Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА SPA ДЛЯ ПОШУКУ РОБОТИ В IT СФЕРІ З ВИКОРИСТАННЯМ ANGULAR, .NET ТА REDIS DB

DEVELOPMENT OF A SPA FOR JOB SEARCH IN THE IT FIELD USING ANGULAR, .NET, AND REDIS DB

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-42
Литвинюк Максим Віталійович

Керівник:
Самчук Людмила Михайлівна

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення*»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Литвинюку Максиму Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка SPA для пошуку роботи в IT сфері з використанням Angular, .NET та Redis DB»*
Керівник роботи: *к.т.н., доцент Самчук Л. М.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *платформа створення дизайну Figma, UML діаграми, фреймворк .NET, фреймворк Angular, СУБД Redis DB, IntelliJ IDEA Rider, Umlet методичні вказівки до виконання кваліфікаційної роботи бакалавра*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *надати інтуїтивну платформу для пошуку роботи, надати можливість публікувати вакансії, забезпечити безпеку даних користувачів, впровадити легкі шляхи взаємодії з платформою*
5. Перелік графічного матеріалу: *9 таблиць, 11 рисунків, 2 додатки*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Самчук Л. М.*		
Специфікація вимог до розробленої системи	*Самчук Л. М.*		
Розробка об'єкта проектування	*Самчук Л. М.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	2,07 %		
Академічна доброчесність	*Самчук Л. М.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	виконано
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	виконано
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	виконано
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	виконано
5	Практична реалізація об'єкта проектування та розробка бази даних	до 26.04.2025 р.	виконано
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	виконано
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	виконано

Здобувач вищої освіти _____ Максим ЛИТВИНЮК

Керівник кваліфікаційної роботи _____ Людмила САМЧУК

АНОТАЦІЯ

Литвинюк М. В. Розробка SPA для пошуку роботи в IT сфері з використанням Angular, .NET та Redis DB. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проаналізовано наявні платформи з пошуку роботи та виявлено їхні ключові проблеми та недоліки.

На основі цього аналізу сформульовано завдання на кваліфікаційну роботу бакалавра.

У другому розділі визначено функціональні та нефункціональні вимоги до майбутньої платформи, створено UML-діаграми варіантів використання, карти сайту та дизайн основних сторінок. Також обґрунтовано вибір технологій: ASP.NET, Angular та Redis DB.

У третьому розділі описано практичну реалізацію повнофункціонального вебзастосунку. Реалізовано серверну та клієнтську частини, налаштовано взаємодію з базою даних, систему аутентифікації та авторизації, захист даних, а також розроблено інтерфейси для користувачів різних ролей та описані підходи, щодо оптимізації продукту для пошукових систем. Проведено тестування основних модулів і підготовку системи до розгортання.

У висновках підсумовано виконану роботу, підтверджено досягнення поставлених цілей та зазначено перспективи подальшого розвитку платформи.

Ключові слова: пошук роботи, вебплатформа, ASP.NET, Angular, Redis DB, безпека, дизайн, розробка програмного забезпечення.

ABSTRACT

Lytvyniuk M. Development of a SPA for Job Search in the IT Field Using Angular, .NET and Redis DB. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, a list of references and appendices.

The first chapter analyzes current job search platforms and identifies key issues such as overloaded interfaces, security flaws, lack of communication tools, and the use of inappropriate technologies. Based on this analysis, the requirements for the new system were defined.

The second chapter outlines the functional and non-functional requirements for the platform, presents UML use case diagrams, site maps, and the design of key pages. The choice of technologies is justified: ASP.NET, Angular and Redis DB.

The third chapter describes the practical implementation of a fully functional web application. Both server-side and client-side components were developed, including database interaction, authentication and authorization systems, data protection, and user interfaces for different roles, and described approaches of product optimization for search engines. Core modules were tested, and the system was prepared for deployment.

The conclusions summarize the completed work, confirm the achievement of the project goals, and outline prospects for future development.

Keywords: job search, web platform, ASP.NET, Angular, Redis DB, security, design, software development.

ЗМІСТ

ВСТУП

Актуальність теми. Проблема пошуку кваліфікованих працівників існувала завжди. Однією з причин цієї проблеми є необізнаність спеціалістів про наявні вакансії які пропонують компанії. Якщо раніше потенційні співробітники дізнавались про можливість влаштуватись у компанію через сарафанне радіо, оголошення в газетах, публічних місцях, або через місцеве телебачення, то зараз для пошуку працівників компанії використовують Інтернет.

Інформація в Інтернеті розміщується у вигляді вебсайтів, аби користувачі могли зручно переглянути її, що означає що кожна компанія, що шукає працівників, повинна створити власну сторінку та публікувати актуальні вакансії на ній. Цей спосіб все ще використовується, та його актуальність стрімко падає, адже користувач повинен цілenаправлено шукати компанії та їх ресурси аби переглянути список вакансій.

Враховуючи, що IT сфера надає найбільше можливостей для віддаленої роботи, кількість вакансій на одну особу є кратно більшою аніж в інших сферах, адже доступні вакансії не лише локальних компаній, а й віддалених. Через це популярними є сайти, що спеціалізуються на пошуку роботи в IT сфері.

Багато сучасних платформ, що пропонують пошук роботи в IT містять сторонні послуги, на яких часто і робиться акцент, а послуга пошуку роботи залишається другорядною та малопомітною. Існують варіанти де пошук роботи є основним або й єдиним напрямком сайту, але пропонуються вакансії не лише з IT сфери, а й інших. Такі сайти не враховують особливості IT сфери, а тому рекрутери часто нехтують ними, що проявляється у малій кількості вакансій та складнощах у пошуку через відсутність фільтрів пристосованих до IT вакансій. Як рішення створюються сайти які пропонують лише IT вакансії, і вони є найкращими варіантами для пошуку роботи в IT сфері.

Подібні сайти мають низку проблем. Дуже часто ці сайти перевантажені інформацією, через що пошук ускладнюється, так як алгоритми пошуку починають працювати неправильно, тому що не враховують усі параметри які

накопичуються разом з інформацією. Часто інтерфейс є незручним, адже важко знайти баланс між швидкістю перегляду вакансій та зручним представленням інформації. Багато подібних сайтів мають профілі користувачів, які потрібно довго налаштовувати, а часто користувачі користуються багатьма сайтами одночасно, і їм потрібно щоразу заповнювати профіль. Якщо потрібно оновити інформацію у профілі, користувач змушений згадати усі сайти на яких реєструвався, та змінити на кожному інформацію про себе.

Метою даної кваліфікаційної роботи є розробка нової платформи пошуку роботи для IT фахівців, що спрямована на вирішення більшості цих проблем шляхом аналізу помилок та досвіду конкурентів.

Об’єктом розробки є повноцінна платформа з пошуку роботи для IT фахівців.

Предметом кваліфікаційної роботи є функціональні та архітектурні особливості розробки платформи «DevJobster» із застосуванням .NET, Angular та Redis DB.

Завдання на кваліфікаційну роботу:

– проаналізувати предметну область та розробки у даній сфері;

– визначити вимоги та спроєктувати програмне забезпечення використовуючи UML діаграми та створити дизайни сторінок;

– обрати засоби та методи створення програмного забезпечення;

– розробити програмне забезпечення відповідно до спроєктованих діаграм та дизайну та обраних засобів і методів розробки;

– написати автоматизовані тести, протестувати в ручному режимі та налагодити розроблену систему.

Апробація роботи. З метою апробації матеріалів роботи підготовлено тези, які були подані до збірників конференцій та були представлені на Міжнародній науково-практичній конференції «Цифрова трансформація: виклики та стратегії», Луцьк, 25 лютого 2025 року (додаток А), та на Міжнародній науково-практичній конференції «Інформаційні технології в освіті, науці і виробництві», Луцьк, 23-24 травня 2025 року (додаток Б).

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Аналіз сучасного стану проблем розпочнемо з проєкту розробленого командою DATAFLAIR TEAM [12]. Розробники платформи пропонують додати реєстрацію лише для рекрутерів, уникаючи потреби в розробці логіки для реєстрації користувачів. Це рішення допоможе зберегти ресурси, адже не потрібно буде зберігати дані користувачів, натомість, шукачі будуть надавати інформацію при кожній відправці заявки на оголошення рекрутера, що, все-таки, має більше недоліків аніж переваг. На перший погляд, може здатись, що відсутність акаунту спрощує шукачам пошук та подання заявки, адже не потрібно згадувати свої дані для входу та перейматись їх безпекою. Популярні браузери давно мають можливість збереження реєстраційних даних для сайтів, отже існують сторонні способи вирішення цієї проблеми, яку намагались вирішити розробники. З недоліків, відсутність реєстрації означає що боти та зловмисники зможуть активніше намагатись нанести шкоди, адже їм не потрібно буде щоразу реєструватись для використання платформи. Відсутність акаунту ускладнює блокування користувача за порушення правил платформи та ускладнює збір статистичних даних на основі яких можна створювати корисні графіки для користувачів та покращувати платформу. Також платформа не надає можливості для спілкування рекрутера та шукача, що в рази ускладнює роботу рекрутера, адже для зв’язку з шукачем на рахунок невеликих розмов (наприклад, для обговорення часу та способу проведення співбесіди) доведеться використовувати надані контакти, а це зазвичай соціальні мережі, номер телефону або пошта, і рекрутеру доведеться витрачати додатковий час на відкриття та пошук людини у соціальній мережі або пошті, а також не гарантовано що людина відповість там, адже ці методи зв’язку не пов’язані з платформою пошуку роботи напряму. Також платформа використовує вбудовану базу даних, яка не підходить для подібного

проєкту адже не має потужної системи захисту, має обмеження в продуктивності порівняно з іншими базами даних та має проблеми з масштабованістю.

Ще одним варіантом є проєкт розроблений Md Imran Hossain з університету Daffodil Institute of IT, Бангладеш [10]. Першим що кидається в очі при аналізі, це наявність функціональної вимоги яка стосується графіків проведення інтерв'ю. Платформа не надає можливості для проведення інтерв'ю, тому незрозуміло яким чином графік інтерв'ю стосується платформи що надає послуги з пошуку роботи, особливо коли рекрутери мають власні, звичні способи створення графіків інтерв'ю. Також адаптація під мобільні пристрої перерахована у секції планів на майбутнє, що свідчить про випуск продукту, неготового до використання у сучасних реаліях, адже більшість людей користуються Інтернетом через смартфон [9], тому запуск проєкту без можливості зручного перегляду через смартфон є великою помилкою, яка призведе до зниження популярності платформи.

Наступним розглянутим проєктом є «Online Job Portal» [4]. При проєктуванні таблиці бази даних для рекрутера була допущена груба помилка з точки зору захисту даних, а саме, збереження паролю, контрольного питання та відповіді до нього в одній таблиці разом з інформацією про акаунт рекрутера та дані компанії. Також помилкою є об'єднання даних компанії і рекрутера в одну таблицю, адже якщо декілька рекрутерів з однієї компанії зареєструються на сайті, їм доведеться дублювати інформацію про компанію. Точно такі самі проблеми спостерігаються з таблицею шукача роботи, де поля які стосуються безпеки акаунту об'єднані в одну таблицю разом з загальною інформацією, а також внесені поля які призведуть до дублювання інформації. Ці проблеми легко вирішуються застосуванням зв'язків таблиць одна до багатьох та багато до багатьох. Також використанням зв'язків можна вирішити проблему з таблицею яка містить інформацію про опубліковану роботу, адже ця таблиця має інформацію як щодо роботи, так і щодо публікації, які є різними об'єктами, адже публікація містить інформацію щодо роботи, тому цю таблицю було б добре розділити на дві частини та пов'язати зв'язком багато до багатьох.

Проєкт «Student Job Portal» розроблений Md Manwar Sayeem [1] загалом дотримується принципів правильної розробки веб додатків, але також не є ідеальним. Інтерфейс додатку є неоднорідним, особливо в плані кольорів. Верхня частина сайту змінює свій колір, так само, як і кнопки сайту, що сплутує користувачів, адже колір використовується для розпізнавання елементів, і, скажімо, червона кнопка символізує небезпечну, деструктивну або незворотню дію, в той час як зелена закликає виконати дію. Чомусь проєкт використовує червоний колір для кнопки що закликає відгукнутись на вакансію, зелену для входу/реєстрації та синю для дії додавання. Також адміністратору сайту надана можливість зміни розташування елементів на сайті, що тільки ускладнить користування сайтом. Для фону використовуються три кольори, чорний, синій та білий, що може викликати враження неякісного сайту який складений з декількох інших.

Наступний проєкт є розробкою студентів Terna Engineering College, що належить Mumbai University [11]. При проєктуванні функціональних можливостей адміністратора було порушено безпеку системи, а саме, надали можливість адміністратору призначати ідентифікатори та паролі користувачам, що має виконуватись лише системою, а призначення та зміна паролю користувачем. Також адміністратор має можливість змінювати інформацію акаунта яка була внесена користувачем, що не є хорошою практикою. Також до проблем даного продукту можна віднести застарілий дизайн та відсутність зручного способу перегляду деталей опублікованих робіт.

Розробка студентів RM Institute of Science and Technology [2] дещо відрізняється від попередніх, адже вона націлена на людей з інвалідністю, тому включає у себе функції голосового розпізнавання та структуру сайту що спрощує навігацію. Основною проблемою вебсайту є узагальнення вакансій відносно типу інвалідності. При створенні вакансії рекрутер мав би вказувати, яким людям було б найзручніше працювати на даній посаді, і найкращим варіантом було б створення параметра вакансії, який був би виокремленим і обов’язковим, що спростило б роботу як тому хто шукає роботу, так і рекрутеру. Також підбір

кольорів не є надто доречним, адже часто зустрічається жовтий колір на білому фоні, що не є зручним для читання.

Студенти Shree Sathyam College of Engineering and Technology, Індія, розробили свій варіант системи з пошуку роботи [19]. Під час аналізу спостерігаються проблеми які були описані раніше, а саме проблеми з безпекою та розподіленням прав доступу і ролей. В цьому випадку розробники вирішили надати адміністратору платформи можливість додавати вакансії на платформу. Важко уявити ситуацію в якій адміністратор сайту буде самостійно додавати вакансії на сайт, адже це не його обов’язки. Також звітування за кількість відвідувань сайту має виконувати адміністратор, що не є логічним, адже допускається людський фактор, та й реалізація подібного функціоналу програмним чином є не складною та куди ефективнішою. Також багато проєктів нехтують зовнішнім виглядом, що дуже негативно впливає на враження користувача від платформи. В даній розробці зовнішній вигляд сайту є максимально мінімалістичним, без основних кольорів, що означає що кожна сторінка відрізняється за кольоровою схемою від інших.

Наступна розробка з Department of Computer Science and Engineering, Lovely Professional University [7]. Автор проєкту доволі безвідповідально поставився до розробки, через що елементи таблиці в якій розміщені вакансії мають назви колонок таблиці бази даних, використовується різний шрифт, різні кольори як фону, так і кнопок. Також головна сторінка надає доступ для входу як шукачам роботи і рекрутерам, так і адміністратору, що є дуже дивним рішенням, адже це велика вразливість з погляду безпеки.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Для успішного досягнення мети кваліфікаційної роботи було сформульовано ряд завдань:

– проаналізувати предметну область та розробки у даній сфері;

– визначити вимоги та спроєктувати програмне забезпечення використовуючи UML діаграми та створити дизайни сторінок;

– обрати засоби та методи створення програмного забезпечення;

– розробити програмне забезпечення відповідно до спроєктованих діаграм та дизайну та обраних засобів і методів розробки;

– написати автоматизовані тести, протестувати в ручному режимі та налагодити розроблену систему.

Висновки до розділу 1

Розробка нової платформи з пошуку роботи в IT є особливо актуальною через стрімке зростання популярності галузі, постійне збільшення кількості фахівців і роботодавців, а також через суттєві недоліки існуючих рішень. Аналіз аналогів виявив типові проблеми, з якими стикаються користувачі: перевантаження інтерфейсів зайвою або нерелевантною інформацією, низька зручність навігації, складна та часом надмірно деталізована система реєстрації й авторизації. Особливу увагу слід приділити питанню безпеки: багато платформ мають проблеми з неправильним розподілом ролей, неналежним контролем доступу до адміністративних сторінок, а також із небезпечним об'єднанням загальних і конфіденційних даних. Також спостерігаються значні порушення дизайнерських принципів, зокрема використання неконсистентних кольорів, типографіки, відсутність чіткої системи маркування елементів інтерфейсу тощо. Додаткові недоліки включають відсутність ефективних комунікаційних засобів між користувачами, використання застарілих або ненадійних технологій, а також перевантаження функціоналу, що ускладнює користування платформою. Урахування цих висновків під час створення нового проєкту дасть змогу розробити сучасну, зручну, безпечну та ефективну платформу для пошуку роботи, яка повною мірою відповідатиме поточним потребам користувачів.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Після аналізу веб платформ з пошуку роботи в IT сфері, виявлено, що багато з них мають серйозні недоліки які ускладнюють пошук роботи спеціалістам та пошук кандидатів роботодавцям. Прийнято рішення розробити власний продукт, з урахуванням помилок, допущених аналогами.

Типовими помилками подібних систем є неправильне розподілення ролей користувачів, поєднання даних безпеки з загальними даними, неконсистентність використаних кольорів, відсутність кольорового маркування елементів, відсутність необхідних функцій для взаємодії користувачів, використання неналежних засобів реалізації програмного продукту, перенасичення функціями.

Неправильне розподілення ролей серед користувачів означає, що користувачі не мають чітко визначених, однакових прав доступу до функцій платформи, що має негативні наслідки у вигляді ускладненого контролю платформи та спрощення взлому.

Поєднання даних безпеки з загальними даними означає, що розробники під час проєктування програмного забезпечення вирішили об’єднати дані загального призначення, такі як ім’я користувача, рік народження, освіта, номер телефону, з даними, що використовуються для захисту акаунту користувача. Збереження інформації в одному місці спрощує роботу зловмисникам, адже отримавши доступ до однієї таблиці даних, вони мають доступ до усієї інформації про користувача.

Неконсистентність кольорів є грубою помилкою при проєктуванні та розробці ПЗ, адже зовнішній вигляд напряму впливає на зручність користування додатком, і відсутність постійного зв’язку між кольорами може викликати дискомфорт, або відчуття незавершеності сайту.

Відсутність кольорового маркування елементів ускладнює навігацію та користування сайтом, тому що користувач на інтуїтивному рівні буде очікувати відповідність кольору елементу до його дії. Прикладом можуть бути червоний для кнопки видалення чи скарги, зелений для згоди, синій для збереження дій.

Відсутність функцій для взаємодії користувачів може бути одним з ключових недоліків любої платформи з пошуку роботу, адже у процесі пошуку зацікавлені дві особи, які мають мати змогу ефективно комунікувати між собою. Відсутність можливостей для комунікації ускладнює процес рекрутингу в рази, адже перед призначенням співбесіди потрібно обговорити невеликі деталі, а також назначити місце, дату та час проведення співбесіди, що зручно робити відразу на платформі де кандидат подав заявку, адже не факт, що кандидат щодня переглядає пошту чи соціальні мережі, які вказав як методи зв’язку, а також, рекрутеру складніше організувати свій робочий процес, адже невідомо звідки кандидат дізнався про вакансію, якщо він зв’язався з рекрутером через контактні дані.

Використання неналежних засобів реалізації програмного продукту призводить до великої кількості негативних наслідків, від багів, незручностей використання та повільної швидкості роботи, до проблем з масштабуванням проєкту та подальшою підтримкою.

Перенасичення платформи функціями може призвести до зміщення основного фокусу платформи з пошуку роботи на соцмережу чи інший інструмент. Також, перенасичення функціями призведе до складнощів у використанні платформи, а також може відштовхнути потенційних користувачів через незрозумілість або неочевидність з точки зору використання.

Після аналізу основних проблем аналогів розпочато визначення вимог до розроблюваного програмного забезпечення. Вимоги до програмного забезпечення поділяються на функціональні та нефункціональні. Функціональні вимоги визначають, який саме функціонал має бути реалізований, тобто які можливості повинна надавати система. Серед них – реєстрація та авторизація користувачів, зокрема створення облікових записів, підтримка авторизації через

соціальні мережі або email, а також можливість відновлення пароля. Також передбачено функціонал для пошуку вакансій із використанням різноманітних фільтрів за спеціальностями, досвідом та типом зайнятості. Подавати заявки на вакансії можна безпосередньо через платформу, причому лише одну заявку на одну вакансію. Адміністративна панель надає можливості перевірки та видалення невідповідних вакансій, керування користувачами та їхньою активністю, а також реагування на відгуки рекрутерів. Нефункціональні вимоги охоплюють підтримку мобільних пристроїв, мультимовність, забезпечення безпеки (включаючи шифрування паролів і приватної інформації, захист від SQL-ін’єкцій, XSS та інших типових атак), а також продуктивність – платформа повинна стабільно працювати при навантаженні до 2000 одночасних користувачів без помітних затримок.

Проєктування програмного забезпечення розпочнімо з діаграми використання, що наведена нижче (рис. 2.1), для відображення можливих дій користувачів на даній платформі у загальному вигляді.

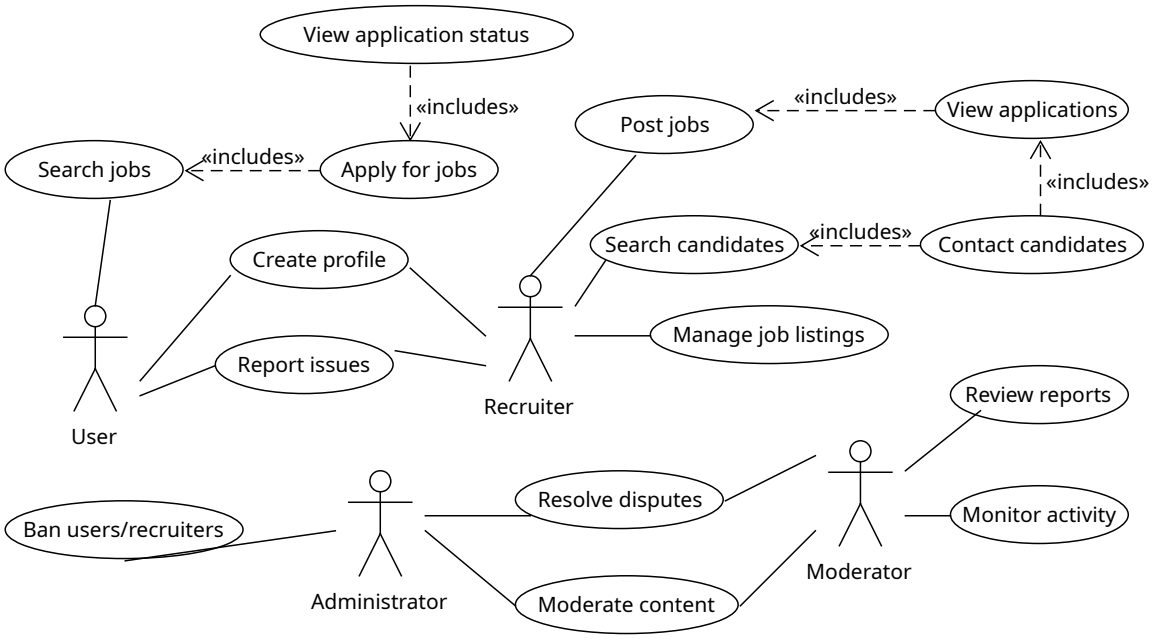

Рисунок 2.1 – Узагальнена діаграма використання платформи для усіх типів користувачів

На діаграмі можна побачити, що багато елементів використовуються декількома користувачами, що свідчить про модульність та спрощеність розробки, адже не потрібно створювати нові компоненти для рекрутера, тому що він зможе використовувати компонент що був створений для звичайного користувача.

Декілька елементів стають доступними лише після певних дій, так як вони є тісно пов’язаними, забезпечуючи, тим самим, передбачувану поведінку системи. Усі елементи вищезгаданої діаграми описані у таблиці 2.1.

Таблиця 2.1 – Опис об’єктів узагальненої діаграми

Об’єкт	Опис
User	Розробник, користувач додатку, який шукає собі роботу за допомогою додатку. Шукає вакансії
Recruiter	Рекрутер, користувач додатку, який шукає розробників у компанію. Опубліковує вакансії
Moderator	Людина що відповідає за порядок та соціальну безпеку додатку. Перевіряє скарги, вирішує конфлікти між користувачами
Administrator	Людина що керує усіма функціями платформи. Може блокувати користувачів якщо їхня поведінка не відповідає вимогам додатку
Search jobs	Пошук вакансій, що здійснюється за критеріями/фільтрами розробником
Create profile	Створення профілю користувача, що включає заповнення персональної інформації, навичок, резюме, контактних даних для розробника, та заповнення персональної інформації, інформації про компанію, контактних даних для рекрутера
Report issues	Звітування про невідповідні вакансії, профілі рекрутерів чи їх поведінку/невідповідні профілі користувачів чи їх поведінку
Apply for jobs	Відправлення заявки на опубліковану вакансію
View application status	Перегляд статусу заявки користувача на вакансію рекрутера. Можливі статуси: непрочитано/прочитано, відхилено/прийнято (рекрутер відхилив заявку що свідчить про невідповідність користувача до вимог вакансії, або прийняв, що свідчить про подальшу співпрацю/стажування/випробувальний термін)
Post jobs	Опубліковування вакансії рекрутером з описом вакансії та вимогами до неї
Search candidates	Пошук рекрутером кандидатів по заданим критеріям/фільтрам
Manage job listings	Редагування, архівування або видалення вакансій рекрутером
View applications	Перегляд та керування заявками користувачів на опубліковану вакансію
Contact candidates	Надсилання повідомлення рекрутером користувачеві

Продовження таблиці 2.1

Об’єкт	Опис
Review reports	Перегляд звітів модератором про неправильне заповнення профілю/вакансії чи неадекватну поведінку користувачів додатку (розробників та рекрутерів)
Monitor activity	Перегляд новостворених або редагованих акаунтів для забезпечення дотримання правил заповнення персональних акаунтів
Resolve disputes	Перегляд скарг користувачів та вирішення конфліктних ситуацій між ними
Ban users/recruiters	Блокування користувачів через жорстке порушення правил користування, або неодноразове порушення правил користування додатком
Moderate content	Вирішення складних питань та конфліктів між користувачами

Після створення діаграми використання платформи розпочалось створення карти сайту. Карта сайту це діаграма що зображує доступні користувачеві сторінки та шляхи переходу між ними. Корисна при розробці шляхів маршрутизації сайту. Розроблено 3 варіанти карти сайту відповідно до кожної групи користувачів.

Карта сайту для користувача (рис. 2.2) дає уявлення про доступні сторінки користувачу та зв’язки між ними.

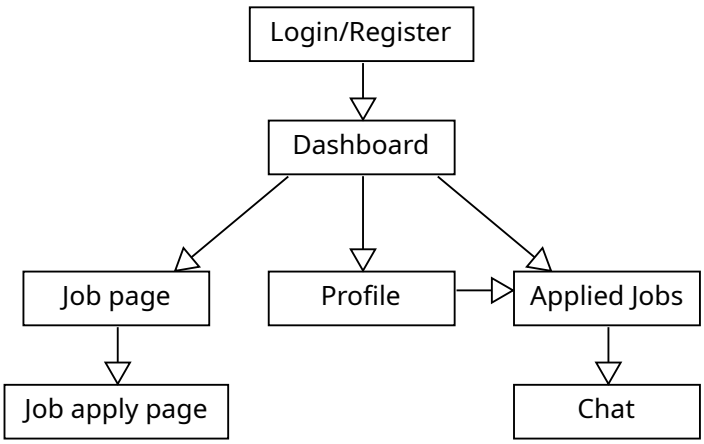

Рисунок 2.2 – Карта сайту для користувача

Першою сторінкою є сторінка авторизації та реєстрації.

Після успішної авторизації користувач потрапляє на головну сторінку, де проводиться пошук вакансій, та звідки він може переходити на інші сторінки, такі як профіль, сторінка з деталями про роботу або подані заявки. Доступ до сторінок відкривається поступово, не перевантажуючи інтерфейс та увагу користувача, при цьому створюючи інтуїтивно зрозумілий шлях для кожної сторінки. Інтерфейс побудований таким чином, щоб навіть нові користувачі могли швидко зорієнтуватися у функціоналі системи. Крім цього, передбачено адаптивний дизайн, що забезпечує зручність використання як на комп’ютерах, так і на мобільних пристроях. Детальний опис об’єктів карти сайту наведений у таблиці 2.2.

Таблиця 2.2 – Опис об’єктів карти сайту

Об’єкт	Опис
Login/Register	Сторінка, що відповідає за реєстрацію або авторизацію користувача. Є основною точкою входу, забезпечуючи безпеку платформи та визначення прав користувача, аби надати відповідні права та можливості користувачу
Dashboard	Основна сторінка, що має роль хабу, звідки можна перейти на усі інші доступні сторінки. Містить вакансії що можуть зацікавити користувача, можливість пошуку серед опублікованих вакансій, а також посилання на перехід на сторінку профілю чи до списку поданих заявок
Job page	Сторінка, що має детальну інформацію про вакансію, з можливістю переходу на сторінку відгуку на вакансію
Job apply page	Сторінка відгуку на вакансію, де користувач може завантажити своє резюме та відправити його рекрутеру
Profile	Сторінка профілю з інформацією про користувача. Вносити зміни до профілю можливо на цій сторінці, так само як і переглядати вигляд профілю
Applied jobs	Сторінка зі списком відгуків на вакансії. Користувач зможе переглянути коли і на які вакансії від відгукувався, їх поточний статус, а також розпочати спілкування з рекрутером
Chat	Сторінка з діалогом між користувачем та рекрутером. Діалог прив’язаний до відгуку користувача на вакансію, аби обом був зрозумілий контекст розмови. Тут користувач зможе запитати рекрутера про деталі вакансії, або ж рекрутер обговорить час та спосіб проведення інтерв’ю

Наступна карта сайту, що розроблена для рекрутера, представлена на рисунку 2.3.

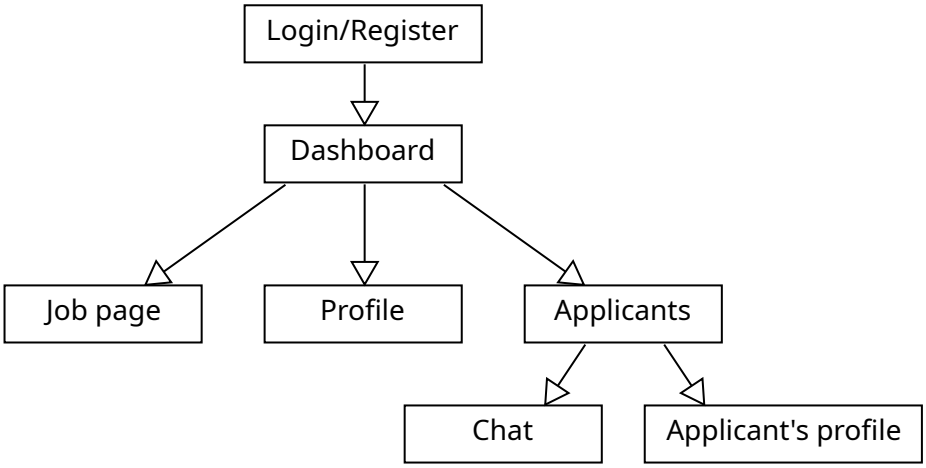

Рисунок 2.3 – Карта сайту рекрутера

Набір сторінок схожий до карти сайту користувача, але рекрутер не має доступу до сторінки подання заявки на вакансію та списку поданих заявок, натомість має доступ до списку користувачів, що відгукнулись на вакансію, та їх профілів. У таблиці 2.3 наведений детальний опис об’єктів з рисунку 2.3.

Таблиця 2.3 – Опис об’єктів карти сайту рекрутера

Об’єкт	Опис
Login/Register	Сторінка що відповідає за реєстрацію або авторизацію користувача. Є основною точкою входу на платформу, забезпечуючи безпеку платформи та визначення прав користувача, що дозволить надати доступ до тих дій та сторінок, що вимагає роль користувача
Dashboard	Основна сторінка, що має роль хабу, звідки можна перейти на усі інші доступні сторінки. Містить опубліковані рекрутером вакансії та статистику відгуків на них
Job page	Сторінка публікації вакансії. Використовується для створення вакансії та її публікації на платформі
Profile	Сторінка профілю з інформацією про користувача. Вносити зміни до профілю можливо на цій сторінці, так само як і переглядати вигляд профілю
Applicants	Сторінка зі списком користувачів, що подали заявку на вакансію. Список прив’язаний до вакансії
Chat	Сторінка з діалогом між користувачем та рекрутером. Діалог прив’язаний до відгуку користувача на вакансію, аби обом був зрозумілий контекст розмови. Тут користувач зможе запитати рекрутера про деталі вакансії, або ж рекрутер обговорить час та спосіб проведення інтерв’ю
Applicant’s profile	Сторінка профілю користувача, де рекрутер може ознайомитись з детальною інформацією про розробника

Остання карта сайту (рис. 2.4) розроблена для користувачів з вищими правами доступу, а саме модераторів та адміністраторів.

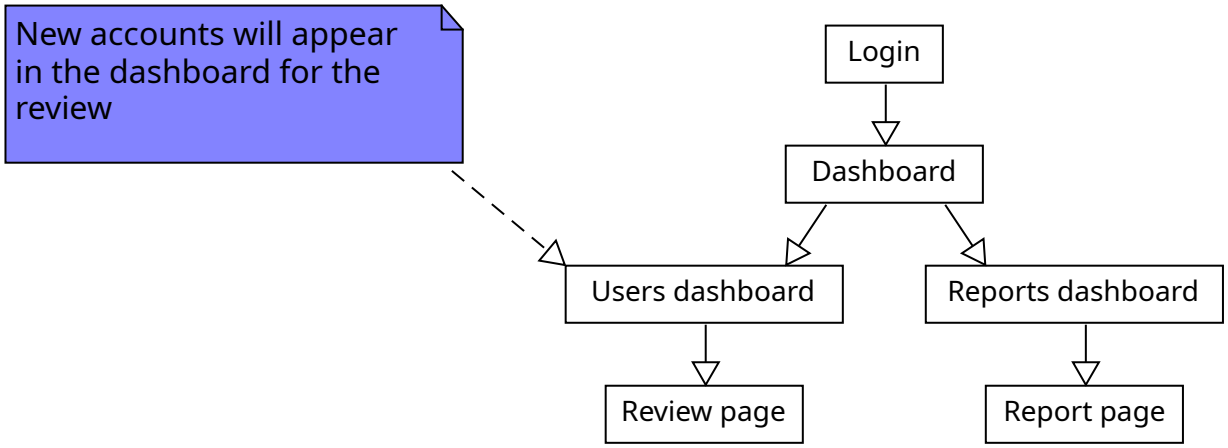

Рисунок 2.4 – Карта сайту для модераторів та адміністраторів

Адміністрація та модерація платформи є довіреними особами, тому немає можливості реєстрації акаунтів з вищими правами доступу. Адміністраторам та модераторам доступні сторінки зі скаргами та новими акаунтами платформи. Ці сторінки будуть використовуватись для запобігання порушень правил платформи та реагування на порушення чи непорозуміння між користувачами.

Опис об’єктів діаграми карту сайту для модераторів та адміністраторів наведений у таблиці 2.4.

Таблиця 2.4 – Опис об’єктів карти сайту для модераторів та адміністраторів

Об’єкт	Опис
Login	Сторінка авторизації. Реєстрація має проводитись через базу даних керівником платформи
Dashboard	Основна сторінка з інформацією про нові акаунти або скарги від користувачів чи рекрутерів
Users dashboard	Сторінка зі списком новостворених акаунтів на платформі, які мають перевірятись модератором на відповідність правилам платформи
Review page	Сторінка з акаунтом, що перевіряється
Reports dashboard	Сторінка зі списком скарг від користувачів платформи
Report page	Сторінка з детальною інформацією про скаргу

Наступним кроком у проєктуванні після UML діаграм [8] є створення дизайну основних сторінок. Дизайн продукту є одним з ключових факторів привернення уваги користувачів, адже це перше, на що користувач зверне увагу при першій зустрічі із сайтом. Зручність користування та привабливість сайту повинні бути на високому рівні, для утримання користувача на платформі та полегшенні популяризації продукту.

Враховуючи сучасну тенденцію використання згладжених кутів, дизайн фокусується на представленні окремих елементів різного забарвлення зі згладженими кутами, що створює сучасний вигляд сайту, та робить дизайн менш строгим, що є важливим при залученні групи молодих користувачів. Такий підхід дозволяє зменшити візуальну напругу та зробити інтерфейс більш дружнім і доступним для широкої аудиторії. Крім того, використання м’яких ліній сприяє кращому сприйняттю інформації та полегшує навігацію.

Для створення приємного враження, було обрано пастельні кольори, темні для фонів, та світлі для контрасту з текстом. Темний фон не дратуватиме користувачів при користуванні платформою в темних приміщеннях або вночі, а світлі блоки для тексту дозволять тексту виділитись на їх фоні для легшого читання. Подібна палітра кольорів дозволить відокремитись дизайном на фоні конкурентів та створить вигляд, що запам’ятовується. Кольорові акценти використовуються для виділення важливих елементів, таких як кнопки чи повідомлення, що покращує взаємодію з інтерфейсом.

Першою сторінкою дизайн якої було створено, стала сторінка авторизації (рис. 2.5). Сторінка авторизації повинна містити поля для вводу інформації та блок з елементами реєстрації за допомогою інших джерел. Для спрощення навігації по сторінці додані іконки, що вказують на тип даних які потрібно вводити в поля розташовані поруч. Також було враховано адаптивність дизайну – сторінка коректно відображається як на десктопах, так і на мобільних пристроях, зберігаючи зручність та функціональність незалежно від розміру екрана.

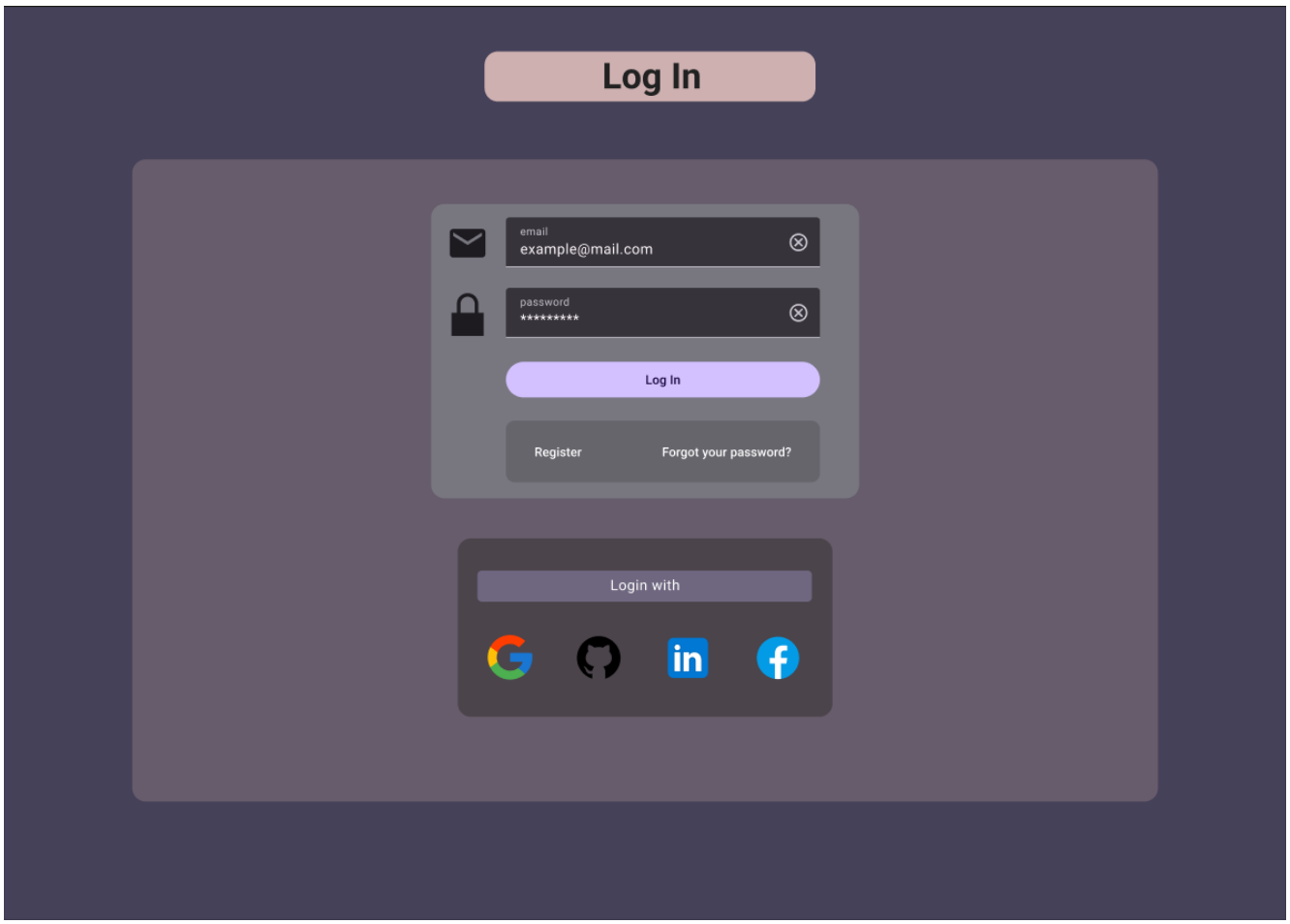

Рисунок 2.5 – Дизайн сторінки авторизації

Наступною сторінкою, що отримала дизайн, стала сторінка реєстрації (рис. 2.6). Сторінки авторизації та реєстрації є дуже схожими між собою, адже потребують однакової інформації від користувача. Основними відмінностями сторінки реєстрації від сторінки авторизації є відсутність посилань на відновлення паролю та сторінку реєстрації. Натомість, посилання надає можливість перейти на сторінку авторизації. Також, для спрощення навігації були змінені написи на сторінці, аби її легко можна було відрізнити від сторінки авторизації. Крім того, на сторінці реєстрації були додані додаткові підказки для заповнення полів, що допомагають уникнути помилок під час введення даних. Це підвищує рівень зручності використання платформи. Дизайн також адаптовано для використання на мобільних пристроях, щоб користувачі могли зручно зареєструватися з будь-якого пристрою.

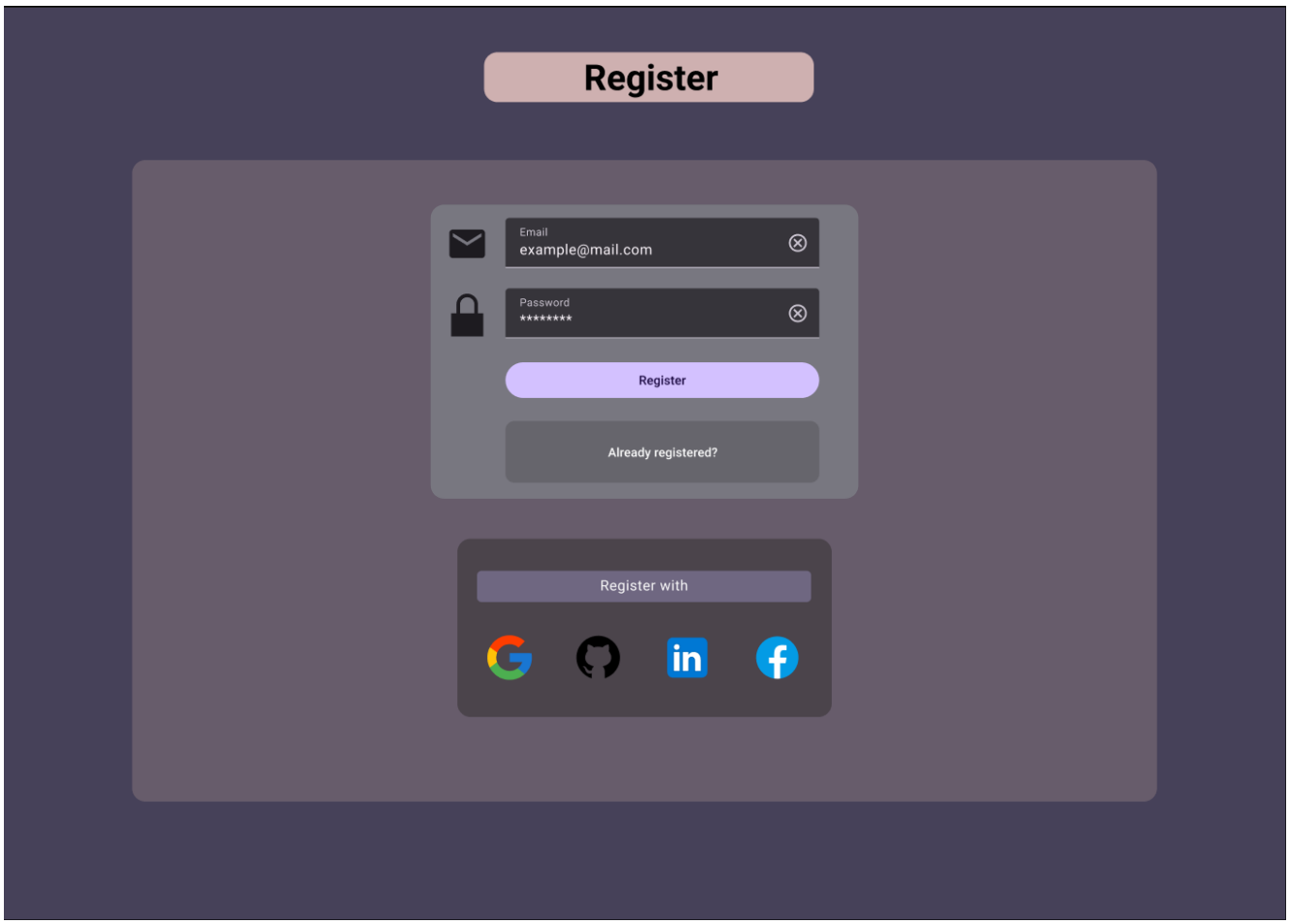

Рисунок 2.6 – Дизайн сторінки реєстрації

Ще однією сторінкою, що отримала дизайн, є основна сторінка, що містить пошук вакансій (рис. 2.7). Ця сторінка буде доступна користувачу та дозволить йому зручно виконувати пошук вакансій за заданими параметрами та переглядати інформацію про них. Так як сторінка повинна містити великий список вакансій, має мати місце для показу детальної інформації про вакансію, та надавати можливість фільтрування, було вирішено зробити фільтр невеликим блоком, що можна згорнути. Це надасть користувачу більше місця для роботи, якщо він буде цього потребувати. Над блоком фільтрів розташований список вакансій, і більшу частину сторінки займає поле, де буде відображуватись детальна інформація про вакансію. Також на сторінці з’явився заголовок з посиланнями на інші сторінки та іконка профілю, за допомогою якої користувач зможе перейти на сторінку власного профілю, що спростить та підвищить зручність навігації.

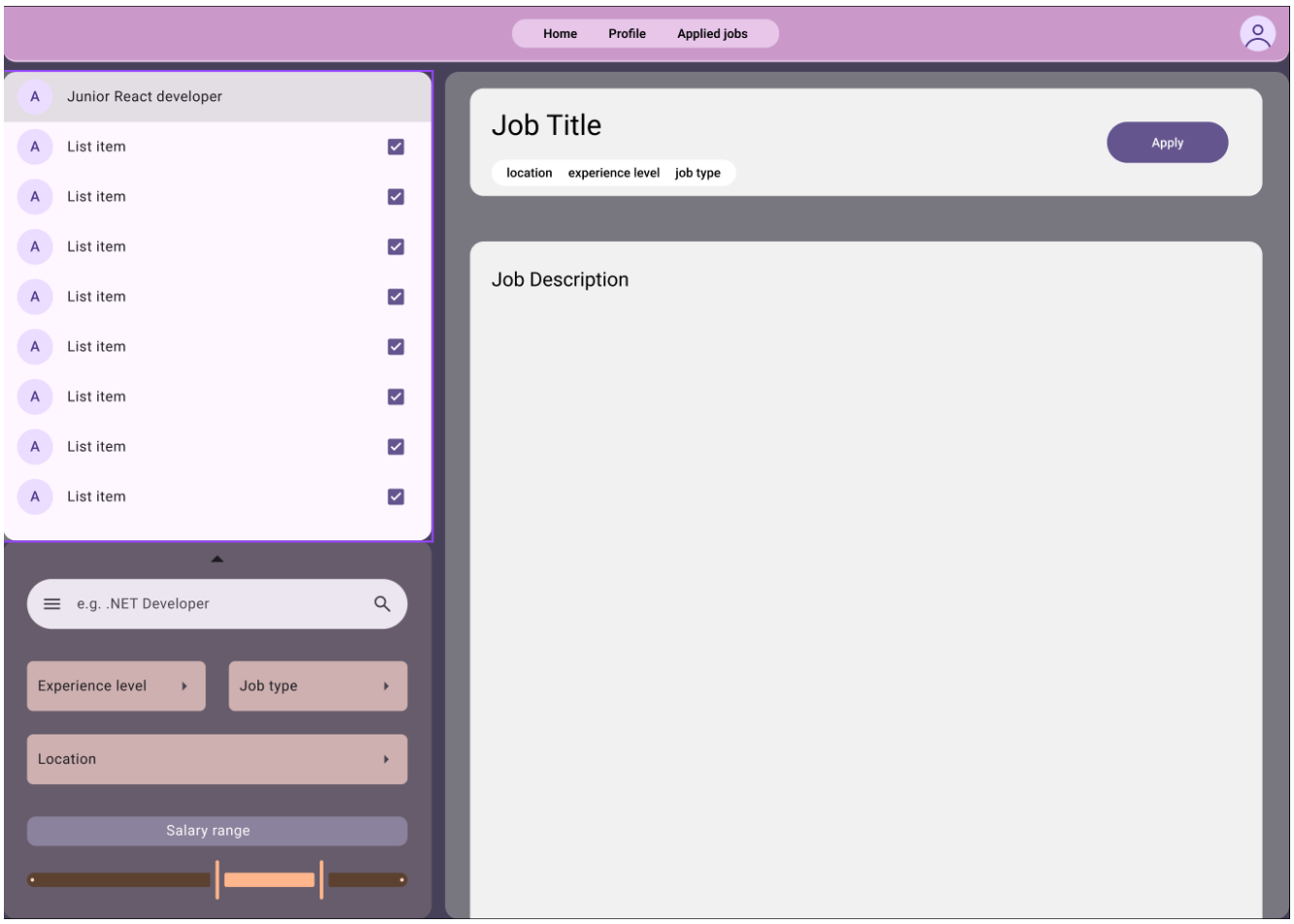

Рисунок 2.7 – Дизайн домашньої сторінки

Ще однією важливою сторінкою, яка отримала дизайн, є сторінка профілю (рис. 2.8). Сторінка профілю повинна надавати користувачеві зручний спосіб перегляду та редагування персональних даних. Користувач зможе вказати роль, на яку претендує, власні навички, кількість років досвіду, країну проживання та рівень англійської. Для зручності редагування дані розділені на окремі блоки, кожен з яких має власне виділене поле. Це дозволяє уникнути перевантаження інтерфейсу та дає змогу фокусуватися лише на потрібній інформації. У майбутньому сторінку можна буде розширити можливістю завантаження фото профілю, елементами портфоліо, відгуками від інших користувачів та можливістю перегляду активності на платформі. Такий підхід до дизайну дозволяє забезпечити інтуїтивність, простоту у використанні та водночас високу інформативність сторінки.

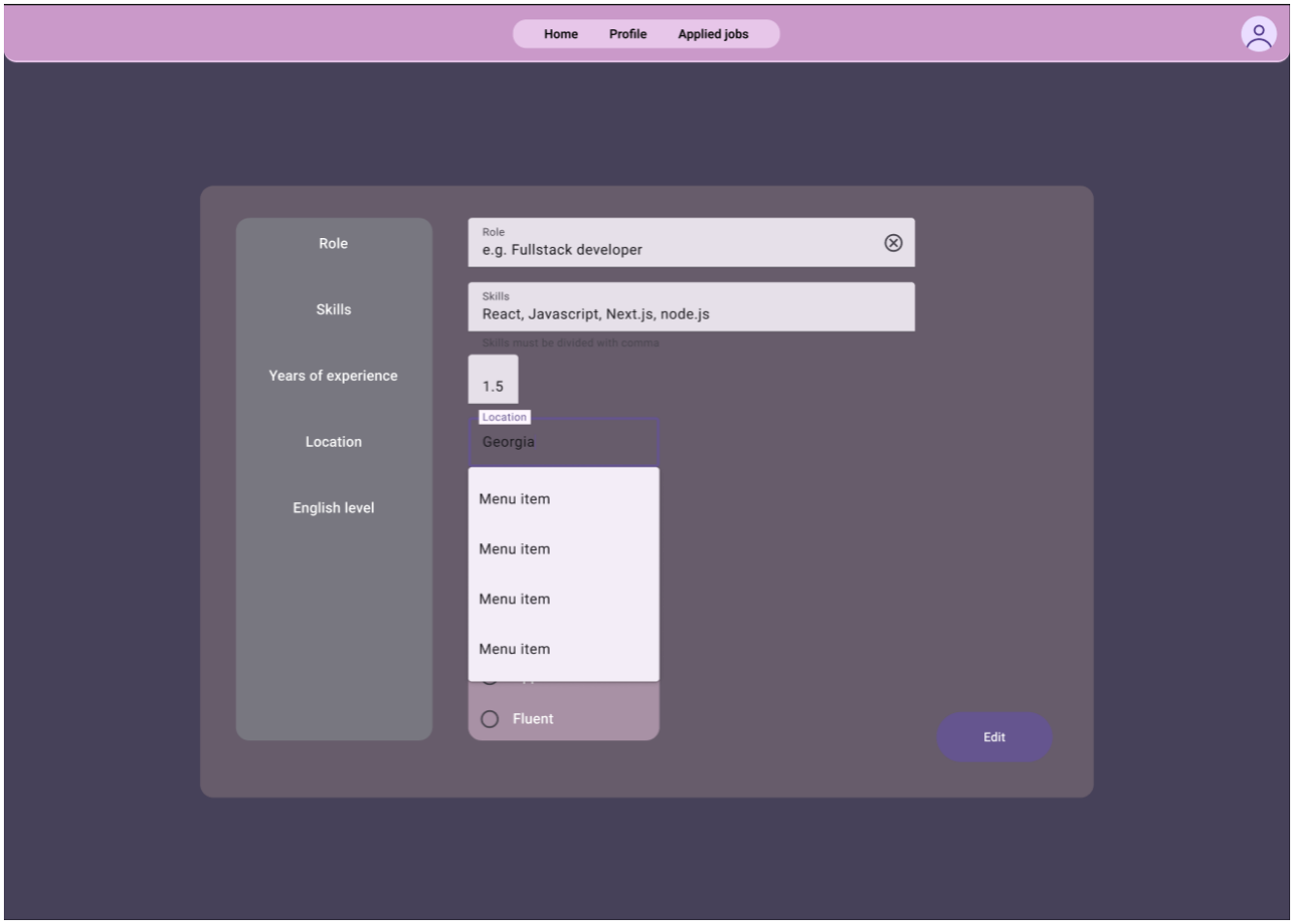

Рисунок 2.8 – Дизайн сторінки профілю

Після закінчення розробки дизайну, варто розпочати роботу над проєктуванням основної частини проєкту, яка визначатиме подальший дизайн та структуру проєкту. Мова йде про базу даних. Для того, аби почати розробку бази даних, потрібно визначити, яка інформація потребує збереження. Провівши аналіз інформації якої потребує платформа, визначили, що потрібно зберігати в базі даних персональну інформацію користувачів, рекрутерів, дані для входу, вакансії, відгуки (скарги), заявки користувача на вакансії, логування дій адміністраторів та логування зареєстрованих акаунтів. На основі отриманої інформації можемо визначити типи сутностей які можуть бути репрезентовані таблицями в базі даних: користувачі, рекрутери, адміністратори, повідомлення, чати, вакансії, скарги, заявки, логи, зареєстровані акаунти. Для завершення поверхневого проєктування бази даних залишилось визначити зв’язки між сутностями. Для визначення необхідних зв’язків потрібно з’ясувати яким чином

сутності будуть комунікувати між собою під час взаємодії користувача з платформою.

Враховуючи можливі шляхи взаємодії сутностей між об’єктами, зв’язки можуть бути наступними: користувач – вакансія (багато до багатьох), рекрутер – вакансія (один до багатьох), рекрутер – скарга (один до багатьох), користувач – скарга (один до багатьох), користувач – повідомлення (один до багатьох), рекрутер – повідомлення (один до багатьох), повідомлення – чат (багато до одного), чат – користувач (багато до одного), чат – рекрутер (багато до одного), адміністратор – скарги (багато до багатьох), адміністратор – логи (один до багатьох), адміністратор – зареєстровані акаунти (багато до багатьох), що зображено на рисунку 2.9.

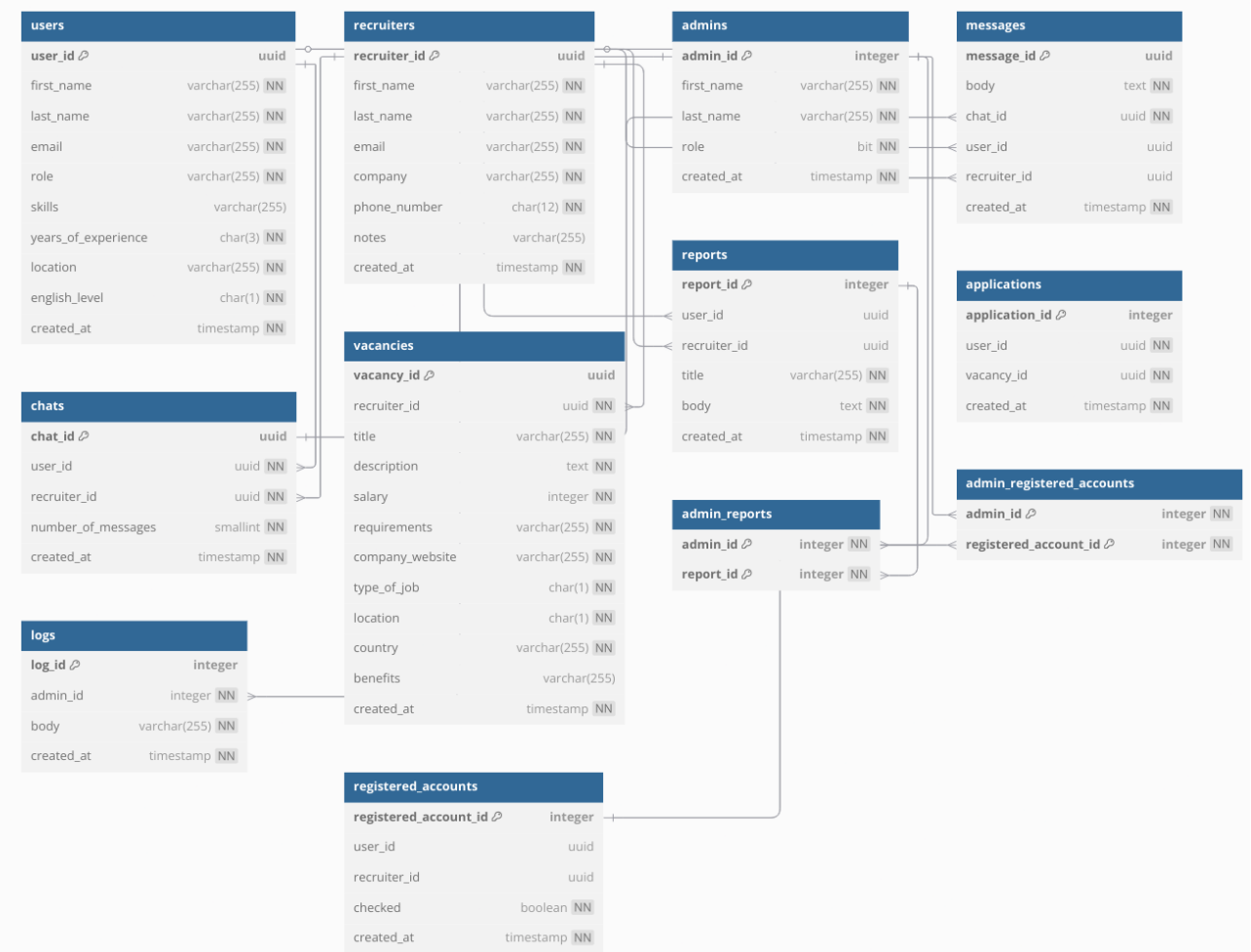

Рисунок 2.9 – Діаграма зв’язків між сутностями бази даних

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для визначення технічних вимог потрібно проаналізувати можливі варіанти технологій, що дозволяють розробити потрібне програмне забезпечення. Від правильного вибору технологічного стеку залежить не лише ефективність реалізації проєкту, а й подальша підтримка, масштабованість та продуктивність системи. На основі досліджень, використаних у першому розділі, вдалося виділити основні технології, що використовувались при створенні аналогів нашого проєкту: Microsoft SQL Server [14], Internet Information Services, HTML, CSS, ASP.NET [3], JavaScript, JQuery [13], Apache HTTP Server [18], PHP, MySQL, Bootstrap, Laravel, Python [6], Django, Node.js, Sequelize ORM, Angular, MongoDB [15], ReactJS, SQLite [17], expressJS, Redis DB [16].

Ці технології охоплюють широкий спектр напрямів розробки: від клієнтської частини до серверної логіки та зберігання даних.

Для початку розподілимо усі технології на групи:

– мови програмування: JavaScript, PHP, Python;
– сервери: Internet Informational Services, Apache HTTP Server;
– бази даних: Microsoft SQL Server, MySQL, MongoDB, SQLite, Redis DB;
– фреймворки: ASP.NET, Laravel, Django, Node.js, Angular, ReactJS, JQuery, Bootstrap, Sequelize ORM, expressJS;
– мова розмітки: HTML;
– мова стилізації: CSS.

Такий поділ дозволяє краще структурувати наявну інформацію та полегшує процес аналізу. Аби зробити остаточний вибір технологій, потрібно порівняти технології за категоріями. Першою категорією для порівняння є мови програмування, порівняння яких виконано у таблиці 2.5. У подальших розділах буде розглянуто також інші категорії, що допоможе обґрунтувати фінальний вибір стеку для реалізації проєкту.

Таблиця 2.5 – Порівняння мов програмування

Технологія	Переваги	Недоліки
JavaScript	Універсальність, широка підтримка браузерами, миттєве виконання в браузері, великий вибір бібліотек	Іноді відсутність суворої типізації може призводити до виникнення помилок. Залежність від браузерів та їхніх обмежень. Може бути менш продуктивним на серверній частині
PHP	Широке використання у веброзробці, гнучкість при роботі з базами даних, підтримка великих проєктів через фреймворки	Може стати важким у великих кодових базах без належного структурування. Проблеми з конфігурацією та старими версіями
Python	Легка синтаксична структура, відмінна підтримка для наукових обчислень, велика кількість бібліотек для різних задач	Повільніша швидкість виконання порівняно з іншими мовами (Java, C++). Може менше підходити для мобільних або високопродуктивних програм. Важлива правильна організація коду для великих проєктів

JavaScript є чимало не єдиним вибором при роботі з фронтенд частиною вебдодатків, тому він дуже популярний. Застосовується у будь-яких додатках, від найпростішого вебсайту з формою запису чи складною анімацією, до складних багатосторінкових додатків. Крім того, завдяки великій кількості бібліотек, таких як React чи Vue, JavaScript забезпечує швидку та ефективну розробку інтерфейсів.

PHP є популярним вибором для розробки логіки вебдодатків малого та середнього розміру. Наявність фреймворків дозволяє спростити застосування у більших додатках, а гнучкість при роботі з базами даних є великою перевагою при розробці невеликих та середніх додатків. Також PHP добре підтримується на більшості хостингів, що робить його доступним рішенням.

Python – мова програмування широкого використання. Своє застосування знайшла і у веброзробці. За допомогою фреймворку Django можливо створити вебсайт будь-якої складності, але швидкість виконання та відсутність суворої типізації роблять цю мову вибором для не надто великих вебдодатків, які фокусуються на великій кількості даних, статистиці та подібному. Завдяки зрозумілому синтаксису Python також є хорошим вибором для команд з менш досвідченими розробниками.

Порівняння серверів наведено у таблиці 2.6.

Таблиця 2.6 – Порівняння серверів

Критерій	Internet Informational Services (IIS)	Apache HTTP Server
Призначення	вебсервер від Microsoft, що використовується для хостингу вебсайтів та вебдодатків на платформі Windows. Ідеально підходить для інтеграції з іншими продуктами Microsoft	Відкритий вебсервер, що працює на багатьох операційних платформах. Підходить для широкого спектру застосувань, від простих сайтів до складних додатків, та має підтримку різних мов програмування
Переваги	Глибока інтеграція з ОС Windows. Простота налаштування через графічний інтерфейс. Підтримка технологій Microsoft. Хороша підтримка для корпоративних середовищ	Відкритий код і безкоштовність. Може працювати на різних ОС. Велика спільнота та безліч модулів для розширення функціоналу. Підтримка різних мов програмування та високий рівень налаштування
Недоліки	Працює лише на Windows, відносно складний для адміністрування в порівнянні з іншими вебсерверами. Може бути важким для новачків через інтеграцію з іншими технологіями Microsoft. Ліцензія може бути дорогою для деяких користувачів	Менш інтуїтивно зрозумілий для користувачів Windows у порівнянні з IIS. Більш вимогливий до ресурсів, якщо потрібно обробляти великий трафік або великі об’єми запитів. Інколи потребує складнішого налаштування для інтеграції з деякими технологіями. Можливі проблеми з безпекою при неправильному налаштуванні сервера

IIS найкраще працює разом з іншими продуктами Microsoft, тому якщо уся система вже зав’язана на їх продукції, то логічним вибором є саме цей сервер. Він має зручну інтеграцію з Windows Server, Active Directory та іншими корпоративними інструментами, що значно полегшує адміністрування та забезпечує високу продуктивність у середовищі Microsoft.

Apache HTTP Server є безкоштовним сервером з відкритим кодом, отже розробляється ентузіастами та небайдужими, що означає більш широкий спектр доступних платформ для запуску сервера та ширшу підтримку мов, а також великий набір навчальних матеріалів від спільноти. Складніший у налаштуванні аніж IIS, менш продуктивний при великих навантаженнях, однак завдяки гнучкості конфігурації може бути адаптований під різноманітні проєкти та середовища.

Порівняння систем керування базами даних наведено у таблиці 2.7.

Таблиця 2.7 – Порівняння систем керування базами даних

Технологія	Переваги	Недоліки
Microsoft SQL Server	Інтеграція з продуктами Microsoft. Потужні інструменти для бізнес-аналітики та звітності. Високий рівень безпеки та підтримка розширених функцій шифрування та доступу. Масштабованість. Підходить для великих підприємств та корпоративних рішень	Висока вартість ліцензії для великих підприємств. Менш гнучка в порівнянні з іншими СУБД щодо підтримки різних платформ. Складність налаштування та адміністрування.
MySQL	Безкоштовність та відкритість. Широка підтримка на різних ОС. Легкість налаштування та використання. Широка документація та активна спільнота. Гарна інтеграція з багатьма популярними вебтехнологіями	Не підтримує деякі функції, доступні в складніших СУБД. Може мати проблеми з масштабованістю при обробці великих обсягів даних. Менша підтримка для аналітичних і бізнес-даних порівняно з SQL Server
MongoDB	Гнучкість у зберіганні неструктурованих даних, масштабованість та висока доступність, підтримка складних запитів через агрегації, легко масштабувати в кластері	Не підтримує транзакції на рівні документа, може бути неефективна при великих складних запитах, не підтримує повну реляційну модель
SQLite	Легка вага та відсутність потреби у сервері, простота налаштування та використання, швидкість для невеликих додатків, підходить для вбудованих систем та мобільних додатків	Обмежена масштабованість для великих додатків, не підтримує повноцінний багатокористувацький доступ на рівні сервера, обмеження щодо великих обсягів даних
Redis DB	Хороші можливості для роботи з великими даними та складними запитами, підтримка ACID транзакцій, відкритий код та активна спільнота, підтримка багатьох типів даних і розширень	Складніше налаштування і адміністрування, може бути занадто важким для простих і малих проєктів, вища вимога до ресурсів при масштабуванні

Microsoft SQL Server орієнтований на великі підприємства і забезпечує розширену функціональність для бізнес-аналітики, шифрування та високої доступності. Він часто використовується у великих організаціях і для складних корпоративних застосунків.

MySQL є більш легким і зручним вибором для малого та середнього бізнесу, зокрема для вебсайтів і вебдодатків, що працюють з невеликими чи середніми обсягами даних. Його переваги полягають у відкритості, простоті налаштування та інтеграції з іншими вебтехнологіями, але з обмеженою підтримкою складних корпоративних функцій.

MongoDB це NoSQL база даних, що означає що вона не реляційна, тобто зберігає дані у вигляді документів, а не рядків даних. Гнучка у збереженні даних, тому що тип і формат даних визначає програма що використовує цю базу даних. Легко масштабується через спрощену логіку обробки даних, підтримує складні запити через агрегацію що є зручним способом отримання даних з бази даних. Має проблеми із комплексними важкими запитами.

SQLite створена для роботи на вбудованих та мобільних пристроях. Спеціально розроблена легкою аби малопотужні девайси могли обробляти запити самостійно, без звернень до серверів. Просто налаштовується, має високу швидкість як для невеликих додатків. Не підтримує багатокористувацький доступ, обмежена щодо розміру даних, обмежена масштабованість.

Redis DB є аналогом Microsoft SQL Server з відкритим кодом, що підтримує ACID транзакції, тригери та складні запити. Має велику кількість розширень для більш широкої конфігурації. При масштабуванні стає вимогливою до ресурсів, не є хорошим варіантом для невеликих чи простих додатків. Складніша в налаштуванні та адмініструванні аніж молодші аналоги.

Наступними порівнюються фреймворки у таблиці 2.8.

Таблиця 2.8 – Порівняння фреймворків

Технологія	Переваги	Недоліки
ASP.NET	Масштабованість, безпека, підтримка Microsoft	Складність для початківців, менша підтримка в Open Source, високі системні вимоги
Laravel	Легкість у навчанні, вбудовані можливості, велика спільнота	Продуктивність, менш ефективний для великих проєктів
Django	Швидка розробка, безпека, масштабованість	Не підходить для малих проєктів, важкий для малих серверів
Node.js	Швидкодія, єдина мова розробки для фронтенду та бекенду, масштабованість, велика екосистема	Однопотоковість, важкість масштабування високонавантажених завдань, менша підтримка інших мов програмування
Angular	Модульність, декларативний підхід, двостороннє зв’язування даних	Висока складність для новачків, великий розмір вихідного коду
ReactJS	Компонентний підхід до розробки, віртуальний DOM, легко інтегрується з іншими технологіями, підтримка SSR	Потребує додаткових бібліотек для керування станом, може мати круту криву вивчення для новачків, великий розмір бібліотеки при включенні багатьох компонентів

ASP.NET – це фреймворк для розробки вебдодатків на платформі .NET. Його основними перевагами є масштабованість, висока безпека та підтримка з боку Microsoft. Водночас він має і недоліки: складний для початківців, має меншу підтримку в спільноті з відкритим кодом, а також вимагає високих системних ресурсів.

Laravel – фреймворк для розробки серверної частини на мові PHP. Його часто обирають через легкість у навчанні, вбудовані можливості та велику спільноту. Однак, його продуктивність може бути нижчою, і він менш ефективний для реалізації великих проєктів.

Django – фреймворк для розробки вебдодатків на Python. Він дозволяє швидко створювати додатки, забезпечує високий рівень безпеки та добре масштабується. Проте, він не найкращий вибір для малих проєктів і може бути надто «важким» для малопотужних серверів.

Node.js використовується для створення серверних додатків і API. Серед його переваг – висока швидкодія, можливість використання однієї мови (JavaScript) для фронтенду та бекенду, масштабованість і велика екосистема. Основні недоліки – це однопоточна природа, складнощі з масштабуванням при великому навантаженні та обмежена підтримка інших мов програмування.

Angular призначений для розробки односторінкових вебдодатків. Його сильними сторонами є модульність, декларативний підхід і двостороннє зв’язування даних. Водночас, Angular досить складний для новачків і генерує великий об’єм вихідного коду.

ReactJS – бібліотека для створення інтерфейсів односторінкових сайтів. Вона підтримує компонентний підхід до розробки, використовує віртуальний DOM, легко інтегрується з іншими технологіями та підтримує серверний рендеринг (SSR). Недоліками є потреба в додаткових бібліотеках для керування станом, крута крива навчання для новачків і великий розмір бібліотеки при використанні багатьох компонентів.

Останніми будуть порівнюватись бібліотеки у таблиці 2.9.

Таблиця 2.9 – Порівняння бібліотек

Критерій	JQuery	Bootstrap	Sequelize ORM	expressJS
Призначення	Бібліотека JavaScript для контролю DOM, AJAX та анімацій	CSS-фреймворк для створення адаптивних інтерфейсів з готовими компонентами	Бібліотека для Node.js для роботи з реляційними базами даних	Мінімалістичний фреймворк для Node.js який спрощує створення додатків та обробку HTTP-запитів
Переваги	Простота використання, крос-браузерність, плагіни	Швидка розробка, адаптивний дизайн, готові компоненти, крос-браузерність	Абстракція над SQL, підтримка кількох баз даних, має підтримку міграцій, підтримує транзакції	Легкий у використанні та налаштуванні, велика кількість плагінів, підтримує шаблони, підтримка RESTful API
Недоліки	Втрата популярності, не підходить для великих проєктів, великий розмір бібліотеки	Обмежена кастомізація, стандартний дизайн	Продуктивність, відсутність складних можливостей, не завжди коректно працює з усіма типами даних, має великий розмір	Може бути занадто базовим для складних проєктів, не має вбудованих функцій для масштабування, вимагає знань Node.js та асинхронного програмування

jQuery найкраще підходить для керування DOM, AJAX-запитів та простих інтерактивних елементів. Зараз його використовують рідше, оскільки сучасні версії JavaScript та фреймворки заміщують його функціональність. Проте він досі корисний у невеликих проєктах або для швидкого додавання ефектів на сторінку без складної логіки.

Bootstrap – зручний інструмент для швидкої розробки адаптивних сайтів з готовими компонентами, але не підходить для унікального або складного дизайну. Його сіткова система та набір шаблонів дозволяють швидко створити базову структуру сайту без потреби писати багато CSS вручну.

Sequelize ORM абстрагує роботу з реляційними базами даних через JavaScript, спрощуючи створення та зберігання даних, однак може бути менш ефективною для складних систем порівняно з чистими SQL-запитами. Крім того, при великій кількості зв’язків між таблицями можуть виникати труднощі з оптимізацією запитів.

Бібліотека expressJS у парі з Node.js спрощує розробку API, обробку HTTP-запитів і серверних додатків. Завдяки великій спільноті має багато плагінів і middleware, але обмежений у функціональності та не має вбудованих засобів масштабування, тому більше підходить для простих рішень. Проте його легкість і мінімалізм дозволяють точно контролювати архітектуру вебдодатків.

Після аналізу було обрано Angular, ASP.NET та Redis DB. Angular приваблює компонентною архітектурою, декларативним підходом і обов’язковим використанням TypeScript – надбудови над JavaScript із типізацією. Angular Material забезпечує набір готових компонентів, що пришвидшить розробку клієнтської частини, дозволяючи зосередитися на логіці замість стилізації елементів.

ASP.NET має багато вбудованих функцій, простий у конфігурації, підтримує роботу з великими обсягами даних і має високий рівень безпеки, що критично важливо для системи з профілями користувачів. Він також має хорошу інтеграцію з інструментами DevOps та підтримує хмарні рішення, що відкриває можливість масштабування проєкту.

У ролі бази даних обрано Redis DB – відкриту альтернативу Microsoft SQL Server, яка краще працює на Linux, що є важливим з огляду на популярність цієї ОС для серверів. Redis забезпечує швидкий доступ до даних завдяки зберіганню інформації в оперативній пам’яті, що особливо важливо для високонавантажених систем.

Як сервер буде використано Apache HTTP Server завдяки оптимізації під Linux та широким можливостям налаштування. Він стабільний, активно підтримується спільнотою, має гнучку конфігурацію та розширюється за допомогою модулів.

Вибір мов програмування продиктований обраними технологіями: Angular вимагає використання TypeScript, HTML і CSS (замість CSS застосовується SCSS – надбудова, яка додає змінні та міксини для зручнішої стилізації); ASP.NET використовує C# для серверної логіки, який є строго типізованим і підтримує об'єктно-орієнтований підхід.

Технічні вимоги проєкту передбачають використання ASP.NET Core для back-end, Angular з Angular Material – для front-end, Redis DB – як базу даних. Для контейнеризації буде застосовано Docker – інструмент, що забезпечує стабільну роботу програми в будь-якому середовищі, незалежно від пристрою. Docker усуває типові проблеми, як-от відсутність потрібних бібліотек, гарантуючи наявність усіх необхідних компонентів і налаштувань. Також він полегшує розгортання проєкту в різних середовищах, таких як staging чи production.

Висновки до розділу 2

Проаналізовано аналоги проєкту та виявлено їхні недоліки, зокрема проблеми з безпекою, дизайном, відсутністю важливих функцій або перенасиченням можливостями, що перетворює платформу з пошуку роботи на соціальну мережу. У деяких випадках спостерігається невиправдане ускладнення функціоналу, що лише відволікає користувача від ефективного пошуку роботи або підбору кандидатів. Визначено функціональні та нефункціональні вимоги до системи з акцентом на простоту взаємодії, надійність та масштабованість. Створено загальну діаграму варіантів використання на основі аналізу подібних платформ, а також карти сайту для кожного типу користувача з урахуванням необхідних сторінок і зв’язків між ними. Розроблено дизайн основних сторінок: входу, реєстрації, головної та профілю користувача з дотриманням сучасних принципів UX/UI-дизайну. Проведено аналіз структури даних і створено діаграму зв’язків сутностей, яка використана при розробці бази даних. Також проаналізовано технології, що застосовувались у подібних рішеннях, та обрано оптимальні за продуктивністю та гнучкістю. Для реалізації бізнес-логіки використано ASP.NET, клієнтську частину реалізовано за допомогою Angular та Angular Material, базою даних слугує Redis DB. Для керування середовищем і створення повноцінної екосистеми застосовано Docker Compose, що забезпечує стабільність і зручність розгортання проєкту.

РОЗДІЛ 3
РОЗРОБКА ПОРТАЛУ З ПОШУКУ РОБОТИ «DEVJOBSTER»

3.1 Практична реалізація об’єкта проєктування

Практична реалізація об’єкта проєктування охоплює створення серверної частини додатка, та клієнтської.

Розробка розпочалась з серверної частини. Робота з серверним додатком зазвичай розпочинається зі створення моделей, за допомогою яких програма буде отримувати інформацію з бази даних. За основу для моделей слугувала діаграма зв’язків між сутностями бази даних. На рисунку 3.1 наведена структура папок, в яких зберігаються моделі.

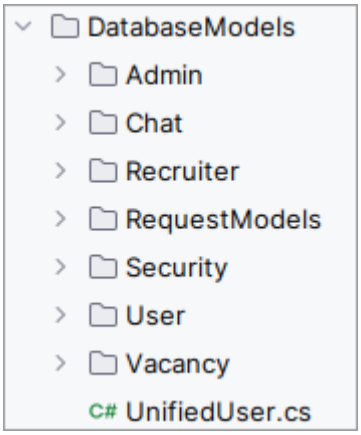

Рисунок 3.1 – Структура папок файлів моделей

Моделі згруповані за приналежністю, що виключає плутанину між моделями, що стосуються адміністрації та, наприклад, користувачів.

Папка RequestModels містить ідентичну структуру папок (рис. 3.2), де містяться спеціалізовані версії моделей, які містять меншу кількість інформації, аніж повноцінні. Це потрібно з міркувань безпеки, аби при запиті користувач отримував лише доступну йому інформацію, і не мав доступу до, припустимо, пошти інших користувачів, а тим паче паролів.

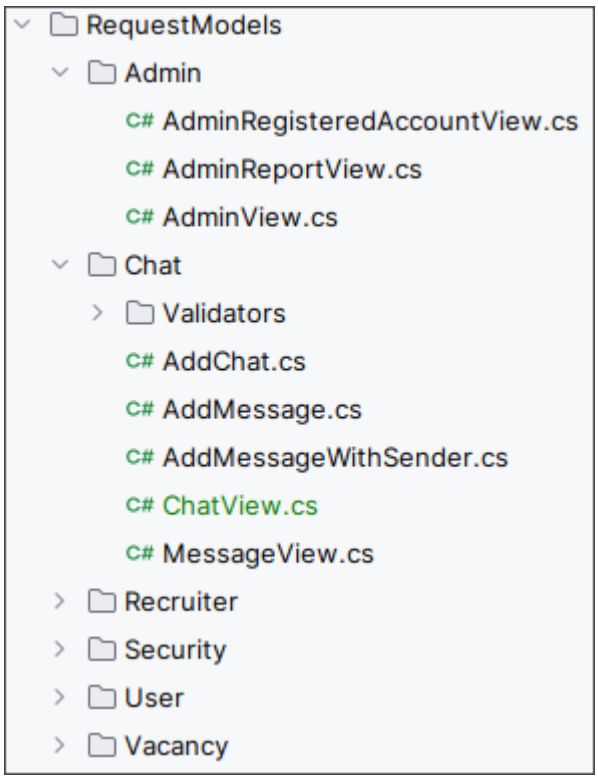

Рисунок 3.2 – Структура файлів та папок в папці RequestModels

Також, деякі з цих моделей перевіряються валідаторами, що додає ще один рівень захисту. Користувач чи зловмисник не зможуть внести будь-яку інформацію. Вона повинна відповідати вимогам валідаторів. Код валідатора моделі повідомлення наведений у лістингу 3.1.

Лістинг 3.1 – Код валідації моделі повідомлення

```
public MessageValidator()
{
    RuleFor(x => x.Body)
        .NotEmpty().WithMessage("Message can't be empty")
        .MaximumLength(5000).WithMessage("Message's length can't
be more than 5000 characters");
}
```

кінець лістингу 3.1

З фрагмента коду вище, можна дізнатись, що валідатор для текстового повідомлення обмежує довжину повідомлення у 5 тисяч символів та забороняє відправлення пустих повідомлень. Це один з найпростіших валідаторів.

Після створення моделей та валідаторів для них розпочато роботу над частиною коду, що буде взаємодіяти з базою даних.

Спочатку було створено клас обробки помилок (ліст. 3.2). Він потрібен для контролю системних помилок, оскільки системні помилки надають детальну інформацію, що може бути використано зловмисниками для пошуку слабких місць системи. Новостворений обробник помилок дозволяє подавати власну інформацію про помилку, заміняючи детальне повідомлення на щось коротке та інформативне.

Лістинг 3.2 – Власний обробник помилок

```
  public static DatabaseException
CatchDatabaseException(PostgresException exception)
    {
        var constraintName = exception.ConstraintName ??
string.Empty;
        var tableName = exception.TableName ?? string.Empty;
        throw exception.SqlState switch
        {
            "23505" => // unique_violation
                new UniqueConstraintViolationException($"A user
with this {constraintName} already exists.", exception,
                    constraintName, tableName),
            "23503" => // foreign_key_violation
                new ForeignKeyViolationException("Referenced
entity does not exist.", exception, constraintName, tableName),
            "23514" => // check_violation
                new ConstraintViolationException($"The value
violates a check constraint: {constraintName}", exception,
                    constraintName, tableName),
            "23502" => // not_null_violation
                new ConstraintViolationException($"The value
violates not null constraint: {constraintName}", exception,
                    constraintName, tableName),
            "23P01" => // exclusion_constraint_violation
                new ConstraintViolationException($"The value
violates exclusion constraint: {constraintName}",
                    exception, constraintName, tableName),
            _ => new DatabaseException("A database error
occurred.", exception)
        };
    }
```

кінець лістингу 3.2

Обробник помилок може розпізнати декілька видів помилок, такі як помилка унікальності чи надсилання пустих даних. Якщо помилка не розпізнається, вона записується як стандартна з повідомленням «Виникла помилка в базі даних», що запобігає отриманню користувачем інформації про особливості будови системи.

Наступним кроком є створення класу з централізованим доступом до бази даних через систему. Цей клас повинен надавати доступ до об’єкта зв’язку, керувати зв’язком та правильно його закривати, через особливості роботи зв’язків баз даних. Код класу з централізованим доступом до бази даних наведений у лістингу 3.3.

Лістинг 3.3 – Клас менеджер зв’язку (контекст бази даних)

```
public class DbContext : IDbContext
{
    private readonly Lazy<NpgsqlConnection> _connectionLazy;
    private bool _disposed = false;

    public DbContext(string? connectionString)
    {
        _connectionLazy = new Lazy<NpgsqlConnection>(() =>
        {
            var connection = new
NpgsqlConnection(connectionString);
            connection.Open();
            return connection;
        });
    }

    public IDbConnection Connection => _connectionLazy.Value;


    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    protected virtual void Dispose(bool disposing)
    {
        if (_disposed) return;
```

```
        if (disposing)
        {
            if (_connectionLazy is not { IsValueCreated: true,
Value.State: ConnectionState.Open }) return;
            _connectionLazy.Value.Close();
            _connectionLazy.Value.Dispose();
        }

        _disposed = true;
    }

    ~DbContext()
    {
        Dispose(false);
    }
}
```

кінець лістингу 3.3

Після створення контексту бази даних можна розпочати роботу над сервісами. Сервіси створені для керування даними бази даних, і кожен відповідає за власну частину бази даних.

В даному проєкті база даних умовно розділена на три частини: адміністративну, користувацьку та користувацького простору.

Адміністративна частина містить методи керування адміністраторами та модераторами платформи, а також їхнім простором, що складається з можливості перегляду логів системи, скарг користувачів та списку зареєстрованих користувачів для модерування інформації якою заповнюють профілі користувачі.

Користувацька частина відповідає за керування користувачами, тобто реєстрація користувачів та рекрутерів, оновлення профілю та відновлення пароля.

Частина користувацького простору відповідає за елементи з якими взаємодіють користувачі, а саме: створення, оновлення, отримання та видалення вакансій, оформлення користувачами заявок на вакансію, створення чатів та надсилання повідомлень, а також створення скарг. Приклад методу з користувацького простору наведений у лістингу 3.4.

Лістинг 3.4 – Фрагмент коду користувацької частини, оновлення профілю рекрутера

```
public async Task<int> UpdateRecruiterAsync(Guid recruiterId,
RecruiterUpdate recruiter)
{
    try
    {
        const string sql =
            """
            UPDATE recruiters SET first_name = @FirstName,
            last_name = @LastName, phone_number = @PhoneNumber,
            company = @Company, notes = @Notes
            WHERE recruiter_id = @RecruiterId;
            """;
        return await dbContext.Connection.ExecuteAsync(sql, new
        {
            RecruiterId = recruiterId,
            recruiter
        });
    }
    catch (PostgresException e)
    {
        throw DatabaseExceptionHandler.CatchDatabaseException(e);
    }
}
```

кінець лістингу 3.4

Після завершення розробки сервісів залишилось розробити контролери, які дозволять отримувати інформацію через запити до серверної частини програми, використовуючи сервіси. Контролери виступають посередниками між клієнтською частиною та логікою, реалізованою у сервісах.

Існують контролери для керування адміністративною частиною, заявками на вакансії, реєстрацією, чатами, повідомленнями, рекрутерами, користувачами та вакансіями. Кожен з них відповідає за окремий блок функціональності, що дозволяє розділити логіку та спростити підтримку коду.

Контролер являє собою набір методів, які можна викликати за допомогою Інтернет-запитів. Кожен метод має свій ідентифікатор у вигляді посилання, що наведено у лістингу 3.5. Також методи контролерів можуть мати параметри, які

передаються через URL або тіло запиту, що забезпечує гнучкість у роботі з даними.

Лістинг 3.5 – Контролер повідомлень

```
  messageGroup.MapGet("/{messageId:guid}",
            async Task<Results<Ok<MessageView>, NotFound>> (Guid
messageId, IUserSpaceService userSpaceService) =>
            {
                var message = await
userSpaceService.GetMessageByIdAsync(messageId);
                if (message == null) return
TypedResults.NotFound();
                var messageView = new MessageView(message.Body,
message.ChatId, message.UserId, message.RecruiterId);
                return TypedResults.Ok(messageView);
            })
        .RequireAuthorization();
    messageGroup.MapPost("/",
            async Task<Results<Created<AddMessageWithSender>,
BadRequest, ForbidHttpResult>> (
                AddMessage addMessage, ClaimsPrincipal user,
IUserSpaceService userSpaceService) =>
            {
                var senderId =
user.FindFirst(ClaimTypes.NameIdentifier)?.Value;
                var role = user.FindFirst(ClaimTypes.Role)?.Value;
                if (senderId is null || role is null) return
TypedResults.BadRequest();
                var senderGuid = Guid.Parse(senderId);
                var isPartOfChat = await
userSpaceService.IsUserPartOfChatAsync(addMessage.ChatId,
senderGuid, role);
                if (!isPartOfChat) return TypedResults.Forbid();
                var message = new
AddMessageWithSender(Guid.NewGuid(), addMessage.ChatId,
addMessage.Body, Guid.Parse(senderId), role, DateTime.UtcNow);
                var result = await
userSpaceService.CreateMessageAsync(message);
                return result > 0
                    ?
TypedResults.Created($"/api/messages/{message.MessageId}",
message)
                    : TypedResults.BadRequest();
```

```
        })
    .WithValidation<MessageValidator>()
    .RequireAuthorization("UserAndRecruiterOnly");
```

кінець лістингу 3.5

Більша частина серверного додатка завершена. Він функціонує та може використовуватись для отримання та внесення інформації. Але для повноцінного проєкту не вистачає заходів безпеки, а саме системи аутентифікації та авторизації. Також варто додати спосіб шифрування даних. Для системи входу вирішено використати просту, але надійну систему токенів. При вході користувач зобов’язаний вказати реєстраційні дані, і якщо дані збігаються, йому надсилається токен, що містить інформацію про роль користувача та його права доступу. Рольова система реалізована задля запобігання доступу до даних інших користувачів. У лістингу 3.6, наведено код генерації токена.

Лістинг 3.6 – Код генерації токена

```
private static string GenerateJwtToken(Guid userId, UserType
userType, IConfiguration configuration)
{
    var tokenHandler = new JwtSecurityTokenHandler();
    var key = Encoding.ASCII.GetBytes(configuration["Jwt:Key"]);

    var claims = new[]
    {
        new Claim(JwtRegisteredClaimNames.Sub, userId.ToString()),
        new Claim(JwtRegisteredClaimNames.Jti,
Guid.NewGuid().ToString()),
        new Claim(JwtRegisteredClaimNames.Iat,
DateTimeOffset.UtcNow.ToUnixTimeSeconds().ToString(),
            ClaimValueTypes.Integer64),
        new Claim(ClaimTypes.Role, userType.ToString()),
        new Claim(ClaimTypes.NameIdentifier, userId.ToString())
    };

    var tokenDescriptor = new SecurityTokenDescriptor
    {
        Subject = new ClaimsIdentity(claims),
        Expires = DateTime.UtcNow.AddDays(7),
        Issuer = configuration["Jwt:Issuer"],
        Audience = configuration["Jwt:Audience"],
```

```
        SigningCredentials = new SigningCredentials(
            new SymmetricSecurityKey(key),
            SecurityAlgorithms.HmacSha256Signature)
    };

    var token = tokenHandler.CreateToken(tokenDescriptor);
    return tokenHandler.WriteToken(token);
}
```

кінець лістингу 3.6

Для подальшого захисту системи потрібно реалізувати шифрування паролів. Зберігати незашифровані паролі категорично забороняється.

Для шифрування паролів вирішено використовувати популярну та надійну бібліотеку хешування BCrypt [5], з алгоритмом хешування SHA384.

Отримуючи пароль від користувача, код використовує бібліотеку хешування для отримання хешу пароля, і зберігає хеш в базі даних як пароль. Даний хеш не містить жодної інформації про пароль. Тобто пароль можна конвертувати в хеш, але хеш назад у пароль – ніяк. Перевірка пароля відбувається наступним чином: при вході в систему користувач вводить пароль, код отримує цей пароль та використовує бібліотеку хешування яка звіряє отриманий пароль з існуючим хешем. Якщо з даного паролю отримується хеш, ідентичний наявному, це означає, що пароль введено правильний, і система надає користувачу токен для доступу до платформи.

Ще одним покращенням з погляду безпеки та простоти використання є уніфікований формат відповіді на запити. Створена модель яка інформує про успішність операції, містить запитану інформацію, повідомлення помилки, якщо є, код помилки та помилки валідації. Незалежно від того, що відбудеться з системою, користувач отримає структуровану відповідь. Тобто, при розробці програми-клієнта, буде достатньо створити одну модель, що буде отримувати відповідь від серверу. Наявність інформації про успішність операції спростить перевірку даних на додатку-клієнті.

Останнім кроком розробки є створення користувацького інтерфейсу, доступ до якого користувачі зможуть отримати через веббраузер.

Розробка клієнтської частини розпочалась зі створення елементів безпеки, таких як перехоплювач запитів, який приєднує JWT токен до запиту для аутентифікації сервером, та двох блокаторів маршрутизації. Перший блокатор перевіряє чи користувач зареєстрований, і якщо ні, повертає його на сторінку входу. Код описаного блокатора наведений у лістингу 3.7.

Лістинг 3.7 – Код блокатора аутентифікації

```
export const authGuard: CanActivateFn = (route, state) => {
const authService = inject(AuthService);
const router = inject(Router);

if (!authService.isLoggedIn()) {
router.navigate(['/login']);
return false;
}

return true;
};
```

кінець лістингу 3.7

Другий блокатор перевіряє відповідність ролі користувача до запитуваної сторінки. Кожна сторінка має визначений перелік допущених ролей, і якщо роль користувача не збігається з жодною, що є у списку, користувач перенаправляється на сторінку, що повідомляє його про заборону доступу до запитуваної сторінки. Таким чином користувач з правами рекрутера не зможе зайти на панель адміністратора. Код блокатора за ролями наведений нижче, у лістингу 3.8.

Лістинг 3.8 – Код блокатора за ролями

```
export const roleGuard: CanActivateFn = (route, state) => {
const authService = inject(AuthService);
const router = inject(Router);
const expectedRoles = route.data['roles'] as string[];
const userRole = authService.getUserRole()?.toLowerCase();

if (!userRole || !expectedRoles.includes(userRole)) {
router.navigate(['/access-denied']);
return false;
```

```
}

return true;
};
```

кінець лістингу 3.8

Наступним кроком стало створення моделей даних, що використовуються при обміні даними з серверною частиною. Основною моделлю є загальна модель (ліст. 3.9), що повертається при будь-якому запиті до серверу.

Лістинг 3.9 – Код загальної моделі

```
export interface ApiResponse<T> {
success: boolean;
data: T;
message?: string;
errorCode?: string;
}
```

кінець лістингу 3.9

Інші моделі відповідають за такі дані як JWT токен, модель реєстрації, що містить у собі пошту та пароль, моделі перегляду звітів, логів, акаунтів адміністратора, користувача та рекрутера, модель вакансії, оновлення вакансії та інші.

Після завершення створення моделей почалось створення сервісів, що використовують дані моделі. Першим створеним сервісом став сервіс авторизації, який оперує токеном, має доступ до точки входу та реєстрації, та надає інформацію про те, чи авторизований користувач та про його роль у системі. Інші сервіси виконують запити до решти ендпоїнтів серверу, таких як отримати дані про рекрутера, користувача, отримати усі вакансії, чи тільки ті, що пов’язані з певним користувачем чи рекрутером.

Після реалізації сервісів, що надають змогу отримувати та надсилати дані до сервера, потрібно було створити компоненти, які будуть презентувати інформацію користувачу.

Першим розробленим компонентом став заголовок, який використовувався на кожній сторінці сайту, окрім сторінок реєстрації та входу. Цей компонент надає можливість переходу на головну сторінку чи сторінку профілю.

Також створювались компоненти навігатори. На основі ролі користувача вони визначають яку сторінку показувати. Це потрібно для того, аби шлях у системі до декількох сторінок був однаковим, що унеможливлює користувачем запит для отримання сторінки, доступу до якої він не має. В поєднанні з блокатором за ролями це створює надійний захист від отримання доступу до захищених даних. Також, були створені сторінки профілів для рекрутера та користувача, сторінки з вакансіями та реалізованою можливістю фільтрування наявних вакансій за заголовком, місцем роботи чи країною. Окремо створені сторінки, що інформують про відсутність запитаної сторінки чи заборону доступу до неї.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування системи розпочалось зі створення окремого проєкту який містить тести для одного з сервісів. Новий проєкт дозволив проводити інтеграційні тести, підключаючи реальну базу даних, ідентичну до існуючої, та використовувати методи сервісу для оперування інформацією. Інтеграційні тестування дозволяють перевірити правильність роботи сервісу з реальною базою даних.

Тестування відбувається у три етапи. Перший етап це аранжування. Під час цього етапу створюються умови для тестування. Тобто заповнюється база даних тестовими даними, визначаються очікувані результати, такі як кількість отриманих об’єктів, чи об’єкт для порівняння. Після чого розпочинається етап дії, де викликається метод, що тестується, та записується його результат роботи. Третій етап – етап перевірки, де результат виконання методу звіряється з очікуваними значеннями. Якщо тест визначає, що є різниця між явним

результатом та очікуваним, це означає, що метод працює не так, як очікувалось, отже потрібен аналіз причини та пошук рішення.

Таким чином тестування забезпечує відповідність методів очікуванням, що виключає неправильну роботу програми від самого початку розробки.

Виправлені, застосовані при тестуванні одного сервісу, були перенесені на інші сервіси. Тобто тестування одного сервісу вплинуло на усю програму.

Інтеграційні тести зазвичай використовуються під час розробки додатку. Для тестування уже готової програми існують файли з власноруч прописаними запитами до програми.

У даному випадку тестування містить 6 файлів, які тестують окремі частини програми, такі як адмін-частину, чати, повідомлення, рекрутерів, користувачів та вакансії.

Ці файли симулюють запит до програми, ніби від реального користувача, дозволяючи протестувати усі компоненти одночасно. Приклад такого файлу наведений у лістингу 3.10.

Лістинг 3.10 – Фрагмент тестування з файлу рекрутерів

```
### Register new recruiter
POST {{baseUrl}}/recruiters
Content-Type: application/json

{
  "email": "recruiter12@example.com",
  "password": "RecruiterPass"
}
### Login recruiter
POST {{baseUrl}}/auth/login
Content-Type: application/json
{
  "email": "recruiter12@example.com",
  "password": "newpass"
}
```

кінець лістингу 3.10

У лістингу вище представлені запити для реєстрації та входу рекрутера.

Запити вимагають надсилання даних, тому структура тесту містить поля пошти та паролю. Усі інші файли містять подібну структуру, з відмінними полями або без них, в залежності від вимог контролера до якого надсилається запит.

Конфігурування додатку включало в себе налаштування ORM, аби вона сприймала складені назви властивостей моделей як слова розділені нижньою рискою, та сприймала власний тип переліку як текст, а не числа.

Також налаштовувалось переведення об’єктів мови програмування у JSON.

Підключалась валідація та налаштовувалось зчитування токенів, що генеруються при вході з правильними даними. Також додані ролі до токена, для розділення прав доступу між звичайними користувачами, рекрутерами та адміністрацією.

3.3 Розробка бази даних

Розробка бази даних розпочинається з діаграми взаємодії сутностей, яка була розроблена у другому розділі. На її основі складається SQL код, який надалі використовується для внесення змін у базу даних.

Основними таблицями є таблиці користувачів, рекрутерів, адміністраторів, повідомлень, чатів, вакансій, відгуків та скарг. Усі інші є допоміжними, які або формують зв’язки багато-до-багатьох, або слугують для логування виконаних дій. Приклад SQL коду для створення таблиць рекрутерів та адміністраторів наведений у лістингу 3.11.

Лістинг 3.11 – Фрагмент SQL коду для таблиць рекрутерів та адміністрації

```
CREATE TABLE "recruiters" (
  "recruiter_id" uuid PRIMARY KEY,
  "first_name" varchar(255) NOT NULL,
  "last_name" varchar(255) NOT NULL,
  "email" varchar(255) NOT NULL,
  "company" varchar(255) NOT NULL,
  "phone_number" char(12) NOT NULL,
  "notes" varchar(255),
  "created_at" timestamp NOT NULL
```

```
);

CREATE TABLE "admins" (
  "admin_id" INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  "first_name" varchar(255) NOT NULL,
  "last_name" varchar(255) NOT NULL,
  "role" bit NOT NULL DEFAULT B'1',
  "created_at" timestamp NOT NULL
);
```

кінець лістингу 3.11

У лістингу можна помітити, що кожна таблиця містить поле *created_at*, яке слугує елементом контролю та збору інформації для логування. У разі виникнення якихось проблем, час створення елементів допоможе звузити коло пошуку причин. Також таблиця адміністраторів містить стовпець з типом bit та початковим значенням 1. Під час створення цієї таблиці виявилося, що Redis DB не конвертує значення самостійно, тому виникала помилка присвоєння типу bit значення типу integer. Для виправлення помилки довелось додати до коду явне перетворення в тип bit (B’1’).

Також важливо звернути увагу на зв’язки між таблицями, що наведені у лістингу 3.12.

Лістинг 3.12 – Фрагмент SQL коду зі зв’язками

```
ALTER TABLE "vacancies" ADD FOREIGN KEY ("recruiter_id")
REFERENCES "recruiters" ("recruiter_id");
ALTER TABLE "reports" ADD FOREIGN KEY ("recruiter_id") REFERENCES
"recruiters" ("recruiter_id");
ALTER TABLE "reports" ADD FOREIGN KEY ("user_id") REFERENCES
"users" ("user_id");
ALTER TABLE "messages" ADD FOREIGN KEY ("user_id") REFERENCES
"users" ("user_id");
ALTER TABLE "messages" ADD FOREIGN KEY ("recruiter_id") REFERENCES
"recruiters" ("recruiter_id");
ALTER TABLE "messages" ADD FOREIGN KEY ("chat_id") REFERENCES
"chats" ("chat_id");
ALTER TABLE "chats" ADD FOREIGN KEY ("user_id") REFERENCES "users"
("user_id");
ALTER TABLE "chats" ADD FOREIGN KEY ("recruiter_id") REFERENCES
"recruiters" ("recruiter_id");
```

```
ALTER TABLE "admin_reports" ADD FOREIGN KEY ("admin_id")
REFERENCES "admins" ("admin_id");
ALTER TABLE "admin_reports" ADD FOREIGN KEY ("report_id")
REFERENCES "reports" ("report_id");
ALTER TABLE "logs" ADD FOREIGN KEY ("admin_id") REFERENCES
"admins" ("admin_id");
ALTER TABLE "admin_registered_accounts" ADD FOREIGN KEY
("admin_id") REFERENCES "admins" ("admin_id");
ALTER TABLE "admin_registered_accounts" ADD FOREIGN KEY
("registered_account_id") REFERENCES "registered_accounts"
("registered_account_id");
```

кінець лістингу 3.12

У лістингу можна побачити, що кожна таблиця містить хоча би один зв’язок з іншою таблицею, що свідчить про якісний підхід до проєктування бази даних. Деякі таблиці містять зв’язки одночасно з таблицею користувачів та таблицею рекрутерів. В цьому випадку це означає, що один з цих зв’язків буде заповнений, а інший буде пустим. Ці зв’язки створені для відслідковування людини, що виконала дію. Можна було б об’єднати користувачів та рекрутерів. Тоді б зв’язок був один, але об’єднання двох різних сутностей суперечить принципам нормалізації.

Під час розробки, виявилось, що дизайн бази даних був не досконалим, та потребував змін.

Було додано дві нові таблиці: unified_users та user_authentication. Перша використовується для спрощеного входу користувачів у систему, де ідентифікація відбувається за допомогою однієї таблиці, а не трьох. Друга таблиця зберігає паролі користувачів, що дозволяє відокремити чутливу інформацію від загальної.

3.4 Захист інформаційно-комп’ютерної системи

Для захисту додатка використано широкий перелік засобів, від хешування даних та паролів, до створення власних обробників помилок та реалізація доступу до даних на базі ролей.

Одним із базових елементів безпеки є захист облікових даних користувачів, зокрема їхніх паролів. Для цього використано надійний алгоритм хешування – BCrypt, який реалізовано з використанням алгоритму SHA-384 та кількістю ітерацій у 12. Це дозволяє значно ускладнити процес зворотного відновлення паролю, навіть у випадку викрадення хешів. На відміну від простих або «поверхневих» хешів, таких як MD5 або SHA1, BCrypt є адаптивним – тобто з часом можна збільшувати кількість ітерацій, підвищуючи рівень безпеки у відповідності до зростання обчислювальних потужностей.

Для автентифікації та авторизації користувачів у системі реалізовано механізм токенізації, зокрема з використанням JWT (JSON Web Token). Токени дозволяють ідентифікувати користувача під час кожного запиту без необхідності повторного введення логіну та пароля, а також забезпечують зручну реалізацію контролю доступу на основі ролей (Role-Based Access Control). Це дозволяє точно визначати, які дії або ресурси доступні конкретному користувачу.

У системі передбачено декілька ролей користувачів, серед яких: користувач, рекрутер, адміністратор та комбіновані ролі (користувач-адміністратор, користувач-рекрутер, рекрутер-адміністратор).

Для зручності та безпеки кожен обліковий запис може мати лише одну активну роль. У разі необхідності надання доступу до функціоналу, що стосується двох ролей, створюються окремі комбіновані ролі. Таким чином, чітко розмежовуються права доступу до різних частин системи, що дозволяє зменшити ризики несанкціонованого використання функцій або витоку даних.

Ще одним важливим аспектом безпеки є обробка помилок. У системі реалізовано власний механізм обробки винятків, який дозволяє контролювати, яку інформацію отримує користувач у відповідь на ту чи іншу помилку. Це запобігає розкриттю внутрішніх деталей реалізації системи, таких як структура бази даних, шляхи до серверних ресурсів або специфіка внутрішніх API. Оскільки така інформація може бути використана зловмисниками для атак, її приховування є одним із найважливіших заходів безпеки.

Крім того, була реалізована уніфікована модель відповіді, яка стандартизує формат усіх відповідей API. Це дозволяє не лише спростити обробку запитів на стороні клієнта, а й приховати технічні подробиці у разі виникнення помилок. Наприклад, у відповідь на помилку система повертає лише загальне повідомлення про збій із відповідним кодом статусу, не розкриваючи стек викликів або точне місце виникнення помилки у коді.

Таким чином, у реалізованій інформаційно-комп’ютерній системі було застосовано комплексний підхід до забезпечення безпеки, що охоплює як захист даних, так і запобігання несанкціонованому доступу та витоку інформації. Комбінація сучасних технологій шифрування, гнучкої системи ролей, контрольованої обробки помилок та стандартизованого обміну даними дозволяє гарантувати високий рівень безпеки та стабільності функціонування системи у реальних умовах експлуатації.

3.5 SEO інформаційно-комп’ютерної системи

Для забезпечення високої видимості створеної інформаційно-комп’ютерної системи у пошукових системах була реалізована стратегія пошукової оптимізації (SEO). Основною метою цих заходів є покращення індексування вебресурсу пошуковими ботами, підвищення позицій у результатах видачі, а також забезпечення зручності використання для кінцевих користувачів.

Одним із ключових елементів реалізації SEO стало впровадження серверного рендерингу сторінок. Цей підхід дозволяє формувати повноцінний HTML-код ще на стороні сервера перед його передачею клієнту. Таким чином, пошукові системи отримують вже готовий до індексації контент, що значно покращує сприйняття ресурсу пошуковими алгоритмами. SSR особливо ефективний порівняно з клієнтським рендерингом, де контент створюється JavaScript-скриптами безпосередньо у браузері користувача, що може ускладнити або уповільнити процес індексації.

Крім того, під час розробки інтерфейсу системи було застосовано семантичну HTML-розмітку. Це означає, що структура сторінок сформована з використанням тегів, які мають чітке смислове значення. Такий підхід не лише сприяє кращому розумінню структури контенту пошуковими системами, а й покращує доступність сайту для різних користувачів, зокрема тих, хто використовує допоміжні технології, такі як екранні читачі.

Особлива увага приділялася параметрам доступності, які є важливими не лише з точки зору етики та інклюзії, але й з позиції сучасних вимог SEO. Було реалізовано підтримку атрибутів ARIA (Accessible Rich Internet Applications), налаштовано коректні текстові альтернативи для зображень, забезпечено логічну навігацію за допомогою клавіатури, а також оптимізовано контрастність кольорів для покращення сприйняття інформації користувачами з вадами зору. Усі ці кроки позитивно впливають на показники якості сторінки, що враховуються пошуковими системами при ранжуванні.

Також варто зазначити, що для покращення SEO були реалізовані мета-теги для кожної сторінки, які динамічно формуються на основі її вмісту. Це дозволяє адаптувати мета-інформацію під ключові запити користувачів і забезпечити релевантність сторінки пошуковим запитам. Додатково використовувались структуровані дані, що дозволяють пошуковим системам краще інтерпретувати контент сторінки, наприклад, розпізнавати відгуки, події, продукти тощо.

У контексті оптимізації швидкості завантаження сторінок, яка також впливає на SEO, були застосовані сучасні підходи до мінімізації ресурсів: стиснення CSS та JavaScript файлів, а також використання кешування. Усі ці заходи сприяють зменшенню часу завантаження сторінки, що позитивно впливає як на користувацький досвід, так і на оцінку сторінки пошуковими системами.

Таким чином, реалізовані заходи дозволили створити оптимізовану з точки зору SEO інформаційно-комп’ютерну систему, яка відповідає сучасним стандартам веброзробки, забезпечує високу доступність, швидкість роботи та ефективну індексацію в пошукових системах.

Висновки до розділу 3

У межах реалізації проєкту створено повнофункціональний вебдодаток із серверною та клієнтською частинами. Основна мета – забезпечити стабільну, безпечну та масштабовану платформу для обробки, зберігання й обміну даними між користувачами з різними рівнями доступу. На сервері реалізовано логіку роботи з базою даних через ORM, створено структуровану систему моделей з валідацією вхідних даних, що підвищує надійність. Обробка помилок супроводжується уніфікованими повідомленнями. Впроваджено сервіси для управління даними відповідно до трьох рівнів доступу: користувачі, модератори, адміністратори.

Особливу увагу приділено безпеці: реалізовано аутентифікацію й авторизацію на основі JWT, паролі шифруються з використанням BCrypt і SHA384. Усі відповіді сервера мають уніфікований формат, що спрощує взаємодію з клієнтською частиною. Фронтенд розроблено з урахуванням модульності та повторного використання коду, реалізовано захист маршрутів, моделі для взаємодії з API та набір оптимізованих інтерфейсних компонентів для основної функціональності. Також додано механізми кешування даних на клієнтському боці для зменшення навантаження на сервер і підвищення швидкодії.

Проведено інтеграційне тестування, налаштовано середовище для розгортання, організовано логування подій і реалізовано обробку JSON та токенів для ефективного обміну даними між клієнтом і сервером. Крім того, додано систему моніторингу продуктивності, яка дозволяє в режимі реального часу відстежувати стан сервісів, реагувати на збої та проводити аналіз продуктивності. У результаті створено сучасний, масштабований, безпечний вебдодаток, готовий до подальшого розширення, інтеграції з іншими системами та адаптації під потреби бізнесу.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи здійснено повний цикл розробки повнофункціональної вебплатформи із серверною та клієнтською частинами для пошуку роботи в IT-сфері, що включав аналітичну, проєктну та практичну частини.

На першому етапі проведено аналіз існуючих рішень, виявлено ключові недоліки: низький рівень безпеки, перевантаженість функціоналом, проблеми з інтерфейсом, відсутність зручної комунікації між роботодавцем і пошукачем. Визначено напрями вдосконалення, яких дотримано під час створення нової платформи.

У другому розділі сформовано функціональні та нефункціональні вимоги, побудовано UML-діаграми та карти сайту для різних ролей користувачів. Створено дизайн ключових сторінок у Figma, що забезпечило зручність і логічність інтерфейсу.

Обґрунтовано вибір технологій: ASP.NET – для серверної частини, Angular з Angular Material – для клієнтської, Redis DB – як СУБД, Docker Compose – для керування середовищем.

Практична частина охоплює реалізацію повнофункціонального вебзастосунку: серверну логіку, взаємодію з базою даних, систему аутентифікації та авторизації з JWT, шифрування паролів, валідацію та обробку помилок. На клієнтській стороні створено компоненти для роботи з профілями, вакансіями, чатами, реалізовано захист маршрутів відповідно до ролей.

Проведено тестування основних модулів, налаштовано логування, обробку JSON-даних, написано інтеграційні тести для перевірки взаємодії сервісів із базою, виконано ручне тестування шляхом імітації дій користувача.

У результаті реалізовано зручну, безпечну та масштабовану платформу для пошуку роботи в IT-сфері, що враховує недоліки аналогів і має потенціал до подальшого розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ammattikorkeakoulut – Theseus. URL: https://www.theseus.fi/bitstream/handle/10024/499566/sayeem_mdmanwar.pdf?sequence=2&isAllowed=y (date of access: 06.01.2025).

2. ArXiv.org e-Print archive. URL: https://arxiv.org/pdf/2403.11769 (date of access: 09.01.2025).

3. ASP.NET core, an open-source web development framework.NET. Microsoft. URL: https://dotnet.microsoft.com/en-us/apps/aspnet (date of access: 17.04.2025).

4. Birla Vishvakarma Mahavidyalaya. URL: https://bvmengineering.ac.in/NAAC/Criteria1/1.3/1.3.4/Online_Job_Portal_416_419_435.pdf (date of access: 05.01.2025).

5. Contributors to Wikimedia projects. Bcrypt – wikipedia. URL: https://en.wikipedia.org/wiki/Bcrypt (date of access: 19.04.2025).

6. Contributors to Wikimedia projects. Python (programming language) – Wikipedia. URL: https://en.wikipedia.org/wiki/Python_(programming_language) (date of access: 14.04.2025).

7. Daffodil Institute of IT. URL: https://diit.edu.bd/media/attachment/Imran_180056_Final_Book.pdf (date of access: 03.01.2025).

8. GeeksforGeeks. Unified modeling language (UML) diagrams. URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/ (date of access: 12.01.2025).

9. GilPress. Internet traffic from mobile devices stats (2024). Whats the Big Data. URL: https://whatsthebigdata.com/mobile-internet-traffic/ (date of access: 03.01.2025).

10. International Journal & Research Paper Publisher IJRASET. URL: https://www.ijraset.com/best-journal/online-job-portal-finding-jobs-in-the-covid19-pandemic (date of access: 06.01.2025).

11. IRJET – International Research Journal of Engineering and Technology. URL: https://mail.irjet.net/archives/V6/i4/IRJET-V6I433.pdf (date of access: 06.01.2025).

12. Online Job Portal – Python Project with Source Code. URL: https://data-flair.training/blogs/online-job-portal-python-project/ (date of access: 03.01.2025).

13. JQuery. URL: https://jquery.com/ (date of access: 20.04.2025).

14. Microsoft SQL server. URL: https://www.microsoft.com/en/sql-server/ (date of access: 22.04.2025).

15. MongoDB – the world's leading modern database. URL: https://www.mongodb.com/ (date of access: 22.04.2025).

16. Redis – the real-time data platform. URL: https://redis.io/ (date of access: 22.04.2025).

17. SQLite Home Page. URL: https://sqlite.org/index.html (date of access: 22.04.2025).

18. Welcome! – The Apache HTTP Server Project. URL: https://httpd.apache.org/ (date of access: 22.04.2025).

19. Welcome to IJNIET. URL: https://www.ijniet.org/wp-content/uploads/2024/05/391.pdf (date of access: 09.01.2025).

ДОДАТКИ

Додаток А – Тенденції розвитку IT ринку в Україні та світі

Литвинюк М. Тенденції розвитку IT ринку в Україні та світі/ Литвинюк М., Самчук Л./ Матеріали міжнародної науково-практичної конференції «Цифрова трансформація: виклики та стратегії» 25 лютого 2025 р., м. Луцьк: ЛНТУ, 2025.с. 161-163. https://dtcs.lntu.edu.ua/tezy-konferencziyi/

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
Lutsk National Technical University (Ukraine), National Technical University "Kharkiv Polytechnic Institute" (Ukraine), National Technical University "Dnipro Polytechnic" (Ukraine), Lublin University of Technology (Poland), Ahlia University (Bahrain), Polytechnic Institute of Braganca (Portugal), Vytautas Magnus University (Lithuania), Marie Curie-Sklodowska University (Poland), OWL University of Applied Sciences and Arts (Germany), Batumi Navigation Teaching University (Georgia), Dunarea de Jos University of Galati (Romania), Western Science Center (Ukraine), Akkon University of Human Sciences (Germany)

МАТЕРІАЛИ
Міжнародної науково-практичної конференції
«Цифрова трансформація: виклики та стратегії»
https://dtcs.lntu.edu.ua/

м. Луцьк, Україна
25 лютого 2025 року

MATERIALS of the
International scientific and practical conference
"Digital Transformation: Challenges and Strategies"
https://dtcs.lntu.edu.ua/

Lutsk, Ukraine
February 25, 2025

Матеріали міжнародної науково-практичної конференції
«Цифрова трансформація: виклики та стратегії»

Максим ЛИТВИНЮК
здобувач вищої освіти факультету комп'ютерних та інформаційних технологій
Луцький національний технічний університет, Україна
Людмила САМЧУК
к.т.н., доцент
Луцький національний технічний університет, Україна

ТЕНДЕНЦІЇ РОЗВИТКУ IT РИНКУ В УКРАЇНІ ТА СВІТІ

IT-сектор є однією з найбільш динамічних та перспективних галузей на глобальному рівні. Згідно дослідження Statista [4], клієнти все більше шукають IT-рішення для підвищення ефективності власного бізнесу.

Ситуація на глобальному IT-ринку. Очікується ріст в США через підвищення попиту на хмарні обчислення та цифрову трансформацію. Сектори медицини та фінансів є найбільшими чинниками росту. На середньому сході IT-ринок повинен зростати через потребу в хмарних обчисленнях та підвищення попиту державних ініціатив. В Латинській Америці ринок IT-послуг буде розвиватись через потребу в кібербезпеці та попит на цифрову трансформацію.

Український IT-ринок розширився. Дослідження DOU [2] вказує, що найбільш помітним ріст кількості працівник був у період пандемії. Кількість ФОПів зросла з 77 до 275 тисяч, що свідчить про розвиток IT-сектору України.

Тип IT-компаній в Україні також змінився. У 2024 році частка спеціалістів, що працюють в продуктових компаніях (45%) перевищила частку тих, хто працює в сервісних (36%). Також зростають аутстафінгові компанії: станом на 2024 рік – 12%.

Стан IT ринку під час війни. Львівський IT Кластер представив дослідження про стан IT-індустрії [3]. З 2023 року рівень експорту IT-послуг продовжує зменшуватись. Загальний обсяг експорту IT-послуг становитиме $6,3 – 6,45 млрд. 4.4% ВВП України на 2024 рік це IT-індустрія, що менше ніж 2023 року. Сектор забезпечує 663 000 – 668 000 робочих місць.

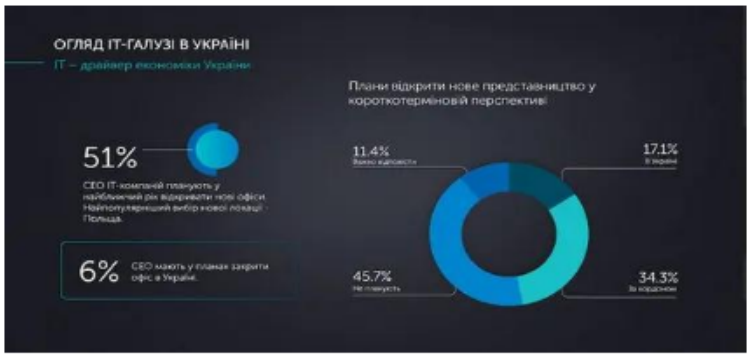

Рисунок 1 – Розподіл планів щодо відкриття нових представництв

Асоціація IT Ukraine та агенція Top Lead дослідили [4], що частка IT в ВВП у 2024 році знизилась відносно 2022 року. За 2022-2023 роки IT-сектор згенерував $14,06 млрд виручки.

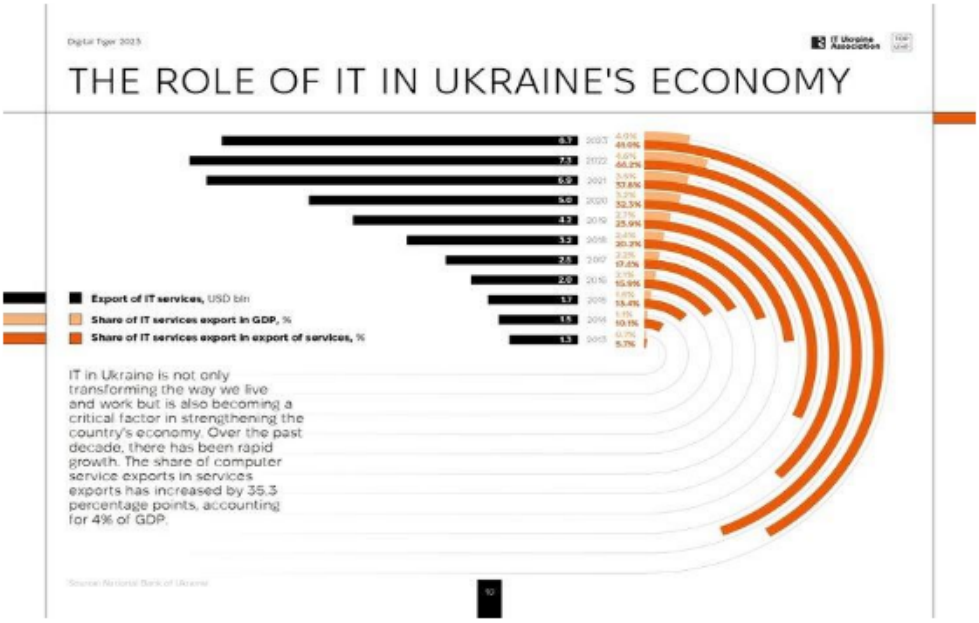

Рисунок 2 – Графік впливу IT сектору на експорт з 2013 по 2023 роки

Попри просідання обсягу доходу у 2023 році за 2022-2023 роки IT-сектор сплатив 68,1 млрд грн податків, що на 11,5% більше, ніж попереднього року. Експорт IT-послуг зріс до 13,2% за 2023 рік, уступаючи лише агросектору.

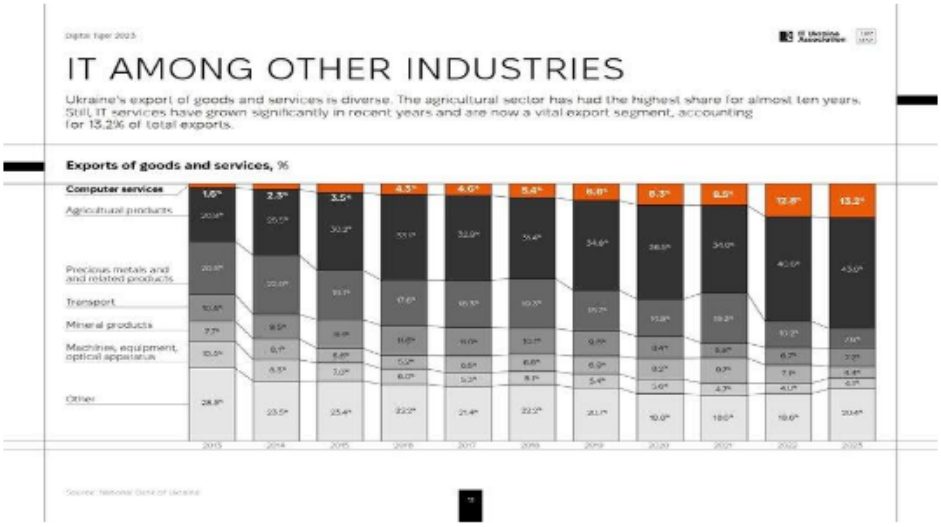

Рисунок 3 – Розподіл експорту України за період з 2013 по 2023 роки

Статистичні дані та графіки показують, що IT-сектор продовжує розвиватись як в Україні, так і у світі. Все більше країн фокусуються на розвитку цієї сфери, адже це сфера надання послуг, що має великий потенціал як у розвитку, так і в прибутках.

Приклад України показує, що IT-сектор може розвиватись та серйозно підтримувати економіку країни навіть у період війни, забезпечуючи розвиток країни, генерацію нових робочих місць, оптимізацію процесів.

Серйозною перевагою IT-сектору є відносна легкість експорту, коли не вимагається залучення десятків спеціалістів з логістики для перевезення товару за кордон.

Прогнозується ріст IT-сектору й надалі, що забезпечить подальший розвиток інших економічних секторів країн світу.

У майбутньому це може призвести до цифровізації усіх аспектів життя людини та створення нових робочих місць.

Список використаних джерел:

1. IT services - worldwide | statista market forecast. *Statista*. URL: https://www.statista.com/outlook/tmo/it-services/worldwide#analyst-opinion (date of access: 18.02.2025).
2. Портрет айтівця 2024. Як змінилося українське IT за 10 років. *DOU*. URL: https://dou.ua/lenta/articles/portrait-2024/ (дата звернення: 18.02.2025).
3. Гмиря А. Українські айтівці у 2024 році: падіння обсягів експорту, зарплати, міграція та підтримка ЗСУ. *The Page*. URL: https://thepage.ua/ua/it/yak-sebe-vidchuvaye-rinok-it-ukrayini-v-2024-roci (дата звернення: 19.02.2025).
4. IT-індустрія в цифрах: найцікавіші дані з дослідження Digital Tiger. *Mind.ua*. URL: https://mind.ua/publications/20270953-it-industriya-v-cifrah-najcikavishi-dani-z-doslidzhennya-digital-tiger (дата звернення: 19.02.2025).

Дмитро Омельчук,
студент
Катерина Мельник,
к.т.н., доцент
Павло Мельник,
аспірант
Луцький національний технічний університет, Україна

АНАЛІЗ МЕТОДІВ ОЦІНКИ МАШИННОГО ПЕРЕКЛАДУ ТА ФАКТОРІВ, ЩО ВПЛИВАЮТЬ НА ЇХ ВИБІР

В епоху глобалізації та стрімкого розвитку інформаційних технологій машинний переклад відіграє важливу роль у забезпечені швидкого та доступного перекладу контенту різними мовами [5]. Однак, залишаються актуальними питання об'єктивної та ефективної оцінки якості перекладу. Вибір оптимального методу оцінки залежить від багатьох факторів, включаючи цілі оцінки, доступні ресурси, мовну пару, специфіку домену та вимоги до точності. Неправильний вибір методу може призвести до некоректних висновків щодо якості перекладу, що, в свою чергу, може негативно вплинути на прийняття рішень щодо використання та вдосконалення систем машинного перекладу [1].

Метою роботи є аналіз існуючих методів оцінки машинного перекладу, а також виявлення ключових факторів, що впливають на вибір найбільш відповідного методу оцінки. Проведено їх порівняльний аналіз за різними критеріями, такими як точність, швидкість та об'єктивність і т. д. (таблиця 1.) Розглянуті такі метрики, як BLEU, NIST, METEOR, ROUGE, TER, chrF++, COMET. На результати оцінки істотно впливають нечіткі збіги (fuzzy matches [2-3]), що використовуються в системах автоматизованого перекладу.

Додаток Б - Вплив віддаленої роботи на продуктивність розробників після COVID-19

Миколи Гоголя. Психолого-педагогічні науки. 2022. № 2. С. 111–120. https://doi.org/10.31654/2663-4902-2022-PP-2-111-120.

4. Хміляр О. Ф. Психологія бойової мотивації воїна. *Вісник Національного університету оборони України*. 2022. № 2 (66). С. 121–131. https://doi.org/10.33099/2617-6858-2022-66-2-121-131.

5. Хміляр О. Ф. Команда серед команд. Чому мала команда є ефективною бойовою одиницею? *Науковий вісник Ужгородського національного університету. Психологія*. 2024. Вип. 1. С. 53–59. https://doi.org/10.32782/psy-visnyk/2024.1.10.

6. Красницька О. В. Постать сучасного викладача: справжність і фейковість. Імідж сучасного педагога. 2023. № 3 (210). С. 11–17. https://doi.org/10.33272/2522-9729-2023-3(210)-11-17.

УДК 004.5

ВПЛИВ ВІДДАЛЕНОЇ РОБОТИ НА ПРОДУКТИВНІСТЬ РОЗРОБНИКІВ ПІСЛЯ COVID-19

Литвинюк Максим Віталійович
Луцький національний технічний університет, здобувач вищої освіти факультету комп'ютерних та інформаційних технологій, max.litvinyuck553@gmail.com
Самчук Людмила Михайлівна
Луцький національний технічний університет, кандидат тех. наук, доцент кафедри прикладної механіки та мехатроніки, Samchuk204@gmail.com

THE IMPACT OF REMOTE WORK ON DEVELOPERS' PRODUCTIVITY POST-COVID-19

Lytvyniuk Maksym Vitaliyovych
Lutsk National Technical University, higher education student of the Faculty of Computer and Information Technologies, max.litvinyuck553@gmail.com
Samchuk Liudmyla Mykhailivna
Lutsk National Technical University, PhD inTechnical Sciences, Associate Professor, Department of Applied Mechanics and Mechatronics Samchuk204@gmail.com

The COVID-19 pandemic significantly accelerated the adoption of remote work across various industries, with the IT sector being one of the most adaptable.

This shift has transformed the global job market, enabling companies to hire talent from virtually anywhere and allowing IT professionals to access a broader range of opportunities. The paper explores the impact of remote work on software developers' productivity in the post-COVID-19 era. It analyzes the shift from traditional office environments to remote and hybrid work models, highlighting changes in work habits, collaboration patterns, and performance metrics. The study considers both the benefits and challenges of remote work, including flexibility, autonomy, communication barriers, and work-life balance. Data is drawn from recent surveys and case studies from tech companies to assess how remote work has reshaped the software development landscape.

***Keywords**: remote work, developer productivity, post-COVID-19, software development, hybrid work, collaboration, work-life balance, communication, performance metrics, tech industry.*

У зв'язку з різким поширенням раніше малопоширеного віддаленого формату праці, зумовленим пандемією COVID-19 та подальшими структурними змінами на ринку праці, виникає нагальна потреба в оцінці ефективності працівників, що працюють поза межами традиційного офісного середовища. Метою даного дослідження є аналіз динаміки змін продуктивності праці в умовах віддаленої роботи на основі відкритих джерел інформації, а також виявлення можливих закономірностей між форматом зайнятості та ефективністю діяльності працівників у різних секторах економіки.

Згідно з даними аналітичного ресурсу *Beyond the Numbers* [1], після масового переходу на віддалений формат роботи спостерігалося суттєве зростання кількості працівників, які продовжили працювати дистанційно навіть після завершення активної фази пандемії. Зокрема, в усіх основних секторах економіки зафіксовано зростання чисельності віддалених співробітників, при цьому в чотирьох найбільших галузях кількість таких працівників збільшилася більш ніж на 30%. Навіть після стабілізації епідеміологічної ситуації понад третина працівників у цих секторах залишилася на дистанційному форматі роботи.

Рисунок 1 – Графік у відсотках кількості віддалених працівників у великих секторах

Крім того, дослідження вказує на загальне підвищення рівня продуктивності у більшості галузей, де впроваджено віддалену форму зайнятості. У деяких випадках зростання коефіцієнта продуктивності сягало 5%. Виявлено також позитивну кореляцію між зростанням продуктивності праці та збільшенням кількості працівників, які здійснюють свою професійну діяльність дистанційно.

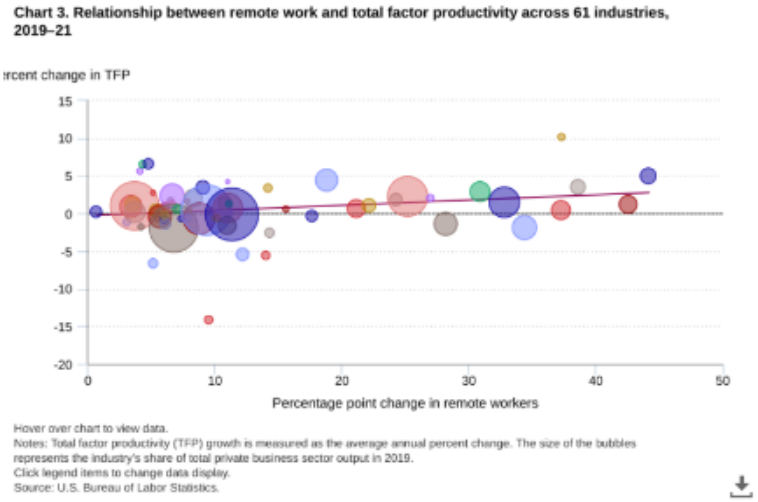

Рисунок 2 – Залежність між віддаленим форматом роботи та коефіцієнтом продуктивності

Додаткову інформацію щодо впливу віддаленої роботи на ефективність праці надає спільне дослідження, проведене фахівцями Гарвардського університету та Університету штату Іллінойс [2]. В рамках дослідження було зібрано дані шляхом анкетування та тестування працівників різних компаній. Результати показали, що при віддаленому форматі роботи середній рівень продуктивності зріс на 33%. Водночас дослідники зазначають, що в перший рік після переходу на дистанційний режим, продуктивність праці знизилася на 10,54% порівняно з допандемічним періодом, що, ймовірно, було зумовлено адаптаційними труднощами та відсутністю належної інфраструктури на початковому етапі переходу.

Аналіз відкритих джерел свідчить про наявність стійкої тенденції до підвищення продуктивності праці в умовах віддаленої зайнятості. Отримані дані дозволяють припустити, що за належної організації робочого процесу дистанційний формат може не лише зберегти, але й підвищити ефективність працівників у ряді галузей. Таким чином, подальше дослідження механізмів адаптації, управління та оптимізації віддаленої роботи є доцільним та актуальним для формування нових стратегій управління персоналом у постпандемічну епоху.

Список використаних джерел

1. Sabrina Pabilonia, Jill Redmond. The Rise in Remote Work since the Pandemic and its Impact on Productivity. Beyond the Numbers. 13, 8. 2024. URL: https://www.bls.gov/opub/btn/volume-13/remote-work-productivity.htm

2. Alexander Bartik, Zoë Cullen, Ed Glaeser, Michael Luca, Christopher Stanton. The Rise of Remote Work: Evidence on Productivity and Preferences from Firm and Worker Surveys. Harvard Business School Working Paper. 20-138. 2023. 14-15 URL: https://www.hbs.edu/ris/Publication%20Files/20-138_eca954c7-dde8-4154-8d7b-5688fd5caf94.pdf

СЕРТИФІКАТ

Кафедра цифрових освітніх технологій

Кафедра інженерії програмного забезпечення

Даний сертифікат засвідчує, що

Литвинюк Максим Віталійович

брав(-ла) участь у

X Міжнародній науково-практичній конференції
з проблем вищої освіти і науки
"Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025)"
загальним обсягом 15 годин (0,5 кредитів ECTS),

яка проводилася 23-24 травня 2025 року у м.Луцьк
на базі Луцького національного технічного університету

Ректор ЛНТУ _____ **Ірина ВАХОВИЧ**

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

№ ІТОНВ-2025-89

Програма конференції: https://itonv.lntu.edu.ua/files/2025/program-itonv-2025.pdf