємодії компонентів системи. Вона допомагає розробникам, архітекторам та зацікавленим сторонам краще розуміти систему, визначити залежності та забезпечити належну організацію програмного забезпечення.

На рисунку 2.3 представлено діаграму компонентів проєкту.

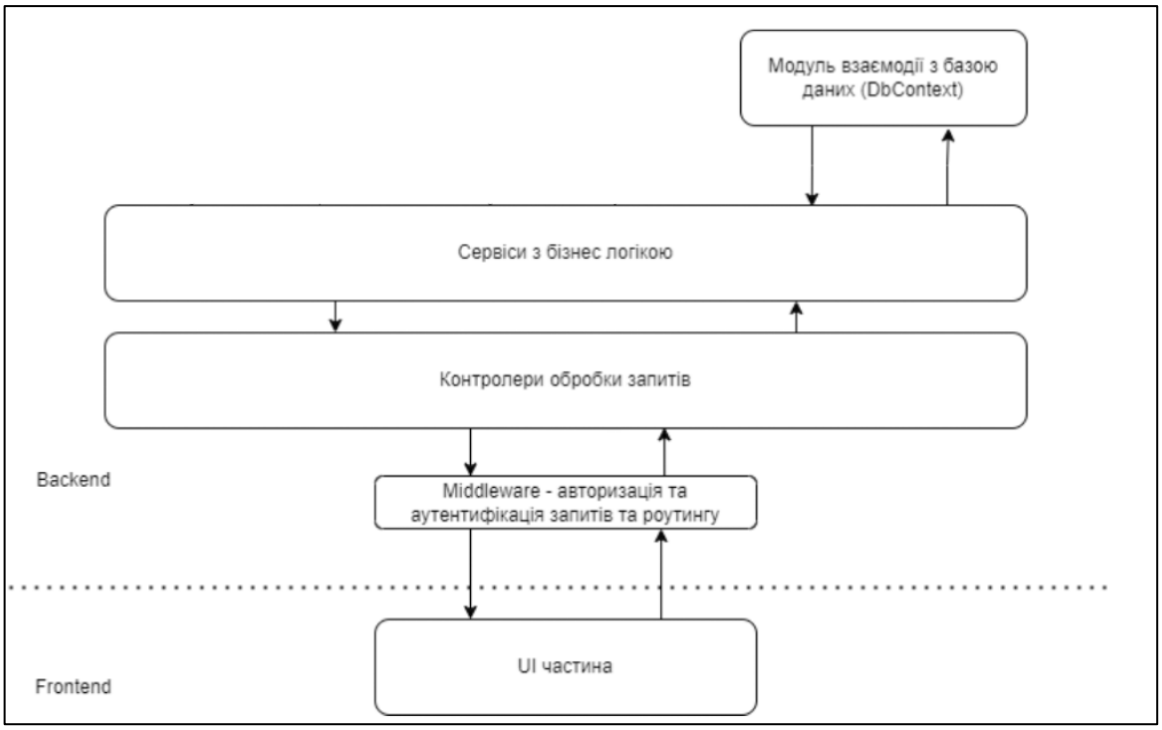



Рисунок 2.3 – Діаграма компонентів

Проєкт складається з двох основних частин – бекенду та фронтенду. Фронтенд відображає усю необхідну інформацію, яка обробляється та зберігається на бекенді.

Бекенд складається з контролерів обробки запитів. Кожен контролер стосується своєї логічної частини системи. Усього є 6 контролерів: Admin, Authorization, Recruitment, User.

В залежності від контролера та роуту – викликається різний сервіс бізнес логіки які вже, в свою чергу, в залежності від бізнес задачі можуть використовувати додаткові модулі системи – як наприклад модуль взаємодії з базою даних (DbContext).

### 2.1.4 Діаграма розміщення

Діаграма розміщення є важливим інструментом в архітектурі програмного забезпечення, який дозволяє візуалізувати фізичну або логічну розстановку компонентів системи та їх взаємодію з фізичними або віртуальними ресурсами. Ця діаграма допомагає розробникам та архітекторам краще розуміти, як компоненти системи розміщені на різних фізичних або логічних вузлах і як вони спілкуються між собою та з навколишнім середовищем.

У діаграмі розміщення компоненти системи відображаються у вигляді прямокутників або куль, а фізичні або віртуальні вузли представлені у вигляді прямокутників або овалів. Кожен компонент та вузол має ім’я та може мати додаткові атрибути, такі як версія, IP-адреса, серійний номер тощо.

Взаємодія між компонентами та вузлами відображається за допомогою стрілок або ліній, які показують напрямок комунікації. Це можуть бути з’єднання між компонентами на одному вузлі, мережеві з’єднання між вузлами, залежності між компонентами та інші типи взаємодії.

Діаграма розміщення також може включати додаткову інформацію про фізичні або логічні характеристики вузлів, такі як обладнання, операційна система, середовище виконання та доступність ресурсів. Це допомагає визначити, як компоненти системи будуть розподілені та працювати в реальному середовищі.

Діаграма розміщення є важливим інструментом для планування, аналізу та оптимізації фізичної архітектури системи. Вона допомагає зрозуміти, як компоненти взаємодіють з фізичними або віртуальними ресурсами, як забезпечувати високу доступність та масштабованість, а також як управляти використанням ресурсів.

Діаграма розміщення може бути корисною як на етапі проєктування системи, так і під час її подальшого розвитку і супроводу. Вона допомагає виявити потенційні проблеми в архітектурі, визначити необхідність масштабування або оптимізації ресурсів та забезпечити належну роботу системи в реальному середовищі.

На рисунку 2.4 представлено діаграму розміщення проєкту.

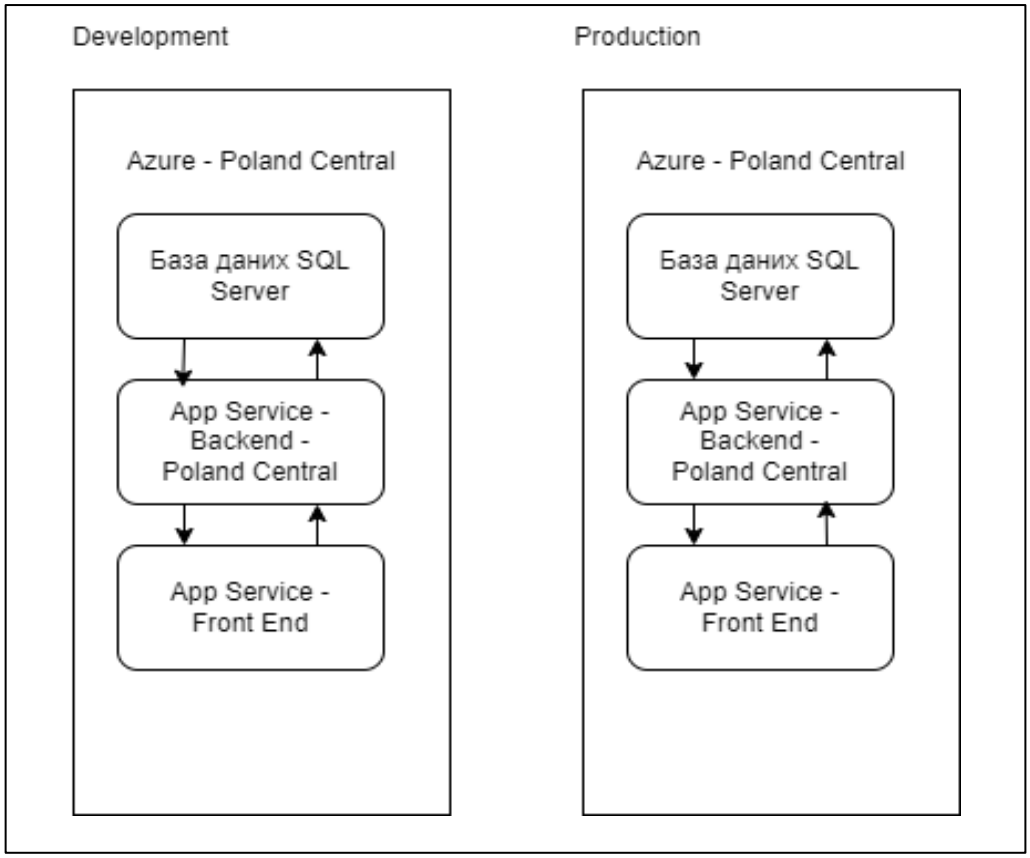



Рисунок 2.4 – Діаграма розміщення

В нашому випадку, наша система розміщена на серверній архітектурі Azure, що знаходиться в Польщі. Azure є хмарною платформою, яка надає надійне та масштабоване середовище для розгортання та виконання нашого додатку. Вибір Azure для розміщення нашої системи має декілька переваг. Перш за все, Azure забезпечує високу доступність і надійність завдяки своїй глобальній інфраструктурі та резервному копіюванню даних. Це означає, що наш додаток буде доступним і функціональним для користувачів у будь-який час. Крім того, Azure надає різноманітні сервіси, які полегшують розгортання, масштабування та керування нашою системою. Наприклад, ми можемо використовувати служби Azure для керування базами даних, зберігання файлів, моніторингу та логування, автоматичного масштабування ресурсів та багато іншого.

Розміщення нашої системи в Польщі важливо з точки зору швидкості взаємодії з системою. Основна локація взаємодії – Україна, найближчий сервер

– Польща. Усі ці фактори роблять Azure відмінним вибором для розміщення нашої системи, забезпечуючи нам надійність, масштабованість та забезпечення дотримання вимог щодо захисту даних.

## 2.2 Архітектура серверної частини

На даному проєкті була реалізована багатошарова архітектура проєктування системи.

Багатошарова архітектура є популярним підходом при проєктуванні ASP.NET Core проєктів, оскільки вона сприяє модульності, розширюваності та підтримці принципів SOLID, що дозволяє краще управляти складністю проєкту та покращує його обслуговування у майбутньому [4]. У такій архітектурі можна виділити кілька ключових шарів, таких як шар бізнес-логіки, шар доступу до даних і шар презентації.

**Перший шар** – шар бізнес-логіки – Business Logic Layer (BLL), відповідає за обробку бізнес-правил та логіки додатку. Він містить класи, які визначають моделі даних, сервіси та менеджери, що здійснюють операції з цими моделями. Шар бізнес-логіки виконує основні операції над даними, реалізує бізнес-правила та логіку додатку.

**Другий шар** – шар доступу до даних – Data Access Layer (DAL), відповідає за роботу з базою даних або іншими джерелами даних. В ньому реалізовано патерн репозиторій, де класи репозиторіїв забезпечують методи для звернення до даних, додавання, видалення та оновлення записів дозволяючи розділити логіку доступу до даних від бізнес-логіки. Це полегшує тестування та зміну джерела даних без зміни бізнес-логіки. Шар доступу до даних може включати ORM (Object-Relational Mapping) для спрощення роботи з базою даних. ORM (Object-Relational Mapping) – це технологія, яка дозволяє зв’язувати об’єктно-орієнтовану модель програми з реляційною базою даних. Вона надає зручний спосіб взаємодії з базою даних, де об’єкти програми відображаються в таблиці

бази даних, а доступ до даних здійснюється за допомогою об’єктів та їх властивостей. ORM дозволяє розробникам працювати з даними в більш об’єктно-орієнтованому стилі, не звертаючи прямої уваги до складностей реляційних баз даних. Замість написання складних SQL-запитів, розробник може використовувати методи та властивості об’єктів для здійснення операцій з базою даних, таких як вставка, оновлення, видалення та отримання даних [5]. На проєкті було використано технологію Entity Framework від Microsoft, яка реалізує в собі концепцію ORM.

**Третій шар** – шар презентації або контролери, відповідає за обробку запитів від клієнта і відображення відповідей. Контролери приймають HTTP-запити, взаємодіють з шаром бізнес-логіки та шаром доступу до даних і повертають відповіді у вигляді HTTP-відповідей або представлень.

Крім основних шарів, багатошарова архітектура може включати інші шари, такі як шар інфраструктури або шар безпеки, в залежності від вимог проєкту. Шар інфраструктури забезпечує зовнішні залежності, такі як підключення до бази даних або зовнішні API. Шар безпеки відповідає за аутентифікацію, авторизацію та захист додатку.

Така багатошарова архітектура дозволяє зберігати логіку додатку відокремлено від інфраструктури та інших компонентів. Вона спрощує тестування, підтримку та розширення проєкту, оскільки кожен шар виконує конкретні функції та має чіткі відповідальності. Крім того, вона дозволяє використовувати принцип інверсії залежностей (Dependency Inversion Principle), де високорівневі компоненти не залежать від низькорівневих компонентів, або залежать від абстракцій, а не від конкретних реалізацій.

Загалом, багатошарова архітектура з патерном репозиторію є потужним підходом для проєктування ASP.NET Core проєктів, оскільки вона допомагає створити добре організовану, розширювану та тестовану систему з чітко визначеними відповідальностями для кожного шару. При правильному проєктуванні та реалізації багатошарової архітектури ASP.NET Core проєкт стає більш масштабованим, підтримуваним та зручним у використанні.

### 2.2.1 Серверні статуси

Серверні статуси, також відомі як статуси відповіді сервера, є кодами, які повертаються сервером у відповідь на HTTP-запити. Ці статуси передають результат обробки запиту сервером і надають клієнту інформацію про стан запиту, чи було все успішно виконано, чи сталася помилка, які дії треба вжити далі тощо. Серверні статуси допомагають узгоджувати комунікацію між клієнтом і сервером, а також визначати наступні кроки для обробки запиту.

Основні групи серверних статусів визначаються першим числом у коді статусу. Десятки вказують на клас статусу, а останні дві цифри вказують на конкретний статус. Основні класи статусів включають:

- 1xx – інформаційні статуси: ці статуси вказують, що сервер отримав запит, і процес обробки триває, наприклад, 100 Continue показує, що сервер готовий прийняти наступну частину запиту;
- 2xx – статуси успішного виконання: ці статуси показують, що запит клієнта був успішно оброблений сервером, наприклад, 200 OK вказує на успішну обробку запиту і передачу результату клієнту;
- 3xx – статуси перенаправлення: ці статуси вказують на необхідність перенаправлення клієнта на іншу адресу або ресурс, наприклад, 301 Moved Permanently означає, що ресурс був переміщений назавжди і клієнт повинен звертатися до нової адреси;
- 4xx – статуси помилок клієнта: ці статуси показують, що виникла помилка на боці клієнта, наприклад, 404 Not Found означає, що запитуваний ресурс не знайдено на сервері;
- 5xx – статуси помилок сервера: ці статуси показують, що виникла помилка на боці сервера, наприклад, 500 Internal Server Error означає, що сталася внутрішня помилка сервера при обробці запиту.

Кожен серверний статус має своє призначення і використовується для передачі певної інформації про стан запиту. Розуміння цих статусів допомагає розробникам та адміністраторам серверів ефективно взаємодіяти з клієнтами і вирішувати проблеми, які виникають під час обробки запитів.

# 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

## 3.1 Опис інструментів для розробки клієнтської частини

### 3.1.1 Мова програмування TypeScript

TypeScript – це мова програмування, розроблена корпорацією Microsoft у 2012 році. Вона позиціонується як інструмент для розробки вебдодатків, що розширює можливості JavaScript [14].

Ключові переваги TypeScript порівняно з JavaScript:

– статична типізація: TypeScript дозволяє виявляти помилки на етапі компіляції, що спрощує подальшу розробку та підтримку коду;
– сумісність із JavaScript: TypeScript є надбудовою над JavaScript, тому він сумісний з усіма існуючими бібліотеками та фреймворками JavaScript;
– передові можливості програмування: використання дженериків, декораторів та інших сучасних функцій мови робить код більш гнучким та потужним;
– покращена підтримка IDE: редактори коду інтелектуально реагують на типізований код, надаючи підказки та автодоповнення.

### 3.1.2 Vite

Vite – це швидкий і гнучкий інструмент, для збірки веб додатків, який має на меті надати швидший та легший досвід розробки. Головні особливості та переваги даного інструменту над іншими інструментами для збірки вебпроєктів:

– **миттєвий запуск сервера**: Vite надає можливість негайно запустити сервер розробки, він обслуговує файли нативних модулів ES без

необхідності упаковки (бандлінгу) – це дозволяє працювати з кодом без зайвих затримок;

- **швидкий Hot Module Replacement (HMR)**: HMR в Vite залишається швидким незалежно від розміру додатка – ви можете вносити зміни в коді, і вони миттєво відображаються без перезавантаження сторінки;
- **багатофункціональний сервер розробки**: Vite надає розширені можливості над нативними модулями ES, такі як підтримка TypeScript, JSX, CSS та інших функцій «з коробки»;
- **оптимізована збірка**: при виведенні коду в продакшн Vite використовує попередньо налаштований Rollup для створення високооптимізованих статичних ресурсів;
- **плагіни та розширення**: Vite підтримує систему плагінів, які дозволяють розширити його функціональність;
- **повна типізація API**: Vite надає гнучкі програмні API з повною підтримкою TypeScript.

### 3.1.3 React

React – це бібліотека для розробки інтерактивних вебінтерфейсів [11]. Основна мета React полягає в спрощенні створення складних користувацьких інтерфейсів, розділяючи їх на компоненти, які оновлюються автоматично при зміні даних [12]. Декларативний підхід робить код передбачуваним та легко налагоджуваним. Компоненти можуть також зберігати внутрішній стан, що дозволяє відстежувати зміни та відображати їх на інтерфейсі [13].

## 3.2 Опис інструментів для розробки серверної частини

Для реалізації серверної частини були використані стек технологій .NET, Visual Studio, C#, ASP.NET Core, MS SQL DB, Swagger, Postman, Azure, .NET

CLI, Nuget, Chrome DevTools та набір технологій для парсингу даних.

C# – це об’єктно-орієнтована мова програмування, що використовується в екосистемі .NET. Вона дозволяє розробникам створювати різноманітні програми для платформ Windows, вебдодатків, служб та інших рішень. Розроблений у 1998-2001 роках групою інженерів під керівництвом Андерса Хейлсберга у компанії Microsoft як мова розробки програм для платформи Microsoft .NET Framework і згодом був стандартизований як ECMA-334 та ISO/IEC 23270. C# відноситься до сім’ї мов з C-подібним синтаксисом, з них його синтаксис найближчий до C++ та Java. Мова має статичну типізацію, підтримує поліморфізм, навантаження операторів (у тому числі операторів явного та неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи та методи, ітератори, анонімні функції з підтримкою замикань, LINQ, винятки, коментарі в формат XML.

C#, мова програмування, багато вчилася від своїх попередників, таких як C++, Pascal, Modula, Smalltalk і Java, і спирається на їхню практику. Проте, вона також уникає деяких моделей, які виявилися проблематичними при розробці програмних систем. Наприклад, C# не підтримує множинне успадкування класів, але дозволяє множинне успадкування інтерфейсів. C# був розроблений як мова програмування прикладного рівня для Common Language Runtime (CLR) і, як такий, залежить від можливостей CLR. Основна система типів C#, яка відображає Base Class Library (BCL), також залежить від CLR. Мовні особливості C# визначаються можливістю їх перекладу в відповідні конструкції CLR. З розвитком CLR від версії 1.1 до 2.0, мова C# значно збагатилася. Очікується подальше збагачення мови відповідно до розвитку CLR. Зауважимо, що з виходом C# 3.0 було порушено цю закономірність, оскільки ця версія мови представляє собою розширення, яке не спирається на розширення платформи. CLR надає C# (а також іншим мовам .NET) багато можливостей, які відсутні в «класичних» мовах програмування. Наприклад, збирання сміття не реалізовано безпосередньо в C#, але виконується CLR для програм, написаних на C#, аналогічно до того, як це відбувається для програм на VB.NET, J# та інших мовах.

Visual Studio – це інтегроване середовище розробки (IDE) для платформи .NET. Воно надає розробникам зручні інструменти для написання, налагодження та візуального проєктування програм на різних мовах, включаючи C#.

ASP.NET Core – це вебфреймворк, розроблений компанією Microsoft, призначений для створення вебдодатків і вебсервісів. Цей фреймворк є наступником попередньої версії ASP.NET і має багато переваг і поліпшень. Однією з ключових переваг ASP.NET Core є його крос-платформеність. Фреймворк підтримує роботу на різних операційних системах, таких як Windows, macOS та Linux, що дає можливість розробляти вебдодатки на будь-якій обраній платформі. ASP.NET Core забезпечує широкі можливості для розробки вебдодатків різного типу. Зокрема, він підтримує розробку вебсторінок з використанням HTML, CSS і JavaScript, а також розробку вебсервісів (API) для обробки і передачі даних. Для роботи з API часто використовується ASP.NET Core WebAPI, який надає зручні засоби для створення RESTful API. ASP .NET WebAPI забезпечує широкі можливості для реалізації контролерів, які обробляють HTTP запити та повертають дані у вигляді JSON або XML.

MS SQL DB (Microsoft SQL Database) – це система управління базами даних, розроблена компанією Microsoft. Вона є однією з найпопулярніших реляційних баз даних і широко використовується в багатьох вебдодатках і підприємствених системах [6]. MS SQL DB надає надійну та масштабовану роботу з даними, підтримує мову запитів SQL для маніпулювання даними і має багато інструментів для адміністрування та оптимізації бази даних [7].

Entity Framework Code First – це підхід до розробки програмного забезпечення, що використовує Entity Framework (EF) для взаємодії з базою даних. Цей підхід дозволяє розробникам визначати модель даних у вигляді класів (класи-сутності), а потім на основі цих класів автоматично створювати схему бази даних. У режимі Code First, розробники визначають класи-сутності, які відображають таблиці бази даних, а також зв’язки між ними [8]. Entity Framework використовує цю модель класів для створення або оновлення схеми бази даних, що відповідає цим класам. Крім того, Entity Framework Code First надає

можливості для налаштування взаємодії з базою даних, включаючи визначення обмежень, індексів, збережених процедур і т.д. За допомогою атрибутів або Fluent API, розробники можуть вказати додаткові деталі про модель даних і взаємодію з базою даних. Один з ключових принципів Code First – це автоматична міграція бази даних. При зміні моделі даних (додавання, видалення або зміна властивостей класів-сутностей), Entity Framework може автоматично створити скрипти для оновлення схеми бази даних, що забезпечує зручну роботу з розвитком бази даних в процесі розробки. Entity Framework Code First надає зручний спосіб розробки програмного забезпечення, що використовує базу даних. Він дозволяє розробникам зосередитися на моделюванні даних у вигляді класів і мінімізує необхідність безпосередньо працювати з SQL-запитами і мануальною роботою з базою даних.

Swagger – це набір інструментів для створення, документування та використання RESTful API. Він надає можливість автоматично створювати документацію для API на основі його коду і опису ресурсів. Swagger також дозволяє виконувати тестування API, візуалізувати його структуру та взаємодіяти з ним через вебінтерфейс. Завдяки Swagger розробники можуть легко спілкуватися та співпрацювати при роботі з API, а також забезпечувати його документацію для сторонніх користувачів.

Postman – це популярний інструмент для тестування та взаємодії з API. Він дозволяє розробникам створювати HTTP запити до API, виконувати їх, переглядати та аналізувати відповіді сервера. Postman надає зручне середовище для налагодження, тестування і належного функціонального тестування API. Він також підтримує автоматизацію тестування та можливість створювати колекції запитів для організації роботи з багатьма API.

Azure – це хмарна платформа, розроблена компанією Microsoft, яка надає набір послуг для розгортання, керування та розширення вебдодатків та інфраструктури. Azure пропонує широкий спектр сервісів, таких як вебхостинг, бази даних, аналітика, інтеграція, машинне навчання та багато інших. Ця платформа дозволяє розробникам швидко та ефективно розгортати свої додатки

в хмарі, масштабувати їх ресурси в залежності від потреб і отримувати надійність і безпеку, які надаються Microsoft. Azure є популярним вибором для розробки та розгортання ASP.NET Core додатків.

Chrome DevTools – це набір інструментів для розробників, що надається в браузері Google Chrome. Вони дозволяють розробникам аналізувати, налагоджувати та оптимізувати вебсторінки та вебдодатки. Chrome DevTools надає розширені можливості для перевірки та відлагодження коду, відстеження мережевих запитів, профілювання продуктивності, зміни елементів сторінки в режимі реального часу та багато іншого. Вони є потужним інструментом для веброзробників, що допомагає покращити продуктивність та якість вебдодатків.

.NET CLI (Command Line Interface) – це інструментарій командного рядка для розробки, збирання та керування проєктами на платформі .NET. Завдяки .NET CLI розробники можуть створювати нові проєкти, компілювати їх, запускати, тестувати та розгортати додатки з використанням команд терміналу. .NET CLI надає простий та уніфікований спосіб управління проєктами на платформі .NET та є невід'ємною частиною розробки .NET-додатків.

NuGet – це менеджер пакетів для платформи .NET, що дозволяє розробникам легко встановлювати, оновлювати та керувати залежностями пакетів в їхніх проєктах. Використання NuGet дозволяє легко і швидко додавати сторонні бібліотеки, компоненти та інші ресурси до проєкту .NET. Він має велику кількість доступних пакетів, які можна встановити з централізованого репозиторію NuGet.

HtmlAgilityPack, HttpClient, XPath і JsonPath – це набір інструментів, які часто використовуються в розробці програмного забезпечення для отримання, обробки та аналізу вебданих.

HtmlAgilityPack – це бібліотека для роботи з HTML-документами у .NET. Вона дозволяє отримувати доступ до різних елементів HTML-сторінки, зчитувати і записувати дані, модифікувати структуру сторінки і виконувати різні маніпуляції з HTML-документами. HtmlAgilityPack дозволяє зручно взаємодіяти з HTML-кодом, використовуючи різні методи та властивості.

HttpClient – це клас у .NET, який надає можливості для відправки HTTP-запитів до вебсерверів і отримання відповідей. HttpClient дозволяє розробникам взаємодіяти з різними вебресурсами, виконувати GET, POST, PUT, DELETE-запити і отримувати дані з вебсерверів. Він забезпечує широкий функціонал для роботи з HTTP-протоколом.

XPath – це мова запитів, яка використовується для навігації і вибору елементів у XML-документах. XPath дозволяє розробникам точно визначати шлях до певних елементів XML, використовуючи певні правила та патерни. Це дозволяє зручно отримувати доступ до конкретних даних в XML-структурі.

JsonPath – це синтаксис для вибору елементів з JSON-документів. Він надає можливість точно визначити шлях до певних елементів у JSON-структурі даних, використовуючи патерни та оператори. З JsonPath можна витягти конкретні значення з великих JSON-об’єктів, робити фільтрацію, сортування та інші маніпуляції з даними.

Комбінування HtmlAgilityPack, HttpClient, XPath і JsonPath дозволяє розробникам легко отримувати доступ до вебданих, отримувати HTML-сторінки або JSON-відповіді з вебсерверів, а потім здійснювати аналіз цих даних, вибирати потрібні елементи за допомогою XPath або JsonPath і обробляти їх відповідно до вимог додатку. Це дозволяє розробникам ефективно працювати з вебданими і забезпечувати потрібну функціональність своїм додаткам.

## 3.3 Процес реалізації серверної частини

### 3.3.1 Процес створення Entity Framework Code First-моделі

Для створення Entity Framework Code First-моделі необхідно налаштувати проєкт та встановити необхідні залежності, такі як Entity Framework.

Процес створення Entity Framework Code First-моделі включає такі кроки:

– визначення класів моделі, які представляють таблиці бази даних;

- визначення взаємозв’язків між класами, таких як один до одного, один до багатьох, багато до багатьох;
- використання атрибутів або Fluent API для налаштування деталей моделі, таких як назви стовпців, обмеження, тощо;
- конфігурування зв’язку з базою даних, включаючи вибір провайдера бази даних та налаштування рядка підключення;
- застосування міграцій для створення або оновлення схеми бази даних з використанням механізму міграцій Entity Framework;
- тестування моделі на взаємодію з базою даних та відповідність вимогам додатку.

### 3.3.2 Процес створення ASP.NET WebAPI-контролера

Для створення ASP.NET WebAPI-контролера необхідно налаштувати проєкт та встановити необхідні залежності, такі як ASP.NET WebAPI Framework [2].

Процес створення ASP.NET WebAPI-контролера включає такі кроки:

- створення нового контролера з використанням шаблонів або генераторів коду, що надаються ASP.NET WebAPI Framework;
- визначення методів дій (actions) контролера, які відповідають на HTTP-запити, такі як GET, POST, PUT, DELETE;
- налаштування маршрутизації, що визначає URL-шаблони, за якими будуть доступні методи дій контролера;
- використання атрибутів, таких як [HttpGet], [HttpPost], [Route], для налаштування різних аспектів контролера, таких як тип HTTP-запиту, URL-шаблон, тощо;
- реалізація логіки дій контролера, включаючи обробку вхідних параметрів, виклик сервісів або репозиторіїв, формування відповіді;
- тестування контролера з використанням інструментів для тестування API, таких як Postman або Swagger.

Ці процеси можуть включати додаткові кроки в залежності від специфіки проєкту та використаних технологій, але наведені кроки є основними для реалізації та тестування Entity Framework Code First-моделі та ASP.NET WebAPI-контролера [3].

### 3.4 Тестування системи (backend, frontend)

Тестування системи, включаючи як бекенд, так і фронтенд, є важливою частиною розробки програмного забезпечення. Вона допомагає перевірити, чи відповідає система вимогам, функціонує правильно і не містить помилок або проблем.

Тестування системи включає такі етапи:

– планування тестування: на цьому етапі визначаються цілі тестування, обсяг тестування, розроблюється тестова стратегія і план, а також визначаються тестові набори і критерії прийняття;
– функціональне тестування: функціональне тестування перевіряє, чи відповідає система функціональним вимогам – це включає перевірку роботи окремих функцій, взаємодію між компонентами, обробку вхідних даних та виведення результатів;
– тестування інтеграції: на цьому етапі перевіряється, чи правильно працюють компоненти системи разом – виконуються тестові сценарії, що охоплюють взаємодію між бекендом і фронтендом, перевірку обміну даними, інтеграцію з базою даних та іншими системами;
– тестування продуктивності: це тестування, що оцінює продуктивність системи за допомогою різних навантажень і сценаріїв використання – можуть використовуватися інструменти для вимірювання швидкодії, часу відгуку, витрат ресурсів і реагування системи на велику кількість користувачів;
– тестування безпеки: на цьому етапі перевіряється система на наявність потенційних проблем безпеки, таких як уразливості, витоки даних,

недостатні заходи безпеки – застосовуються різні методи тестування, включаючи вторгнення, перехоплення даних та аналіз вразливостей;
- тестування відповідності: на цьому етапі перевіряється, чи відповідає система вимогам, стандартам і регуляторним вимогам, які можуть стосуватися конкретної галузі або ринку;
- валідація та верифікація: цей етап включає перевірку, чи виконується система відповідно до очікувань користувачів і бізнес-вимог, а також валідацію вхідних та вихідних даних;
- документування результатів: після завершення тестування важливо документувати результати, виявлені проблеми, заплановані виправлення і рекомендації для поліпшення системи.

## 3.5 Unit тести

Модульне тестування, або юніт-тестування – процес у програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми.

Ідея полягає в тому, щоб писати тести для кожної нетривіальної функції чи методу. Це дозволяє досить швидко перевірити, чи не привела чергова зміна коду до регресії, тобто до появи помилок у вже відтестованих місцях програми, а також полегшує виявлення та усунення таких помилок.

## 3.6 Керівництво користувача

### 3.6.1 Рівень підготовки користувача

Для роботи на сайті користувач повинен мати певну кваліфікацію, включаючи навички роботи з ПК та web-браузером. Це охоплює вміння працювати з операційною системою, встановлювати програми, працювати з файловою системою, а також розуміння основних понять та операцій в браузері.

Крім того, знайомство з керівництвом користувача допоможе зрозуміти функціонал сайту та використовувати його відповідно до інструкцій.

### 3.6.2 Підготовка до роботи

Запуск системи.

Для запуску системи необхідний доступ до сайту через інтернет за допомогою стандартного web-браузера, такого як Google Chrome, Mozilla Firefox, Opera або Safari.

Після входу на сайт користувач буде перенаправлений на сторінку входу до системи. На сайті розрізняються дві групи користувачів: анонімні користувачі, які не увійшли або не зареєстровані, і користувачі – приватні особи, які авторизувалися на сайті та мають доступ до функцій користувача у системі.

### 3.6.3 Авторизація, вхід в систему

Авторизація та вхід в систему є важливим функціоналом для забезпечення безпеки та обмеження доступу до конфіденційної інформації.

На рисунку 3.1 зображена сторінка авторизації в системі.

Для керівництва користувача надаються такі деталі: вхід в систему, захист від несанкціонованого доступу.

*Вхід в систему*:

– користувач вводить свої облікові дані (електронну пошту та пароль) на сторінці входу в систему;
– система перевіряє правильність введених даних та перевіряє, чи існує користувач з такими обліковими даними;
– при вдалій авторизації користувачу надається доступ до захищених ресурсів та функціоналу системи.

*Захист від несанкціонованого доступу*:

– для забезпечення безпеки, система використовує різні механізми, такі як шифрування паролів та механізми аутентифікації;
– зазвичай, система також встановлює обмеження на кількість невдалих спроб входу перед тим, як тимчасово заблокувати обліковий запис;
– користувачам рекомендується використовувати міцні паролі та забезпечувати їх конфіденційність.

Розробник повинен ретельно виконувати всі заходи безпеки та передбачати потенційні загрози, пов’язані з авторизацією та входом в систему [1]. Користувачам слід бути обізнаними про необхідність використання надійних паролів та правил безпеки для збереження цілісності свого облікового запису та конфіденційності особистих даних.

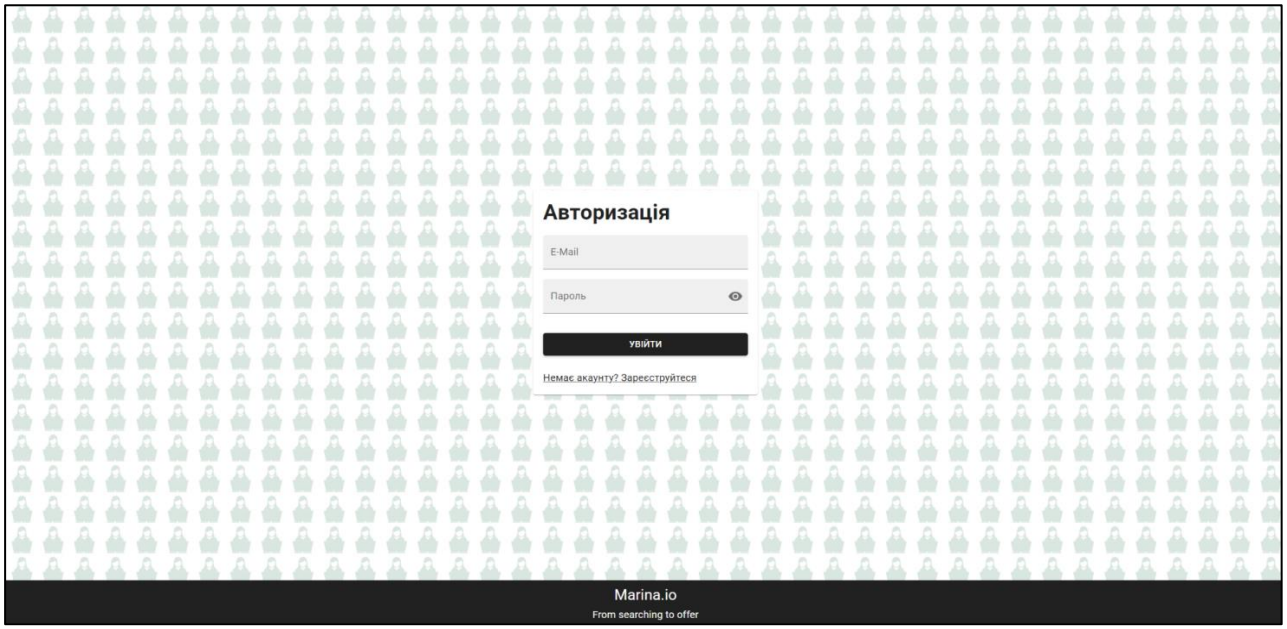



Рисунок 3.1 – Авторизація в системі

### 3.6.4 Реєстрація нового акаунту

При натисканні «Немає акаунту? Зареєструйтеся», користувач може створити новий обліковий запис, заповнивши необхідні поля.

На рисунку 3.2 зображена сторінка з реєстрацією нового акаунту.

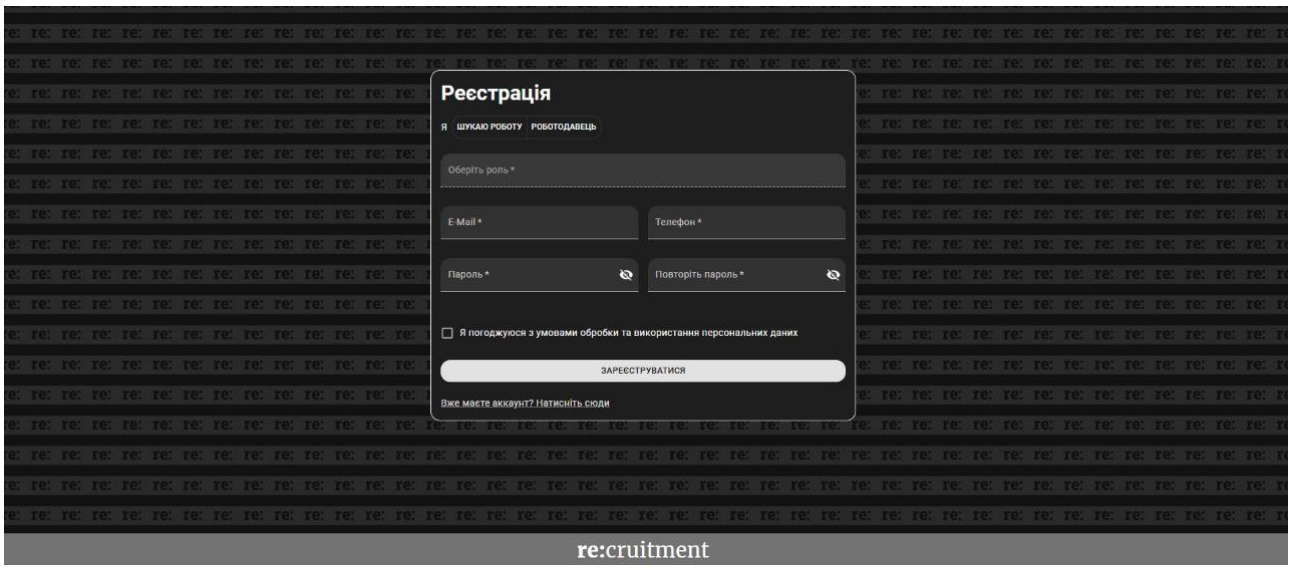



Рисунок 3.2 – Реєстрація нового акаунту

На цій сторінці користувач повинен ввести такі дані:

– обрати роль, яка відповідає вашому статусу, наприклад, «Шукаю роботу» або «Роботодавець»;
– ПІБ або назву організації в полі «Ім’я»;
– введіть свій електронний адрес у поле «E-Mail»;
– введіть свій номер телефону у поле «Телефон»;
– введіть пароль у поле «Пароль»;
– повторіть пароль у поле «Повторити пароль»;
– обов’язково позначте прапорець, якщо ви згодні з умовами обробки та використання персональних даних.

Після того, як усі поля будуть заповнені, користувач може натиснути кнопку «Зареєструватися», щоб створити новий акаунт.

Якщо користувач вже має акаунт, він може повернутися на сторінку авторизації, натиснувши на кнопку «Вже маєте акаунт? Натисніть сюди».

### 3.6.5 Особистий кабінет

На сторінці «Мій профіль» користувач може переглянути та оновити свої особисті дані та інформацію про навички та досвід роботи.

В залежності від того, хто увійшов у систему (Рекрутер або Кандидат), відображаються відповідні поля.

На рисунках 3.3 – 3.6 зображені сторінки профілів кандидата та рекрутера.

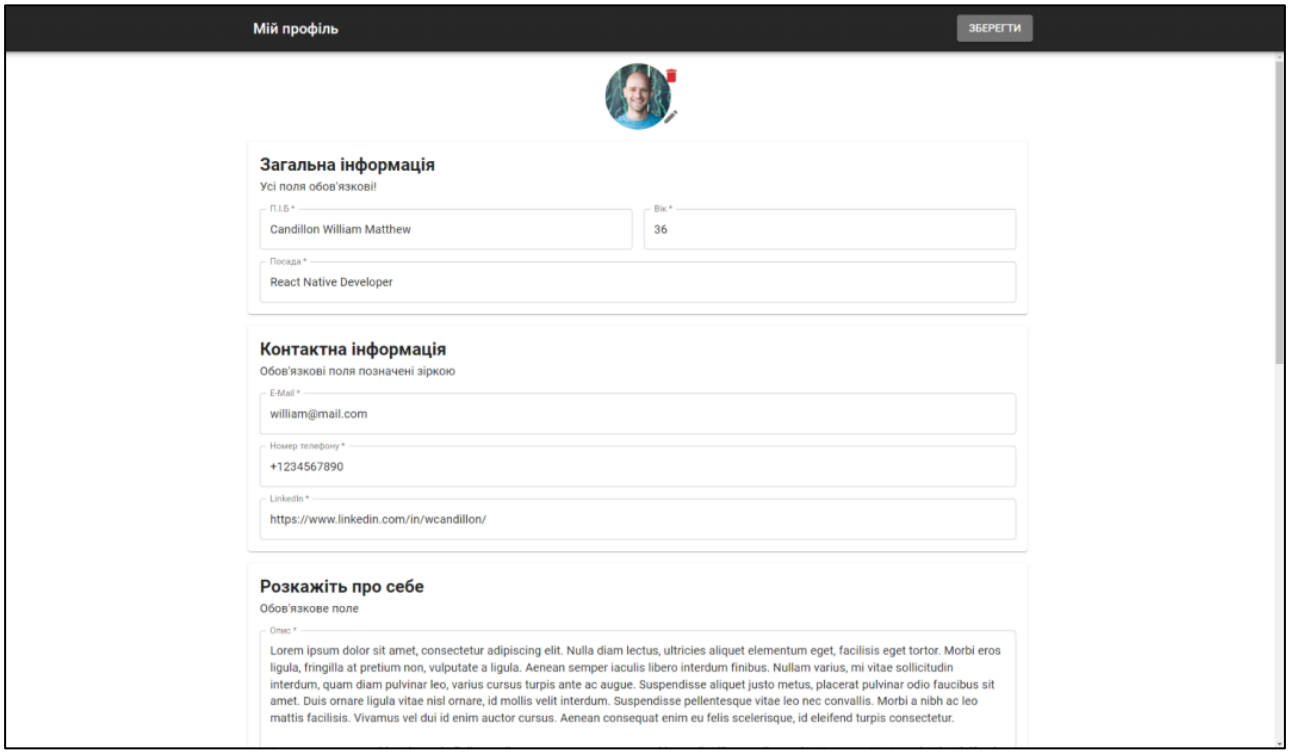



Рисунок 3.3 – Профіль Кандидата

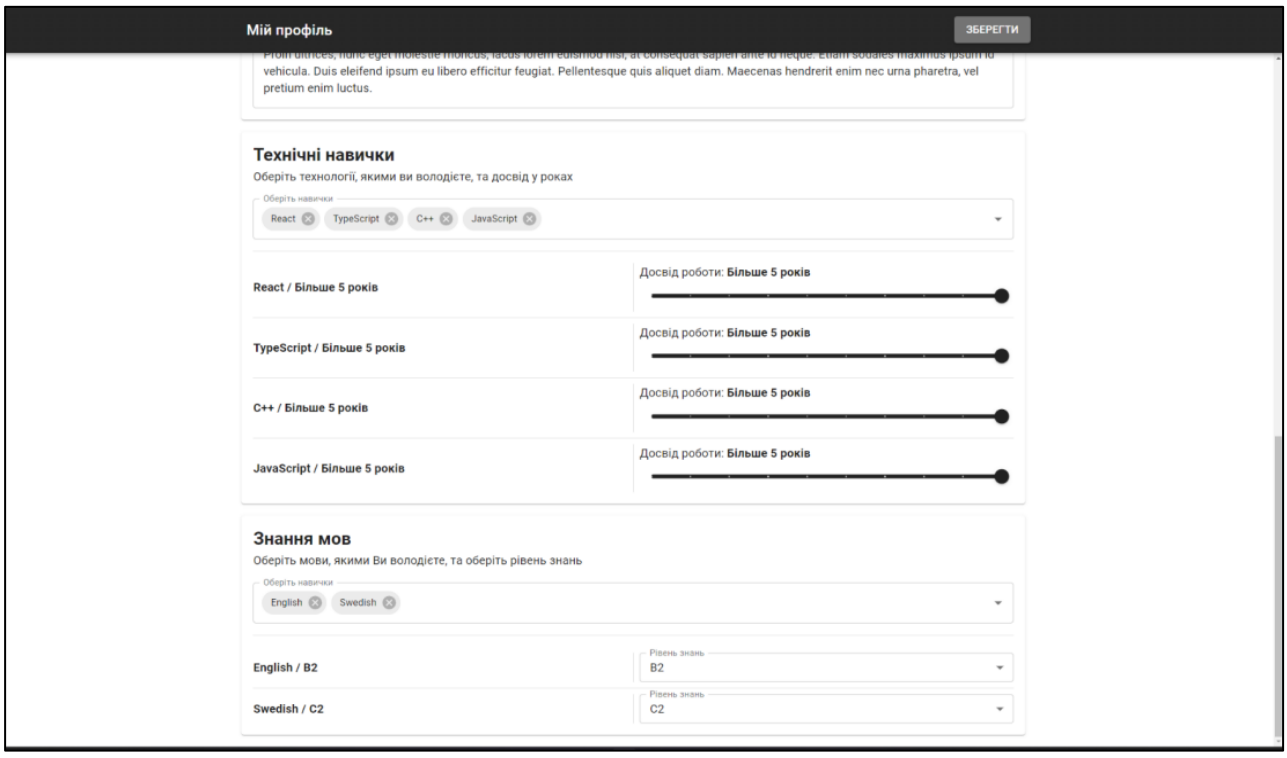



Рисунок 3.4 – Профіль Кандидата

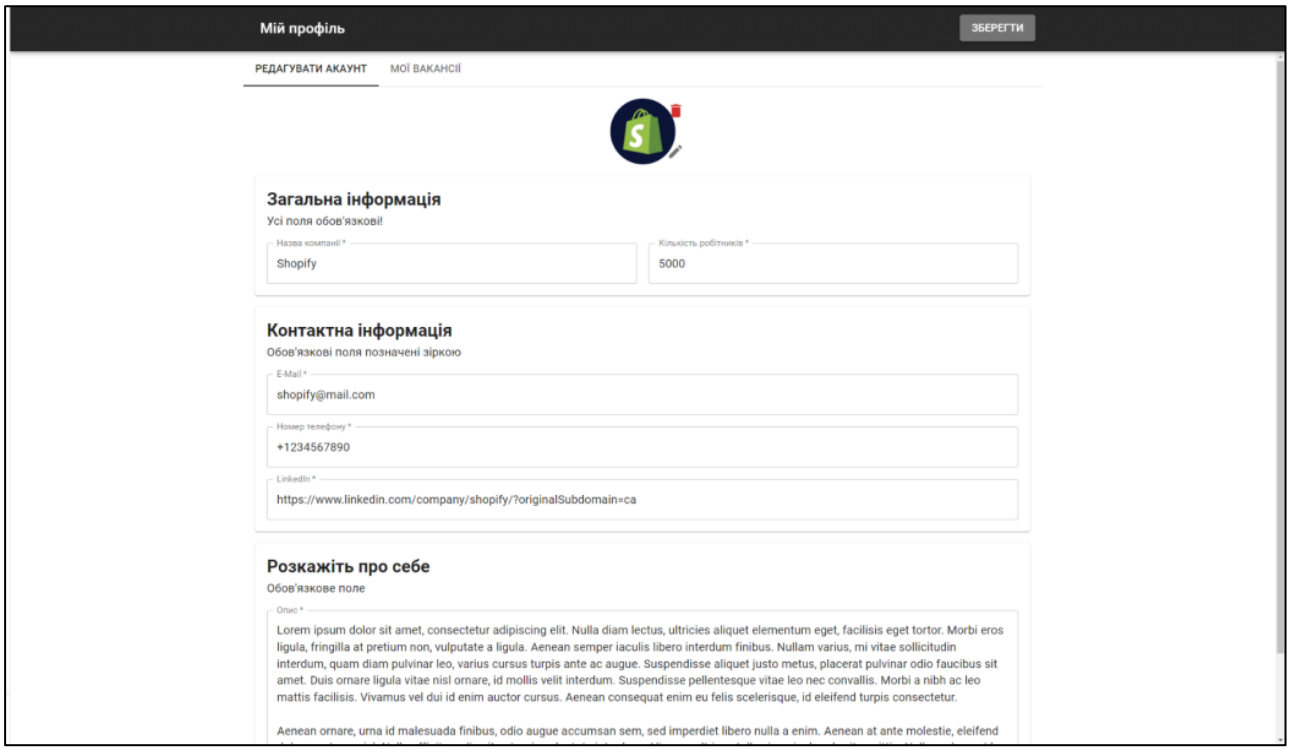



Рисунок 3.5 – Профіль рекрутера

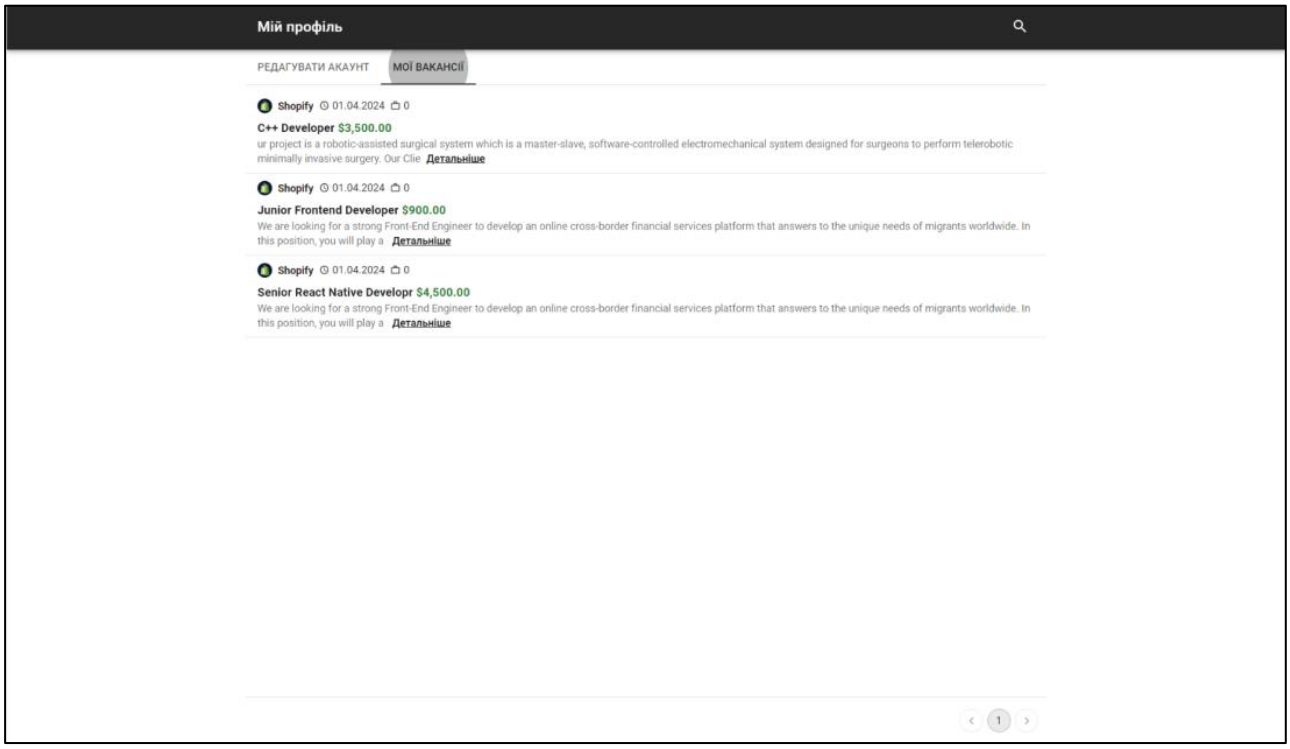



Рисунок 3.6 – Перегляд власних вакансій, створені компанією

Загалом, на цій сторінці користувач може вносити зміни щодо інформації про свій досвід роботи, або деталі компанії (якщо це Рекрутер). Для кандидата доступні такі поля для заповнення свого резюме (див. рис. 3.3 – 3.4):

- **П.І.Б**: прізвище, ім’я та по-батькові;
- **посада**: поточна посада кандидата;
- **E-Mail**: електронна пошта кандидата;
- **номер телефону**: номер телефону кандидата;
- **LinkedIn**: посилання на профіль кандидата на LinkedIn;
- **розкажіть про себе**: в цьому полі кандидат може розповісти про себе, свої навички, досвід та будь-яку іншу корисну інформацію;
- **технічні навички**: кандидат може обрати технології, якими він володіє, та вказати рівень володіння кожною з них – для оновлення цієї інформації, користувач може перетягнути повзунок для вибору рівня володіння кожною технологією;
- **знання мов**: кандидат може вказати мови, якими він володіє, та рівень володіння кожною мовою.

Після внесення змін користувач може натиснути кнопку «Зберегти», щоб застосувати зміни до свого профілю.

Інформація про рекрутера та компанію, яку вона представляє ідентичні, за виключенням того, що компанія може зазначити кількість працівників, які працюють там (див. рис. 3.5). До того ж, на цій сторінці, рекрутер може переглядати свої власні вакансії у розділі «Мої вакансії» (див. рис. 3.6).

### 3.6.6 Пошук вакансій

На рисунку 3.7 зображена сторінка для перегляду списку доступних вакансій.

На сторінці «Вакансії» кандидат може бачити переглядати список доступних вакансій з коротким описом, які релевантні до його навичок (що зазначені у профілі), та їх деталі. На цій сторінці, кандидату доступні такі можливості:

- **перегляд вакансій**: кандидат може переглядати список вакансій, які розміщені на платформі – кожна вакансія містить інформацію про назву

посади, компанію, дату публікації, заробітну плату та короткий опис вакансії;

– **детальна інформація**: кандидат може отримати детальну інформацію про вакансію, **натиснувши** на неї із списку – це може включати розширений опис вакансії, вимоги до кандидатів, обов'язки та іншу важливу інформацію.

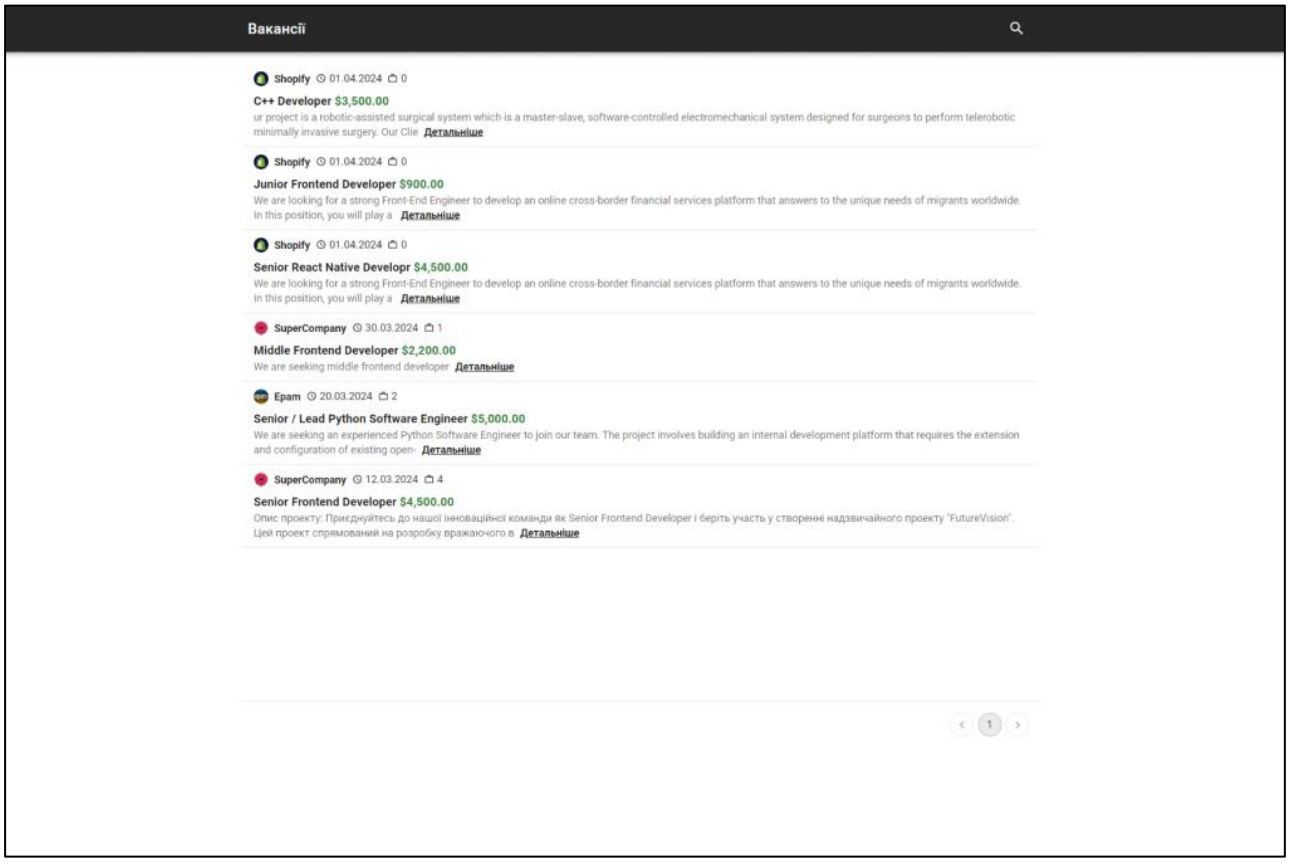



Рисунок 3.7 – Перегляд доступних вакансій

Іконка пошуку, яка розташована у верхньому правому куту сторінки (див. рис. 3.7), слугує для здійснення пошуку вакансій за ключовими словами, які містяться в назві вакансії.

Користувач може ввести слова або фрази, що відображають його інтереси або вимоги щодо робочого місця, і система виведе вакансії, які відповідають цим критеріям.

На рисунку 3.8 зображений процес пошуку вакансій за ключовим словом "Frontend".

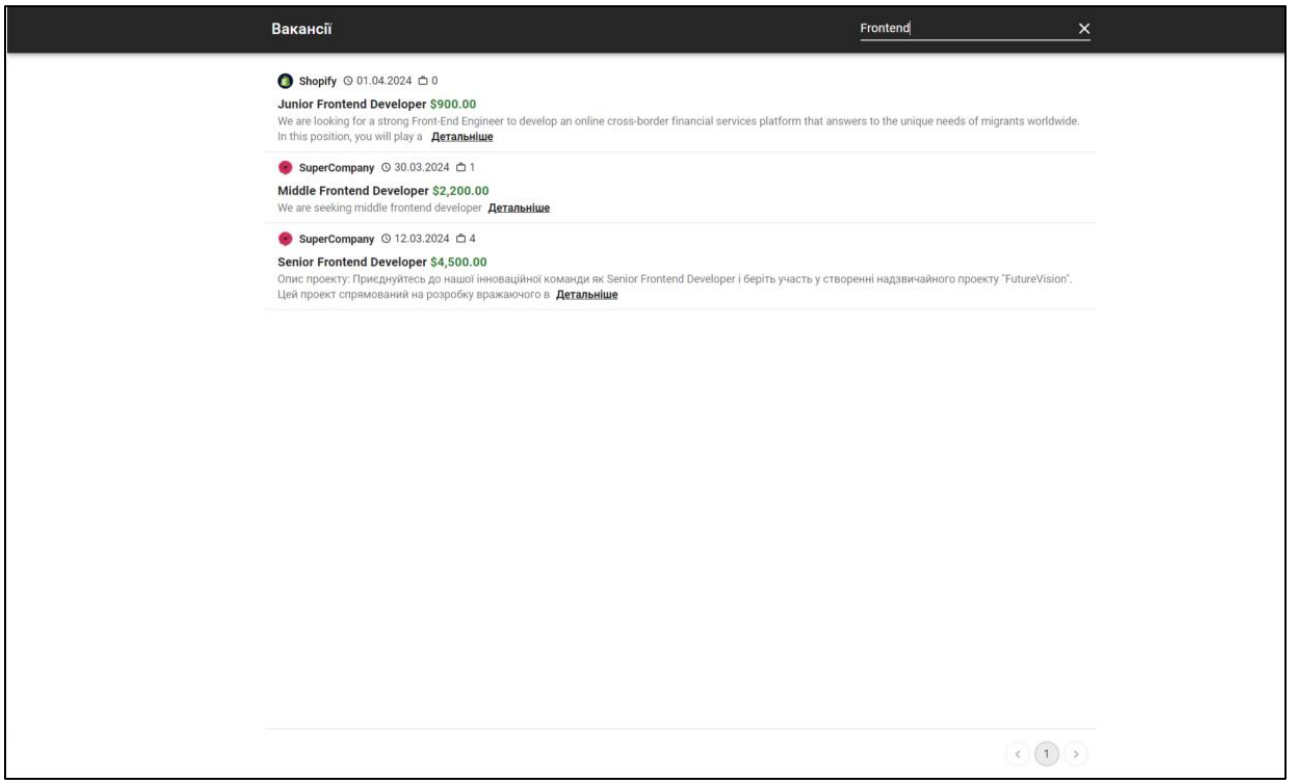



Рисунок 3.8 – Пошук вакансій за ключовим словом у назві посади

### 3.6.7 Перегляд вакансії

На рисунках 3.9 – 3.10 зображена сторінка з деталями про вакансію.

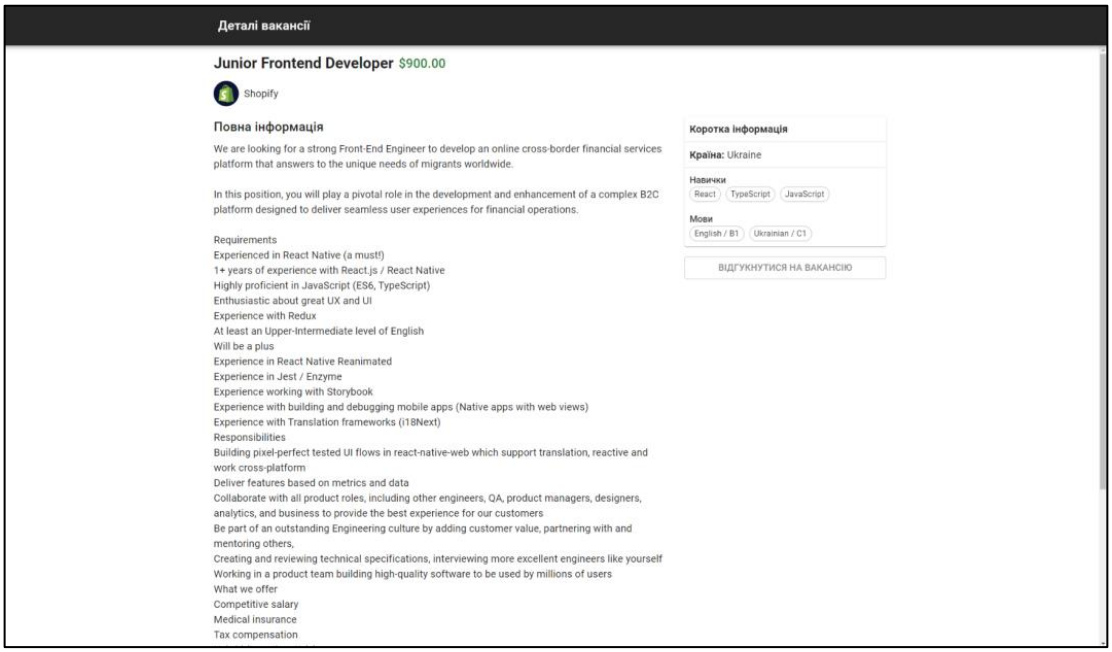



Рисунок 3.9 – Деталі вакансії

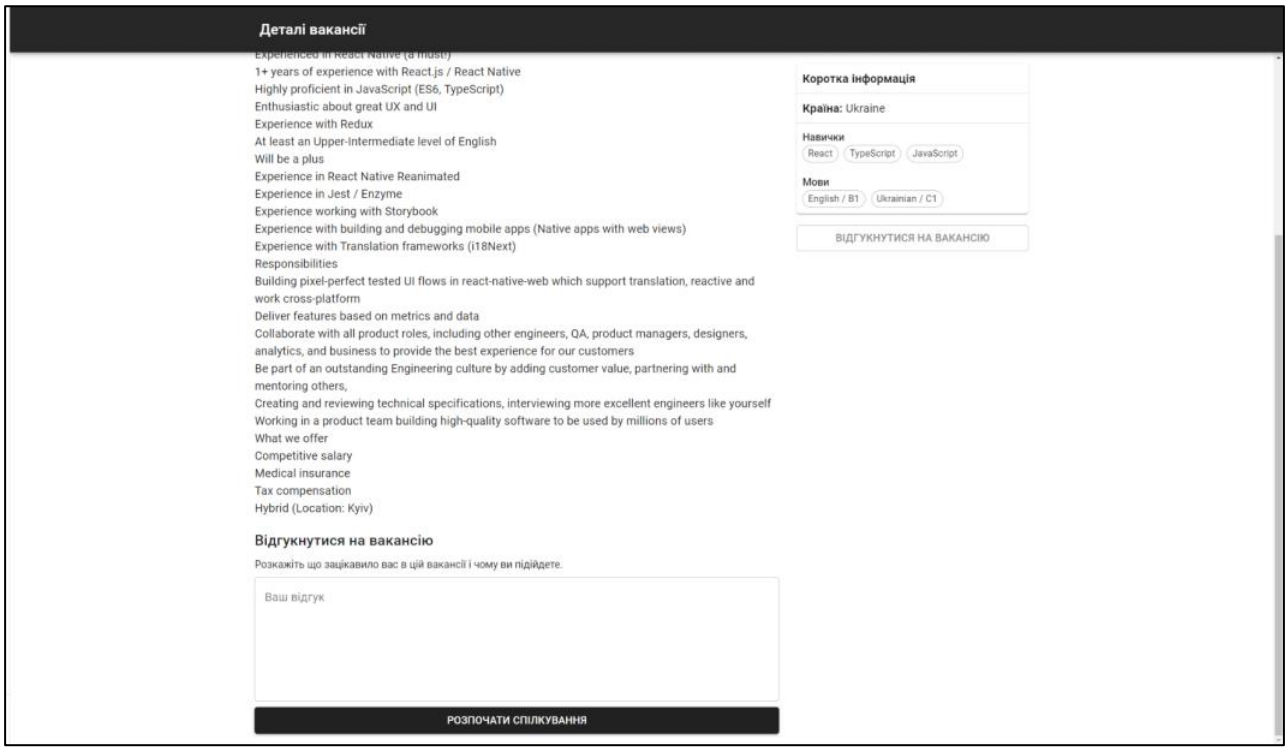



Рисунок 3.10 – Деталі Вакансії

Коли кандидат обирає конкретну вакансію зі списку на сторінці «Вакансії», відображається сторінка з деталями про цю вакансію. На цій сторінці кандидат може переглянути розширену інформацію про вакансію, таку як: опис вакансії, вимоги до кандидатів, обов’язки, які входитимуть до кандидата, та пропозиції щодо заробітної плати та переваги. Крім того, користувач може залишити свій відгук натиснувши відповідну кнопку «Відгукнутися на вакансію» та вказати інформацію про себе та свій досвід роботи з технологіями, які зазначені у вакансії. Після відправлення відгука, створюється окрема кімната з рекрутером, де Кандидат та Рекрутер можуть обговорити умови роботи яка прив’язана до вакансії, відгукнута Кандидатом.

На рисунку 3.11 зображено сповіщення, яке з’являється при успішному відгуку на вакансію.

На рисунку 3.12 зображено зовнішній вигляд деталей вакансії, якщо вона відкрита її власником.

Компанія, яка створила вакансію, може переглядати як опис цієї вакансії, так і список кандидатів, які відгукнулися на неї.

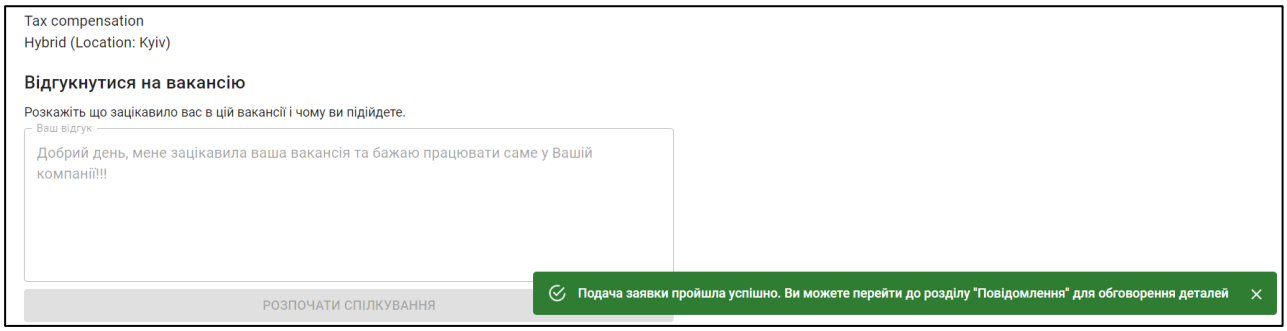



Рисунок 3.11 – Сповіщення про успішний відгук на вакансію

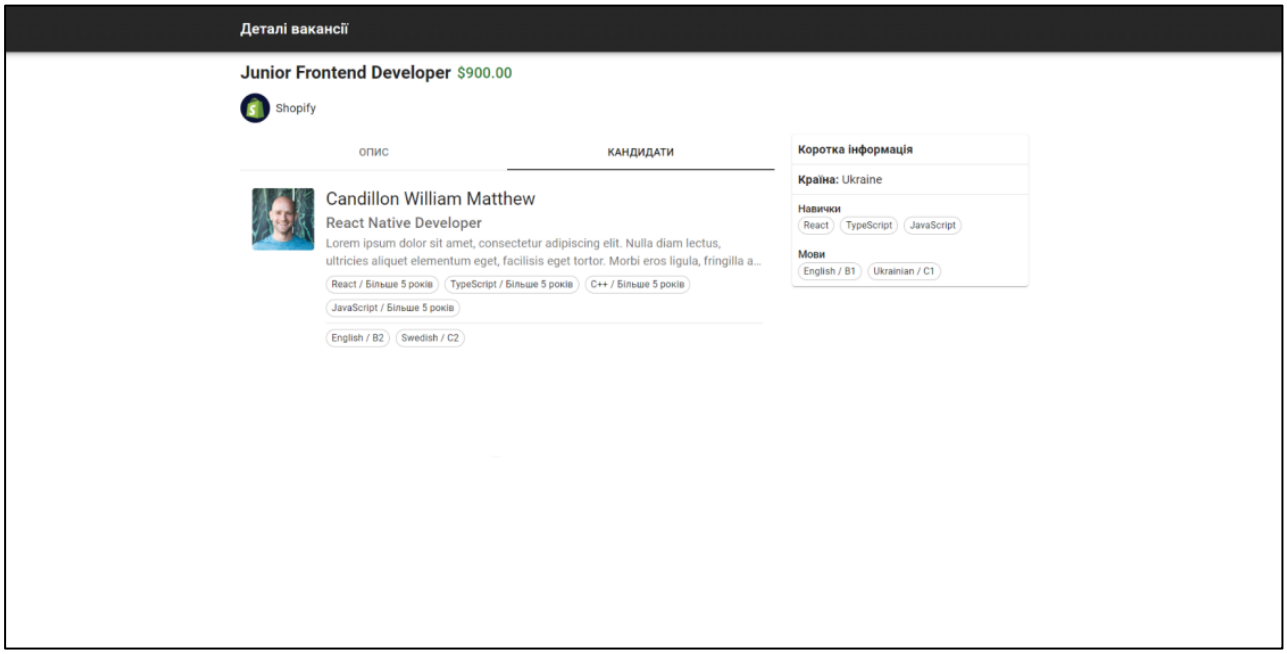



Рисунок 3.12 – Перегляд кандидатів, які відгукнулися на вакансію

### 3.6.8 Створення вакансії

На рисунках 3.13 – 3.14 зображений процес створення нової вакансії.

Ця сторінка дозволяє рекрутерам створювати нові вакансії на платформі. Вона містить кілька розділів і полів, які допомагають рекрутерам якнайточніше описати вакансію та вимоги до кандидатів (див. рис. 3.13 – 3.14).

Форма для створення вакансії, містить такі блоки:

– **загальна інформація**: в цьому розділі рекрутер повинен ввести загальну інформацію про вакансію, таку як назва посади, зарплата та країна;

– **розкажіть про проєкт та вимоги до кандидата**: тут рекрутер повинен надати опис проєкту та очікування до кандидата;
– **технічні навички**: у цьому розділі можна вибрати технічні навички, які кандидат повинен мати для цієї вакансії, а також вказати рівень володіння кожною з них;
– **знання мов**: тут можна вибрати мови, якими кандидат має володіти та вказати їх рівень.

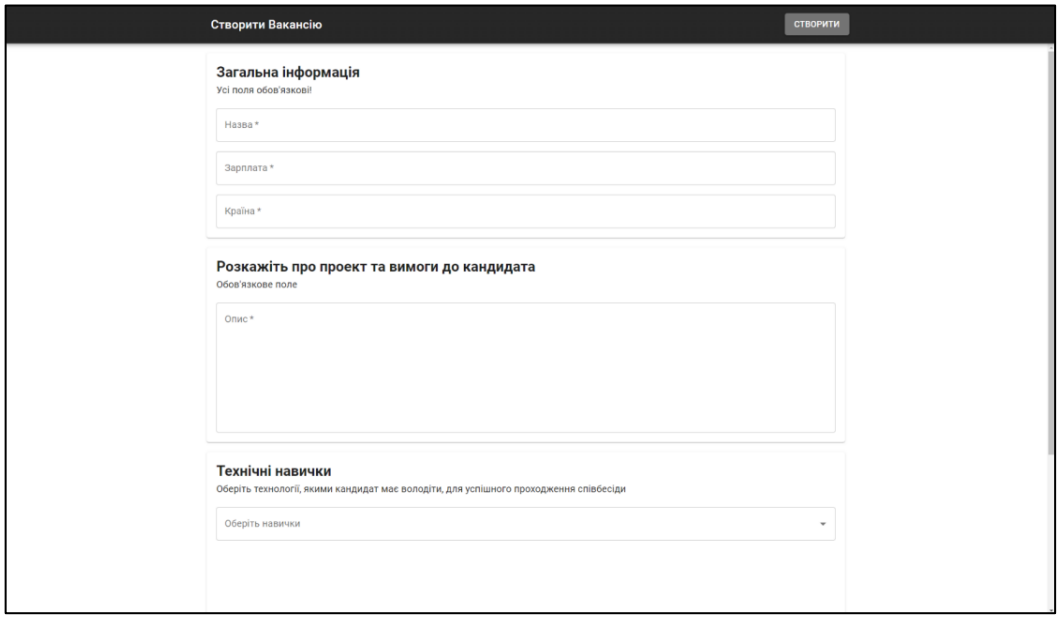



Рисунок 3.13 – Форма для створення вакансії

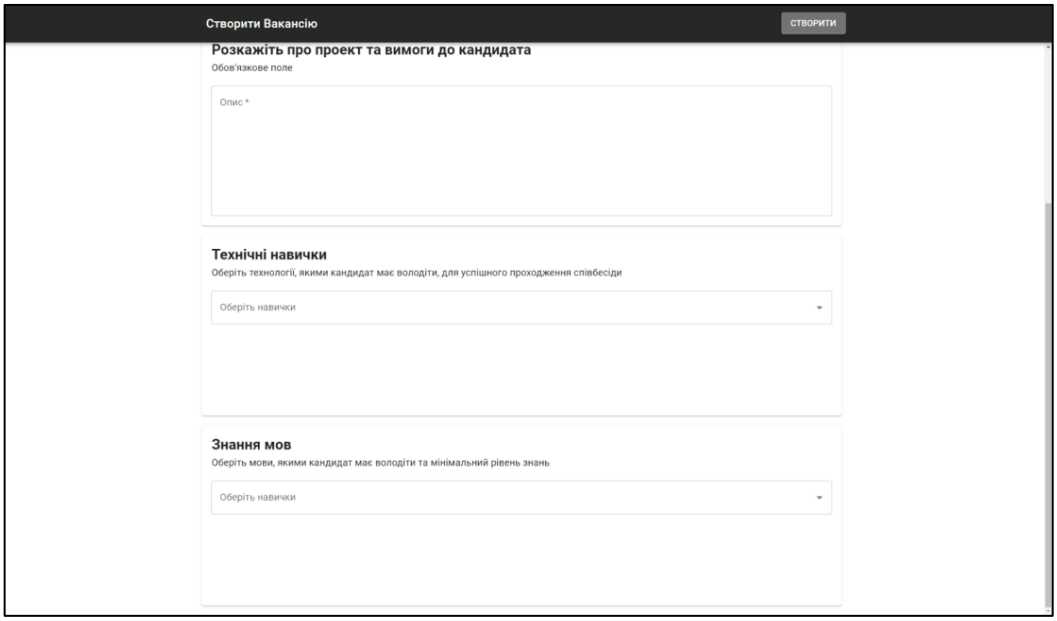



Рисунок 3.14 – Форма для створення вакансії

Кожне поле має важливе значення для правильного опису вакансії та визначення вимог до кандидатів.

Після заповнення всіх полів рекрутеру потрібно натиснути кнопку «Створити», щоб створити нову вакансію на платформі.

На рисунках 3.15 – 3.16 зображений приклад заповненої форми для створення вакансії.

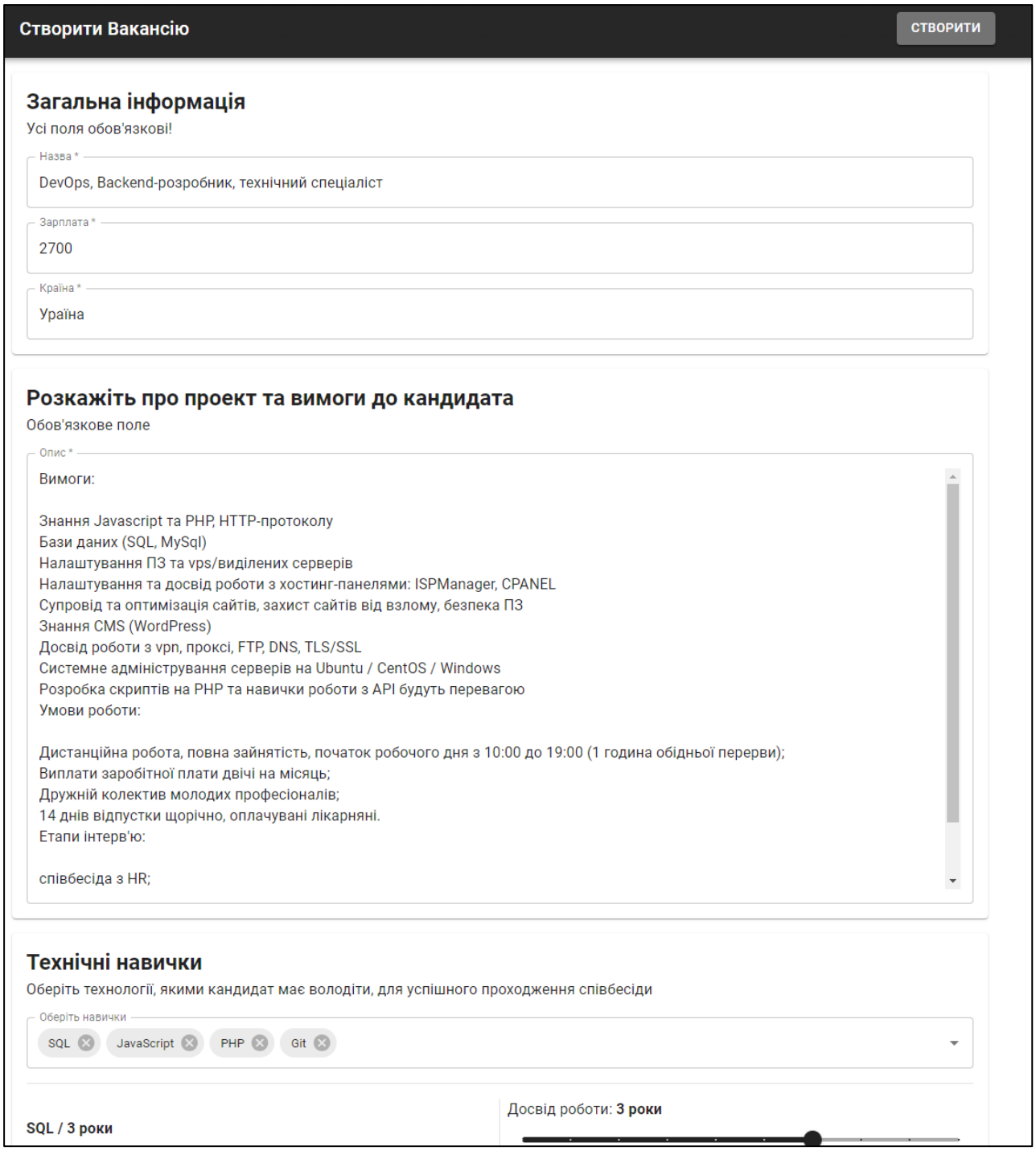



Рисунок 3.15 – Заповнена форма вакансії

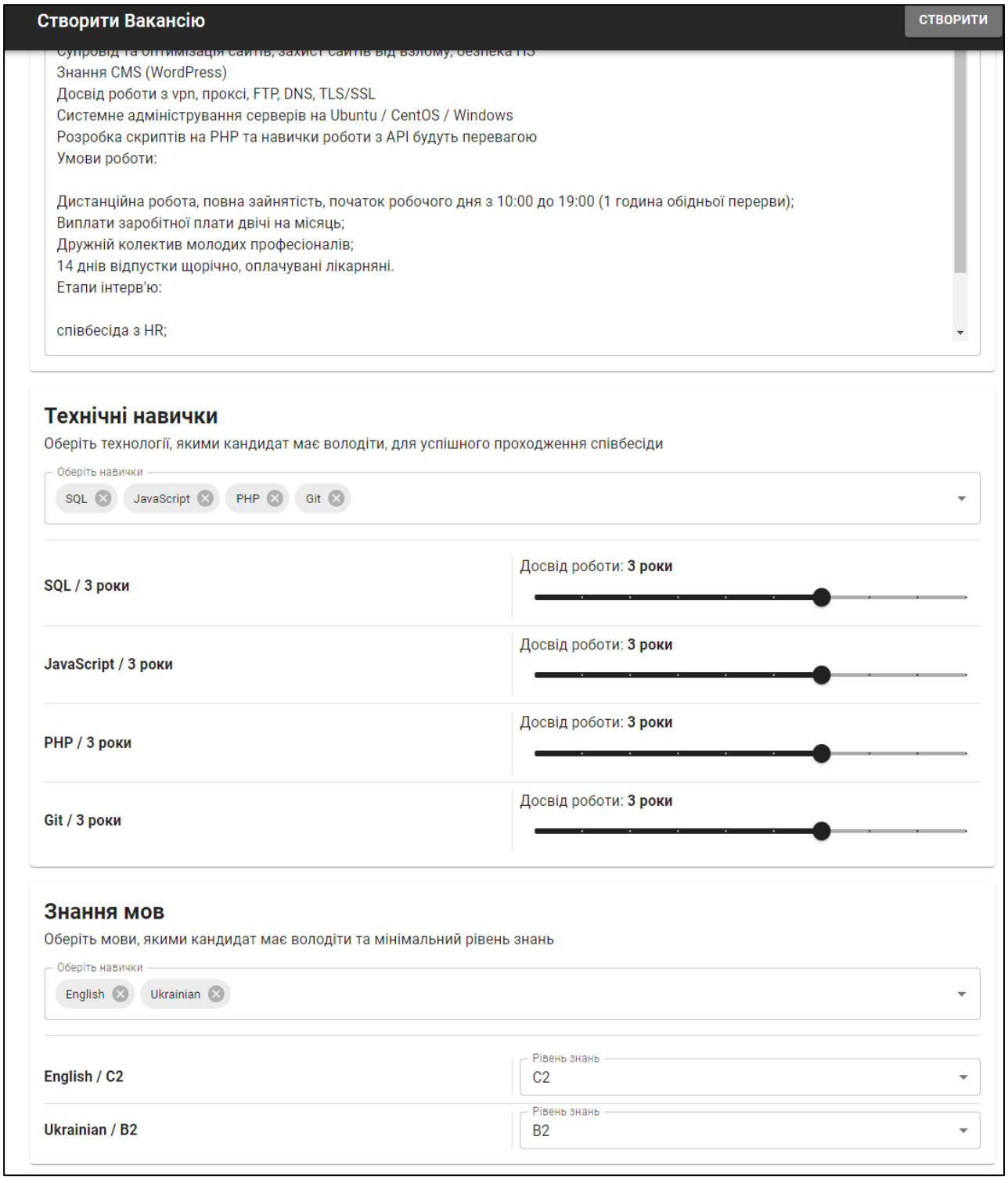



Рисунок 3.16 – Заповнена форма вакансії

### 3.6.9 Перегляд кандидатів

На цій сторінці рекрутер може переглядати профілі кандидатів, які зареєстровані на платформі. Вони можуть бачити фотографії кандидатів, їхні

імена, посада, та короткий опис про них. Крім того, рекрутер може переглядати додаткові дані про кандидатів, такі як досвід роботи та навички. За необхідності рекрутер може натиснути на профіль кандидата, щоб переглянути більше деталей.

На рисунку 3.17 зображена сторінка для перегляду кандидатів, які зареєстровані на цій платформі.

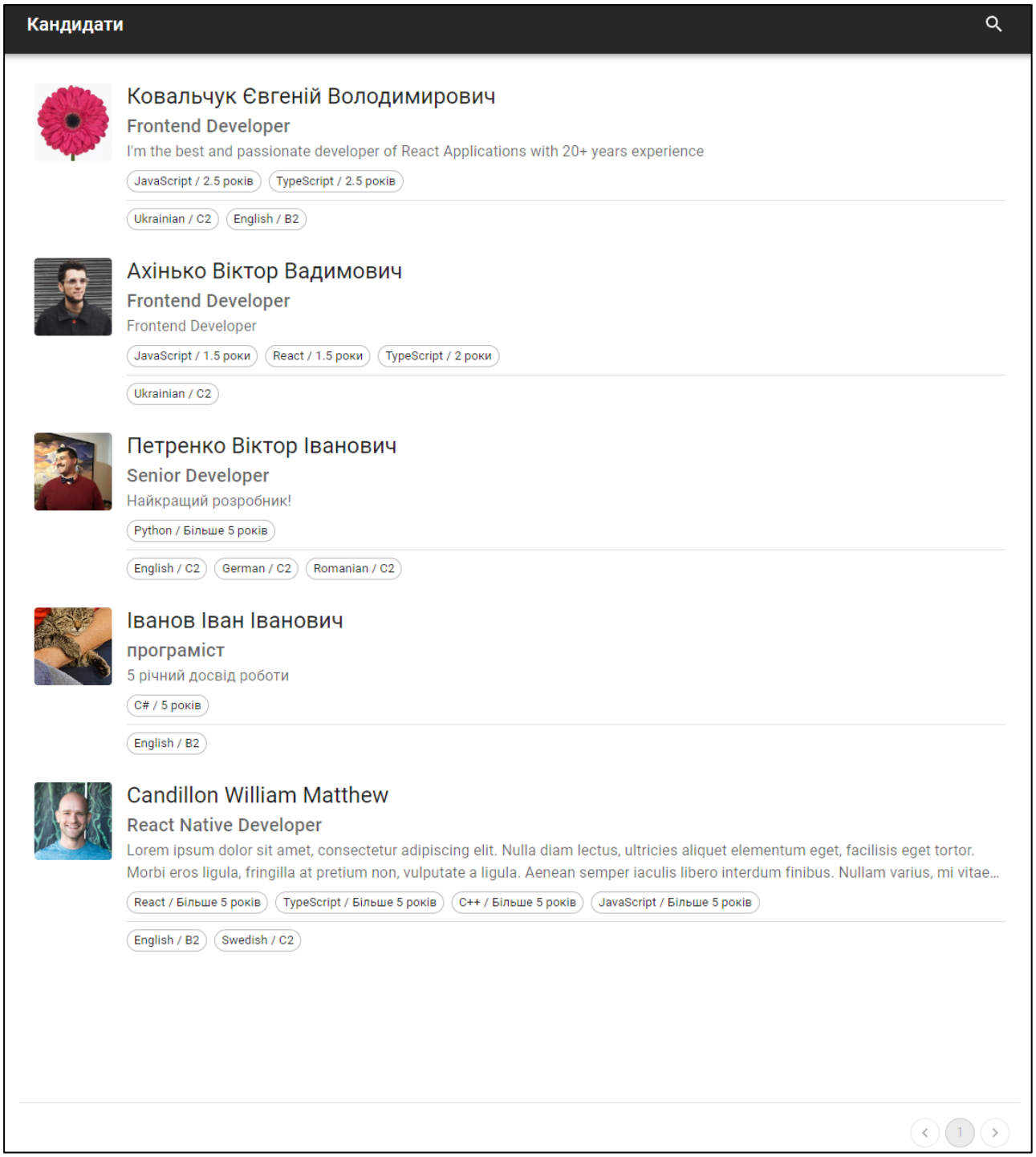



Рисунок 3.17 – Перегляд списку кандидатів

### 3.6.10 Перегляд анкети кандидата

На цій сторінці рекрутер може переглядати детальну інформацію про конкретного кандидата. Вони можуть побачити фотографію кандидата, його ім’я, посаду, телефонний номер та посилання на профіль LinkedIn, якщо вони вказані. Крім того, рекрутер може прочитати короткий опис про кандидата, який може включати інформацію про їхні навички, досвід роботи та особисті якості.

На рисунку 3.18 зображена детальна інформація про кандидата та процес відправлення пропозиції Рекрутером.

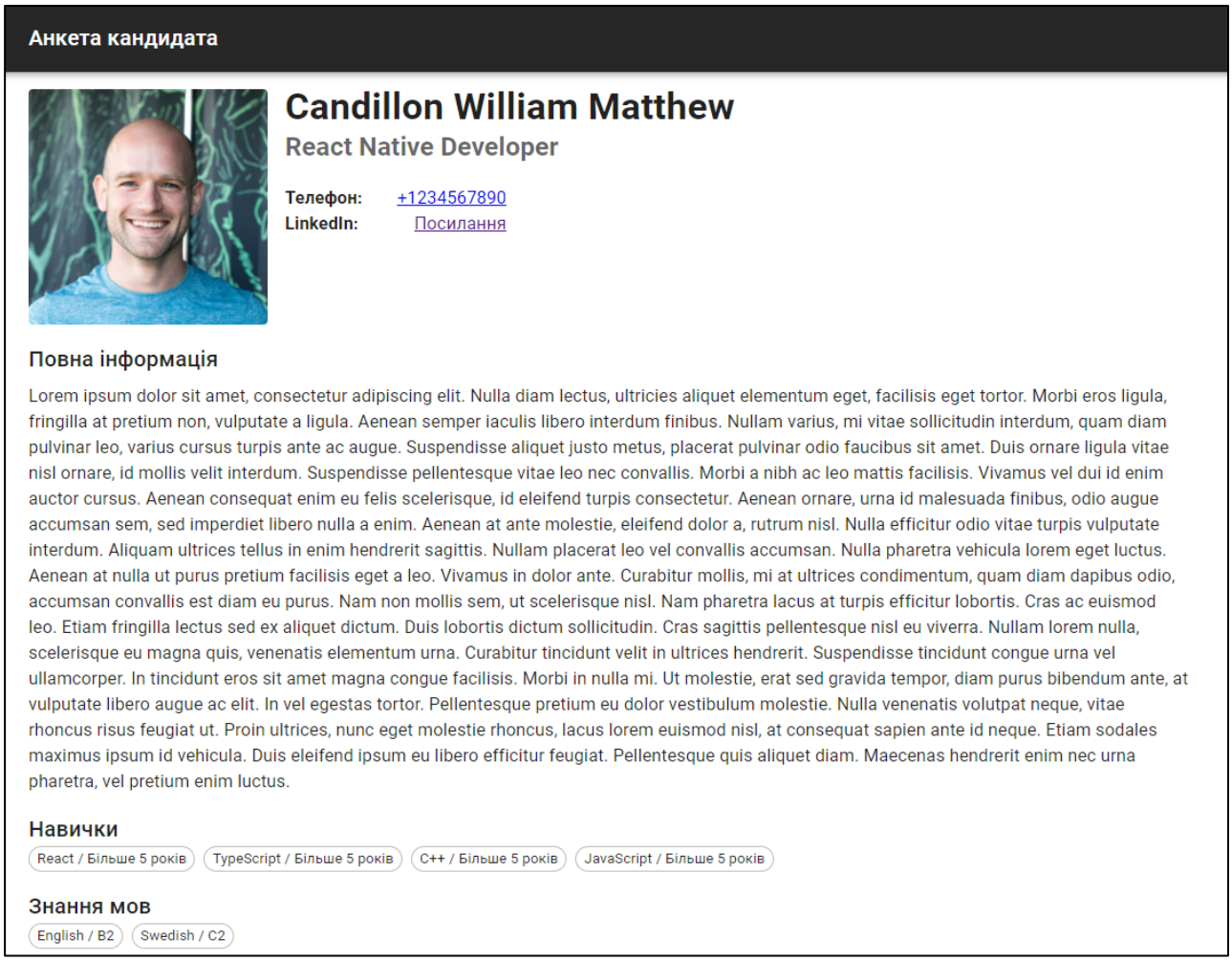



Рисунок 3.18 – Деталі анкети конкретного кандидата

Найбільш важливою функцією для рекрутера є можливість натиснути на кнопку «Написати» (див. рис. 3.18), щоб ініціювати контакт з кандидатом або надіслати запит на співбесіду.

### 3.6.11 Перегляд компаній

На цій сторінці представлений список компаній, які зареєстровані на платформі. Кожна компанія відображається зі своїм логотипом та коротким описом. Кандидати можуть переглядати цей список для ознайомлення з компаніями, які вакансії розмістили на платформі.

На рисунку 3.19 зображена сторінка зі списком доступних вакансій.

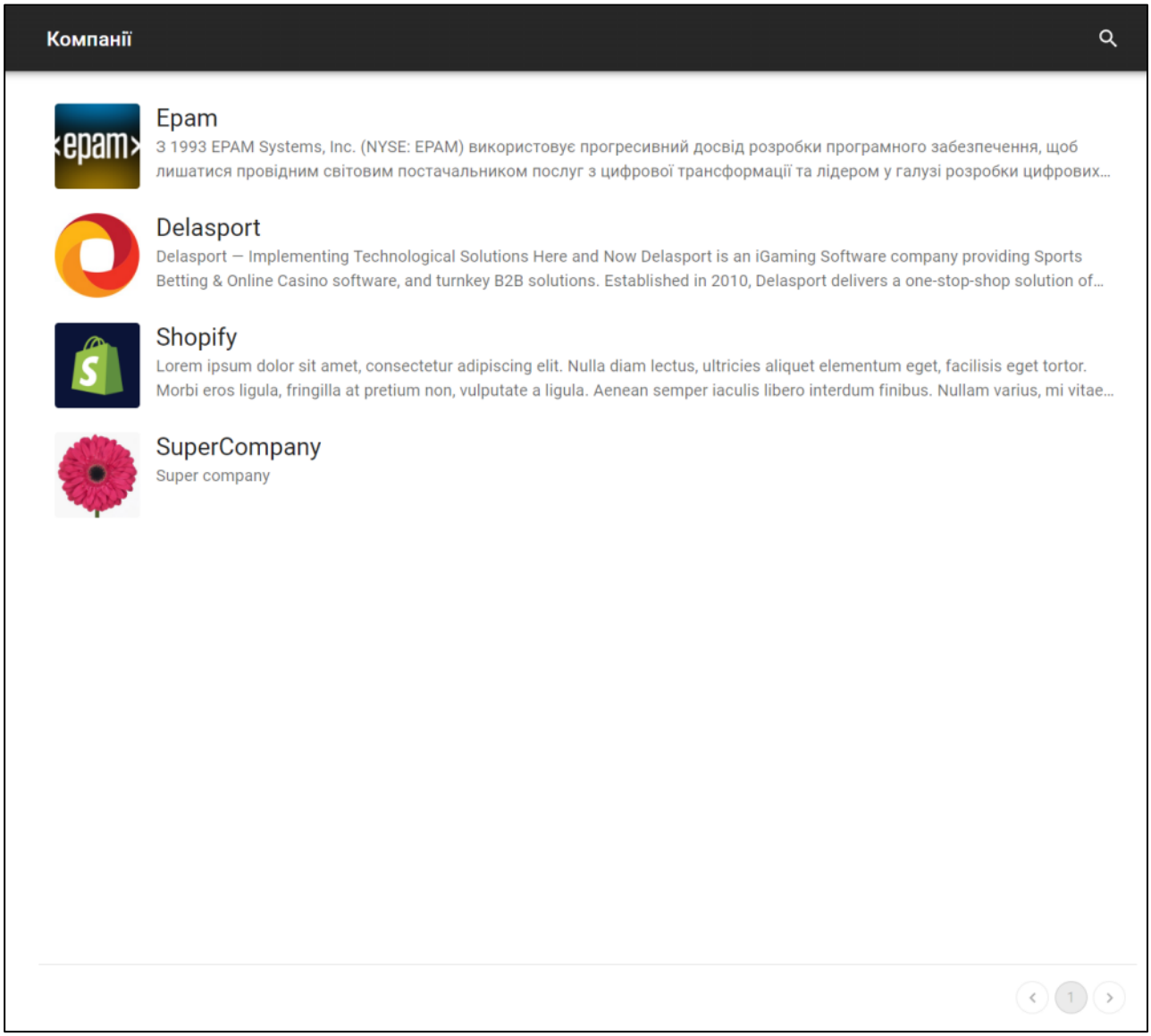



Рисунок 3.19 – Список доступних вакансій

Кожна компанія має власну сторінку з детальною інформацією, доступ до якої може бути наданий по кліку на логотип або назву компанії.

На рисунках 3.20 – 3.21 зображена сторінка з детальною інформацію про компанію, та її вакансіями.

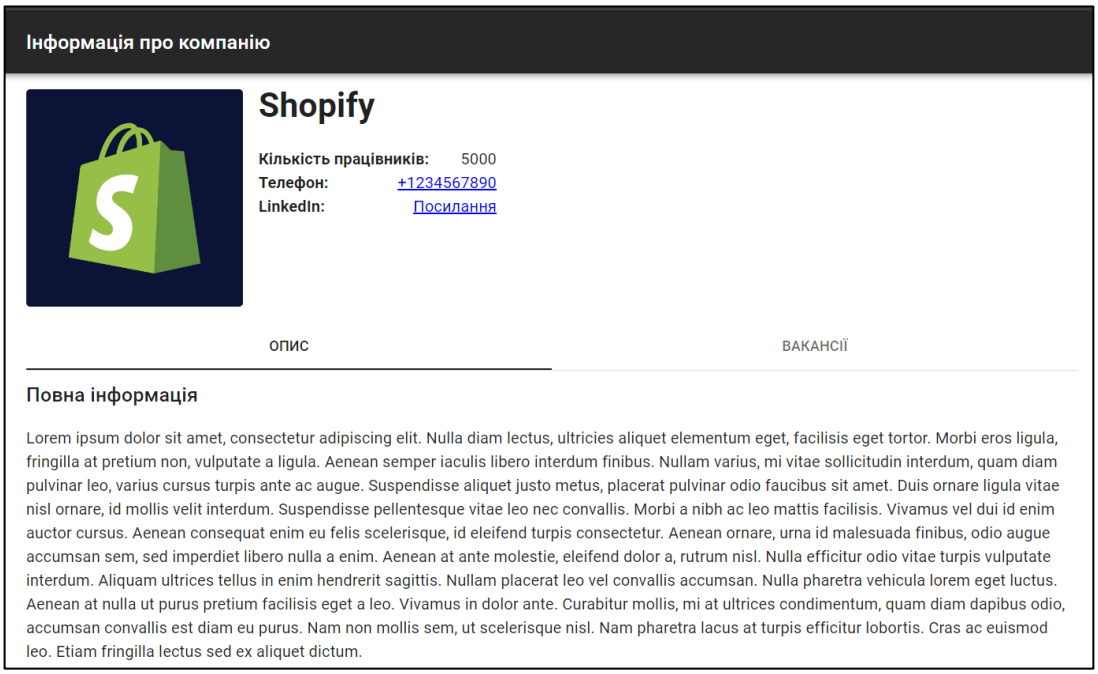



Рисунок 3.20 – Детальна інформація про компанію

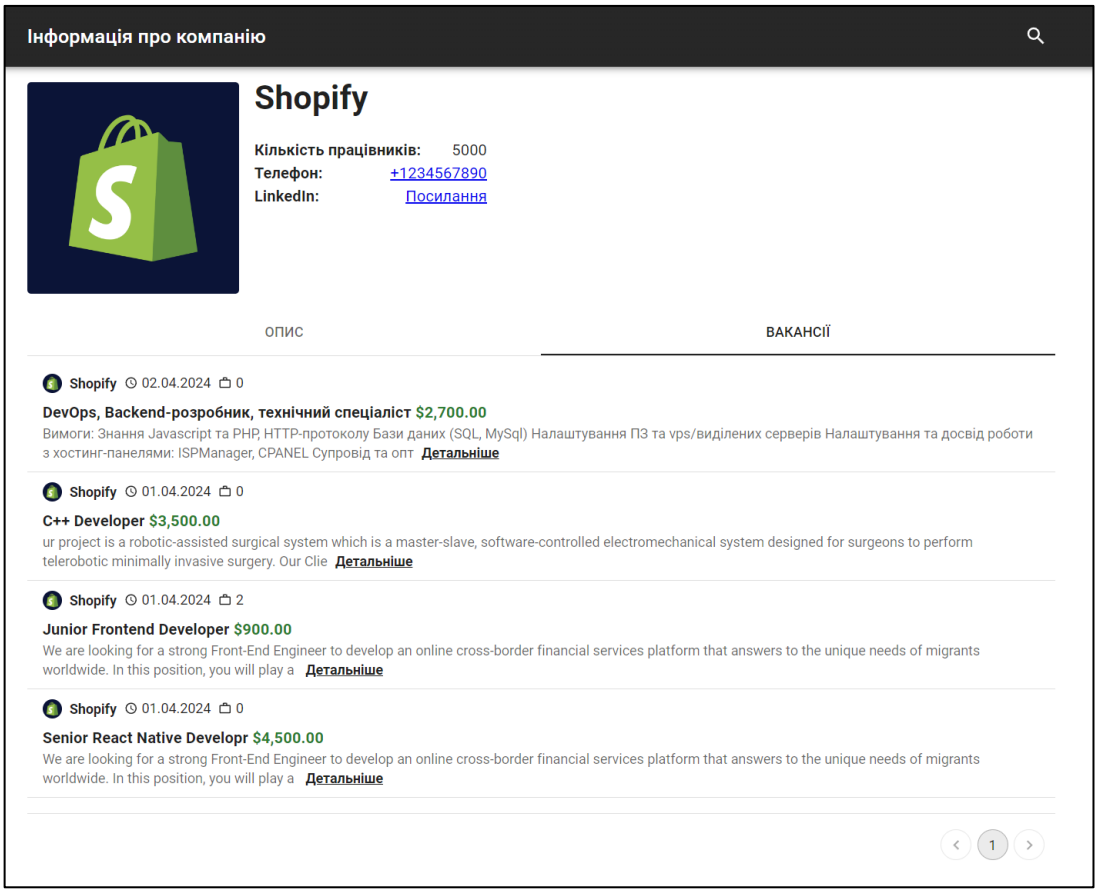



Рисунок 3.21 – Список вакансій конкретної компанії

# ВИСНОВКИ

В результаті роботи було написано технічне завдання на розробку системи IT рекрутингу. Для створення цієї системи були обрані такі технології:

- .NET та Visual Studio для розробки back end;
- C# та ASP .NET Core для реалізації API;
- MS SQL DB для зберігання даних;
- Entity Framework для доступу к базі даних;
- React для розробки front end;
- Feature Sliced Design (FSD).

Згідно з поставленими завданнями, були виконані такі етапи створення системи:

- сформульовані вимоги до системи (функціональні та нефункціональні (інтерфейс, кросбраузерність, безпека, продуктивність); також проведено огляд предметної області та інструментів розробки;
- спроєктована та побудована структура системи (розроблені діаграми прецедентів, діяльності та послідовності; надано детальний опис прецедентів);
- реалізована система рекрутингу IT фахівців;
- надано інструкцію по створенню компонентів системи, керівництво користувача та структуру проєкту;
- протестована робота системи.

# ПЕРЕЛІК ПОСИЛАНЬ

1. JSON Web Token Authentication. URL: https://jwt-auth.readthedocs.io/en/develop/ (дата звернення: 01.06.2024).

2. Офіційна документація ASP.NET Core. URL: https://docs.microsoft.com/en-us/aspnet/core/ (дата звернення: 05.06.2024).

3. ASP.NET Core GitHub. URL: https://github.com/dotnet/aspnetcore (дата звернення: 06.09.2024).

4. Freeman A. Pro ASP.NET Core MVC 2. Apress, 2017. 1046 p.

5. Офіційна документація Microsoft SQL Server. URL: https://docs.microsoft.com/en-us/sql/sql-server/ (дата звернення: 20.06.2024).

6. SQL Server Central. URL: https://www.sqlservercentral.com/ (дата звернення: 21.06.2024).

7. Petkovic D. Microsoft SQL Server 2019: A Beginner’s Guide : 7th ed. McGraw Hill, 2020. 864 p.

8. Hansen G. W., Hansen J. V. Database Management and Design : 2nd ed. Prentice Hall, 1985. 582 p.

9. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software : 1st ed. Addison-Wesley Professional, 2003. 560 p.

10. Портал для .NET розробників. URL: https://metanit.com/ (дата звернення: 05.07.2024).

11. Stefanov S. React: Up & Running: Building Web Applications : 2nd ed. O’Reilly Media, 2021. 230 p.

12. Chinnathambi K. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux : 2nd ed. Addison-Wesley Professional, 2018. 304 p.

13. Accomazzo A., Murray N., Lerner A. Fullstack React: The Complete Guide to ReactJS and Friends. Kindle Store, 2019. 1214 p.

14. Bertoli M. React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns. Packt Publishing, 2017. 318 p.