Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

СЕРВІС ДЛЯ ПОШУКУ РОБОТИ ЗАСОБАМИ C# BLAZOR SERVER APP

JOB SEARCH SERVICE BY MEANS C# BLAZOR SERVER APP TOOLS

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-42
Янчар Олександр Русланович

(підпис)

Керівник:
к.т.н., доцент
Пех Петро Антонович

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 10 » червня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т. Терлецький
« 10 » 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Янчару Олександру Руслановичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Сервіс для пошуку роботи засобами C# Blazor Server App*

Керівник роботи *к.т.н., доцент Пех Петро Антонович*

затверджені наказом закладу вищої освіти від «04» січня 2025 року № 11/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 10.06.2025р.

3. Вихідні дані до роботи *джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз предметної області та постановка завдання
Проектування сервісу пошуку роботи
Реалізація програмного забезпечення
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Діаграма класів
Процес обирання портфоліо для подачі заявки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області та постановка завдання	*Пех П.А., доцент*		
Проектування сервісу пошуку роботи	*Пех П.А., доцент*		
Реалізація програмного забезпечення	*Пех П.А., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	___%		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Аналіз предметної області та постановка завдання*	до 10.02.2025 р.	Виконано
2.	*Проектування сервісу пошуку роботи*	до 02.03.2025 р.	Виконано
3.	*Реалізація програмного забезпечення*	до 02.04.2025 р.	Виконано
4.	*Висновки та пропозиції*	до 10.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 15.04.2025 р.	Виконано
6.	*Формування додатків*	до 02.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 10.05.2025 р.	Виконано
8.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	до 15.05.2025 р.	Виконано
9.	*Нормоконтроль*	до 30.05.2025 р.	Виконано
10.	*Інструментальна перевірка на академічний плагіат*	до 03.06.2025 р	Виконано
11.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ (підпис) Янчар О.Р. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Пех П.А. (прізвище, ініціали)

АНОТАЦІЯ

Янчар О. Р. Сервіс для пошуку роботи засобами C# Blazor Server App. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2023.

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

Перший розділ присвячено аналізу існуючих сервісів пошуку роботи та технологій розробки веб-додатків, обґрунтовано вибір фреймворку Blazor. Проведено огляд популярних платформ (LinkedIn, Indeed.com, Monster.com, Glassdoor та інших).

В другому розділі здійснено проектування архітектури системи, спроектовано базу даних, користувацький інтерфейс та систему безпеки. Розроблено комплект UML-діаграм для опису структури та поведінки системи.

Третій розділ описує практичну реалізацію програмного забезпечення, включаючи розробку серверної та клієнтської частин, інтеграцію системи автентифікації, реалізацію основних функціональних модулів та їх тестування.

Об'єкт – процес розробки веб-сервісу для пошуку роботи з використанням сучасних технологій.

Предмет – методи та технології розробки веб-додатків на базі фреймворку Blazor.

Метою роботи є розробка сучасного веб-сервісу для пошуку роботи з використанням фреймворку Blazor, який забезпечить ефективну взаємодію між кандидатами та роботодавцями.

Ключові слова: веб-сервіс, пошук роботи, Blazor, ASP.NET Core, Entity Framework Core, SignalR, автентифікація, авторизація, кандидати, роботодавці.

ABSTRACTS

Yanchar O. Job search service by means C# Blazor Server App tools. Manuscript.

Qualification work for bachelor's degree in Computer Engineering, speciality 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2023.

The qualification work consists of an introduction, three chapters, conclusions, a list of references and appendices.

The first chapter is devoted to the analysis of existing job search services and web application development technologies, the choice of the Blazor framework is justified. An overview of popular platforms (LinkedIn, Indeed.com, Monster.com, Glassdoor, etc.) is provided.

In the second section, the system architecture is designed, the database, user interface, and security system are designed. A set of UML diagrams was developed to describe the structure and behaviour of the system.

The third section describes the practical implementation of the software, including the development of the server and client parts, the integration of the authentication system, the implementation of the main functional modules and their testing.

Object – the process of developing a web service for job search using modern technologies.

Subject – methods and technologies for developing web applications based on the Blazor framework.

The purpose of the work is to develop a modern web service for job search using the Blazor framework, which will ensure effective interaction between candidates and employers.

Keywords: web service, job search, Blazor, ASP.NET Core, Entity Framework Core, SignalR, authentication, authorisation, candidates, employers.

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ 9
1.1 Аналіз існуючих сервісів пошуку роботи 9
1.2 Огляд технологій розробки веб-додатків 11
1.3 Обґрунтування вибору технології Blazor 14
1.4 Постановка завдання розробки 16
РОЗДІЛ 2 ПРОЕКТУВАННЯ СЕРВІСУ ПОШУКУ РОБОТИ 19
2.1 Архітектура програмного забезпечення 19
2.2 Проектування бази даних 20
2.3 Проектування користувацького інтерфейсу 22
2.4 Розробка діаграм UML 25
2.5 Проектування системи безпеки та авторизації 29
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 32
3.1 Розробка серверної частини 32
3.2 Реалізація клієнтської частини на Blazor 34
3.3 Інтеграція системи авторизації та автентифікації 37
3.4 Реалізація основних функціональних модулів 40
3.4.1 Модуль реєстрації та авторизації 40
3.4.2 Модуль управління профілями 42
3.4.3 Модуль пошуку вакансій 44
3.4.4 Модуль управління вакансіями 49
3.4.5 Система відгуків та комунікації 50
3.5 Тестування розробленого програмного забезпечення 56
3.6 Вимоги до апаратної частини 59
ВИСНОВКИ 60
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 62
ДОДАТКИ 64

ВСТУП

Сучасний ринок праці характеризується динамічним розвитком та постійною трансформацією під впливом цифрових технологій. Процес пошуку роботи та підбору персоналу все більше переміщується в онлайн-середовище, створюючи потребу в ефективних цифрових інструментах для взаємодії між кандидатами та роботодавцями.

Розробка спеціалізованих платформ для пошуку роботи стає необхідністю в умовах зростаючої конкуренції на ринку праці. Такі системи повинні забезпечувати швидкий та релевантний пошук вакансій, зручну комунікацію між учасниками процесу, надійний захист даних та ефективні інструменти для оцінки кандидатів.

Технологія Blazor від Microsoft відкриває нові можливості для створення сучасних веб-додатків, дозволяючи використовувати C# як для серверної, так і для клієнтської розробки. Це забезпечує єдиний підхід до розробки, високу продуктивність та безпеку, що особливо актуально для систем, які працюють з персональними даними користувачів.

Мета роботи полягає у розробці сучасного веб-сервісу для пошуку роботи з використанням фреймворку Blazor, який забезпечить ефективну взаємодію між кандидатами та роботодавцями.

Для досягнення поставленої мети необхідно вирішити наступні завдання. Спочатку слід проаналізувати існуючі рішення та визначити ключові вимоги до системи. Далі потрібно обрати відповідні технології для розробки та обґрунтувати їх вибір.

Після цього треба розробити архітектуру системи. Потім реалізувати основні модулі: реєстрація та авторизація, управління профілями, пошук вакансій, управління відгуками та комунікація.

На завершення провести тестування розробленого програмного забезпечення.

Об’єктом дослідження є процес розробки веб-сервісу для пошуку роботи з використанням сучасних технологій.

Предметом дослідження є методи та технології розробки веб-додатків на базі фреймворку Blazor.

Методи дослідження включають аналіз існуючих рішень, проектування архітектури програмного забезпечення, об’єктно-орієнтоване програмування, тестування програмного забезпечення.

Практичне значення одержаних результатів полягає у створенні повнофункціонального веб-сервісу, який може бути використаний для оптимізації процесів пошуку роботи та підбору персоналу. Розроблена система надає зручні інструменти для публікації вакансій, пошуку кандидатів, проведення співбесід та аналізу ефективності рекрутингу.

Структура роботи включає вступ, три розділи, висновки та список використаних джерел. У першому розділі проведено аналіз предметної області та обґрунтування вибору технологій. Другий розділ присвячено проектуванню архітектури системи. У третьому розділі описано реалізацію основних модулів та результати тестування.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз існуючих сервісів пошуку роботи

Сучасний ринок онлайн-платформ для пошуку роботи представлений численними рішеннями, кожне з яких має свої особливості та функціональні можливості. LinkedIn [1], як провідна професійна мережа, пропонує користувачам не лише пошук вакансій, але й створення професійного профілю, встановлення ділових контактів та участь у професійних спільнотах. Платформа дозволяє роботодавцям переглядати детальні профілі кандидатів, їх рекомендації та досвід роботи, що суттєво спрощує процес підбору персоналу.

Indeed.com [2] зарекомендував себе як глобальний агрегатор вакансій, який збирає інформацію про відкриті позиції з різних джерел, включаючи корпоративні сайти компаній, рекрутингові агентства та спеціалізовані job-портали. Система пропонує розширені можливості фільтрації за різними параметрами: місце розташування, рівень заробітної плати, тип зайнятості, необхідний досвід роботи. Користувачі можуть налаштовувати сповіщення про нові вакансії за обраними критеріями.

Monster.com [3] виділяється серед конкурентів своїми інструментами для створення та оптимізації резюме. Платформа використовує алгоритми машинного навчання для аналізу резюме та надання рекомендацій щодо їх покращення. Сервіс також пропонує функцію автоматичного співставлення навичок кандидата з вимогами роботодавців, що підвищує ефективність пошуку відповідних вакансій.

Glassdoor [4] додає унікальний елемент прозорості до процесу пошуку роботи, надаючи користувачам доступ до відгуків про компанії від колишніх та поточних співробітників. Платформа містить інформацію про корпоративну культуру, рівень заробітних плат, процес співбесіди та можливості кар’єрного росту в різних організаціях. Ця інформація допомагає кандидатам приймати більш зважені рішення щодо потенційних роботодавців.

Work.ua [5], як провідний український сервіс пошуку роботи, адаптований під специфіку local-ринку праці. Платформа пропонує зручний інтерфейс для розміщення вакансій та пошуку роботи, функціонал для проведення онлайн-співбесід, а також можливість створення власного кар’єрного профілю. Сервіс регулярно оновлює базу вакансій та надає статистичні дані щодо тенденцій на ринку праці .

Djinni [6] спеціалізується на IT-секторі та пропонує унікальний підхід до пошуку роботи, де роботодавці самі знаходять відповідних кандидатів за їхніми профілями. Платформа використовує систему рейтингів та верифікації навичок, що підвищує довіру між учасниками процесу пошуку роботи. Сервіс також надає можливість анонімного пошуку роботи для кандидатів, які ще працюють.

HeadHunter [7] вирізняється розвиненою системою професійного тестування та оцінки компетенцій. Платформа пропонує інструменти для проведення відеоспівбесід, створення брендованих сторінок компаній та публікації контенту про корпоративну культуру. Сервіс також надає розширену аналітику для роботодавців щодо ефективності їхніх вакансій та активності кандидатів.

Dice [8] фокусується на технологічному секторі та пропонує спеціалізовані інструменти для IT-спеціалістів. Платформа включає функціонал для оцінки технічних навичок, створення портфоліо проектів та відстеження трендів у галузі. Сервіс також надає можливість анонімного пошуку роботи та захисту персональних даних кандидатів.

ZipRecruiter [9] використовує технології машинного навчання для персоналізації пошуку роботи. Платформа автоматично розсилає резюме кандидатів потенційним роботодавцям та надає рекомендації щодо релевантних вакансій. Сервіс також пропонує інструменти для відстеження статусу заявок та комунікації з роботодавцями.

Робота.ua [10] пропонує комплексне рішення для українського ринку праці, включаючи функціонал для створення відеорезюме, проведення онлайн-тестувань та організації віртуальних співбесід. Платформа надає

можливість публікації статей про кар’єрний розвиток та експертних матеріалів щодо тенденцій ринку праці [11].

1.2 Огляд технологій розробки веб-додатків

Сучасний процес розробки веб-додатків характеризується широким спектром доступних технологій та інструментів. JavaScript залишається основною мовою клієнтської частини веб-додатків, пропонуючи розробникам потужні можливості для створення інтерактивних інтерфейсів. Фреймворк React, створений компанією Facebook, надає компонентний підхід до розробки користувацьких інтерфейсів. React [12] використовує віртуальний DOM для оптимізації продуктивності веб-додатків. Бібліотека Redux забезпечує централізоване управління станом додатку. Екосистема React містить безліч додаткових інструментів та бібліотек для розширення функціональності.

Vue.js [13] пропонує прогресивний підхід до розробки веб-інтерфейсів, дозволяючи поступово впроваджувати фреймворк у проект. Фреймворк містить вбудовану систему реактивності для автоматичного оновлення DOM при зміні даних. Vue.js надає зручний синтаксис для створення шаблонів та компонентів. Інструмент Vue CLI спрощує процес налаштування проекту та збірки додатку. Vuex забезпечує управління станом для складних додатків. Vue Router дозволяє створювати маршрутизацію на стороні клієнта. Фреймворк підтримує серверний рендеринг для покращення SEO.

Angular [14] представляє повноцінний фреймворк для розробки веб-додатків, використовуючи TypeScript як основну мову програмування. Фреймворк включає потужну систему залежностей для модульної розробки. Angular CLI автоматизує створення компонентів, сервісів та інших елементів додатку. RxJS забезпечує реактивне програмування та обробку асинхронних операцій. Angular Material надає готові компоненти користувацького інтерфейсу. Фреймворк підтримує двостороннє зв’язування даних. Система модулів дозволяє організувати код у логічні блоки.

Node.js [15] забезпечує серверну розробку на JavaScript, дозволяючи створювати масштабовані веб-додатки. Express.js надає мінімалістичний та гнучкий фреймворк для створення веб-серверів. Система модулів NPM спрощує управління залежностями проекту. Node.js підтримує асинхронне програмування через callback-функції та Promise. Фреймворк дозволяє створювати RESTful API та WebSocket сервери. Модуль Cluster забезпечує масштабування додатків на багатоядерних системах. Node.js інтегрується з різними базами даних та сервісами.

ASP.NET Core [16] представляє крос-платформний фреймворк для створення сучасних веб-додатків на мові C#. Фреймворк підтримує модульну архітектуру та впровадження залежностей. Entity Framework Core забезпечує об’єктно-реляційне відображення для роботи з базами даних. SignalR дозволяє реалізувати двосторонню комунікацію в реальному часі. ASP.NET Core підтримує контейнеризацію через Docker. Фреймворк включає вбудовану систему автентифікації та авторизації. Middleware компоненти дозволяють налаштовувати конвеєр обробки запитів.

Порівняльний аналіз основних технологій розробки веб-додатків подано у таблиці 1.1, де відображено їхні переваги, недоліки та ключові особливості.

Таблиця 1.1 – Порівняння технологій розробки веб-додатків

Технологія	Переваги	Недоліки	Особливості
React	– віртуальний DOM; – екосистема; – компонентний підхід;	– висока складність; – потреба в додаткових бібліотеках;	– JSX синтаксис; – одностороння передача даних;
Angular	– повноцінний фреймворк; – TypeScript; – двостороннє зв’язування;	– висока крива навчання; – розмір;	– Dependency Injection; – CLI інструменти;
Vue.js	– простота вивчення; – гнучкість; – реактивність;	– менша екосистема; – обмежена підтримка TypeScript;	– прогресивний фреймворк; – шаблонний синтаксис;
Node.js	– асинхронність; – NPM екосистема; – єдина мова;	– однопотоковість; – Callback hell;	– Event-driven архітектура; – V8 engine;
ASP.NET Core	– крос-платформність; – висока продуктивність; – Entity Framework;	– потребує більше ресурсів; – складна конфігурація;	– Middleware; –Dependency Injection;

Django забезпечує високорівневий Python фреймворк для швидкої розробки веб-додатків. ORM Django спрощує взаємодію з базами даних через Python-класи. Адміністративний інтерфейс автоматично генерується на основі моделей даних. Система шаблонів підтримує успадкування та включення блоків. Django Rest Framework дозволяє створювати API-інтерфейси. Фреймворк містить вбудовані механізми захисту від поширених веб-атак. Система кешування підвищує продуктивність додатків.

Laravel надає елегантний синтаксис та потужні інструменти для PHP-розробки. Eloquent ORM забезпечує простий доступ до бази даних через об’єктну модель. Blade шаблонізатор спрощує створення динамічних веб-сторінок. Artisan CLI автоматизує рутинні завдання розробки. Laravel Mix спрощує збірку frontend-ресурсів. Фреймворк підтримує черги завдань для асинхронної обробки. Laravel включає вбудовану систему тестування.

Spring Framework представляє комплексне рішення для розробки Java-додатків. Spring Boot спрощує конфігурацію та розгортання додатків. Spring Security забезпечує захист веб-додатків. Spring Data спрощує доступ до різних типів баз даних. Spring MVC підтримує розробку веб-додатків за патерном MVC. Spring Cloud спрощує створення мікросервісної архітектури. Фреймворк підтримує аспектно-орієнтоване програмування.

GraphQL представляє альтернативний підхід до побудови API-інтерфейсів. Apollo Client спрощує інтеграцію GraphQL у клієнтські додатки. Schema Definition Language дозволяє описувати структуру даних. GraphQL підтримує отримання пов’язаних даних одним запитом. Система типів забезпечує валідацію даних. Інструменти розробника спрощують налагодження запитів. GraphQL дозволяє клієнтам визначати структуру відповіді.

Progressive Web Apps поєднують найкращі практики веб-розробки та нативних додатків. Service Workers забезпечують офлайн-функціональність та кешування ресурсів. Web App Manifest визначає метадані для встановлення додатку. Push API дозволяє реалізувати push-сповіщення. Background Sync API

забезпечує синхронізацію даних. PWA підтримують встановлення на домашній екран. Технологія забезпечує швидке завантаження додатків [17].

1.3 Обґрунтування вибору технології Blazor

Вибір Blazor для розробки сервісу пошуку роботи базується на низці технічних та практичних міркувань. Фреймворк дозволяє створювати інтерактивні веб-додатки використовуючи C# як на стороні сервера, так і на стороні клієнта. Blazor забезпечує високу продуктивність завдяки компіляції коду в WebAssembly. Технологія підтримує двосторонню комунікацію між клієнтом та сервером через SignalR. Єдина мова програмування спрощує розробку та підтримку проекту. Фреймворк інтегрується з екосистемою .NET, надаючи доступ до багатої колекції бібліотек та інструментів. Blazor підтримує компонентний підхід до розробки інтерфейсів.

Blazor Server забезпечує швидкий старт розробки без необхідності завантаження великих клієнтських бібліотек. Модель виконання на сервері гарантує безпеку бізнес-логіки та конфіденційних даних. SignalR забезпечує ефективну передачу даних між сервером та клієнтом. Серверна модель спрощує інтеграцію з базами даних та зовнішніми сервісами. Blazor Server автоматично синхронізує стан компонентів між клієнтом та сервером. Фреймворк підтримує авторизацію та автентифікацію на рівні компонентів. Модель дозволяє швидко розгортати додатки на production-середовищі.

Blazor WebAssembly надає можливість виконання C# коду безпосередньо в браузері. Технологія забезпечує швидкий відгук інтерфейсу та зменшує навантаження на сервер. WebAssembly дозволяє створювати прогресивні веб-додатки з офлайн-функціональністю. Клієнтська модель підтримує кешування даних та локальне зберігання стану. Blazor WebAssembly інтегрується з браузерними API та JavaScript бібліотеками. Фреймворк оптимізує розмір завантаження через lazy loading модулів. Технологія забезпечує хорошу продуктивність веб-додатків.

Система компонентів Blazor спрощує повторне використання коду та підтримку проекту. Компоненти підтримують параметри та події для гнучкої конфігурації поведінки. Каскадні параметри дозволяють передавати дані через ієрархію компонентів. Життєвий цикл компонентів надає точки для оптимізації та налаштування. Blazor підтримує вкладені компоненти та динамічне рендеринг. Розділення логіки та представлення покращує тестування компонентів. Система шаблонів спрощує створення інтерфейсів.

Entity Framework Core забезпечує зручну роботу з базами даних через об’єктну модель. ORM автоматично генерує схему бази даних на основі C# класів. Система міграцій спрощує версіонування структури бази даних. LINQ забезпечує типобезпечні запити до бази даних. Entity Framework підтримує різні провайдери баз даних. Фреймворк оптимізує продуктивність через кешування та відкладене завантаження. Система відстеження змін спрощує оновлення даних.

Dependency Injection в Blazor забезпечує гнучку архітектуру та тестованість коду. Сервіси можуть бути зареєстровані з різними життєвими циклами. DI контейнер автоматично вирішує залежності компонентів та сервісів. Інверсія управління спрощує модульне тестування. Blazor підтримує заміну реалізацій сервісів для різних середовищ. Система дозволяє створювати масштабовані додатки. Фреймворк забезпечує ефективне управління ресурсами.

Інтеграція з ASP.NET Core Identity спрощує реалізацію автентифікації та авторизації. Система ролей забезпечує гнучке управління доступом до функціоналу. OAuth підтримка дозволяє інтегрувати зовнішні провайдери автентифікації. JWT токени забезпечують безпечну передачу даних автентифікації. Identity підтримує користувацькі поля та валідацію даних. Система подій дозволяє розширювати функціонал автентифікації. Фреймворк захищає від поширених атак.

Blazor надає вбудовані засоби для валідації форм та користувацького вводу. Система валідації підтримує анотації даних та користувацькі валідатори. Валідація виконується як на клієнті, так і на сервері. Повідомлення про помилки можуть бути локалізовані. Blazor підтримує асинхронну валідацію даних.

Система форм забезпечує двостороннє зв’язування даних. Фреймворк надає компоненти для відображення помилок.

Локалізація в Blazor дозволяє створювати багатомовні інтерфейси. Ресурси локалізації можуть бути організовані в файли або бази даних. Система підтримує форматування дат, чисел та валют. Локалізація працює як на сервері, так і на клієнті. Blazor автоматично визначає мову користувача. Фреймворк підтримує переключення мов без перезавантаження сторінки. Система локалізації інтегрується з валідацією форм.

Розгортання Blazor додатків підтримується на різних платформах та хостингах. Docker контейнеризація спрощує розгортання та масштабування. Azure App Service забезпечує інтеграцію з хмарними сервісами. Blazor підтримує статичний хостинг для WebAssembly додатків. IIS надає розширені можливості конфігурації. Фреймворк підтримує розгортання через CI/CD pipeline. Система забезпечує моніторинг та діагностику [18].

1.4 Постановка завдання розробки

Створення сервісу пошуку роботи на базі фреймворку Blazor передбачає реалізацію комплексної системи взаємодії між роботодавцями та кандидатами. Проект розподіляється на серверну та клієнтську частини з використанням Blazor Server та Blazor WebAssembly. Архітектура системи будується за принципами чистого коду та розділення відповідальності. База даних проектується з урахуванням масштабованості та продуктивності. Система автентифікації забезпечує розмежування прав доступу між різними типами користувачів. Інтерфейс розробляється з фокусом на зручність використання та адаптивність. Проект включає модульне та інтеграційне тестування [16].

Функціонал реєстрації користувачів має забезпечувати створення двох типів облікових записів: для роботодавців та кандидатів. Система повинна перевіряти унікальність електронної пошти та валідність введених даних. Процес реєстрації включає підтвердження електронної пошти через відправку

посилання активації. Користувачі мають можливість відновлення паролю через електронну пошту. Профілі користувачів захищаються від несанкціонованого доступу. Система зберігає історію входів та активності користувачів. Реєстрація підтримує інтеграцію з соціальними мережами.

Профілі кандидатів містять розширену інформацію про освіту, досвід роботи та професійні навички. Система підтримує завантаження резюме в різних форматах та їх конвертацію. Кандидати можуть додавати посилання на портфоліо та професійні сертифікати. Профіль включає секцію рекомендацій та відгуків від попередніх роботодавців. Система автоматично пропонує релевантні вакансії на основі профілю кандидата. Користувачі можуть налаштовувати видимість різних секцій профілю. Профіль підтримує додавання фотографії та контактної інформації.

Роботодавці отримують інструменти для створення та управління вакансіями компанії. Система дозволяє встановлювати різні параметри вакансій: тип зайнятості, рівень заробітної плати, вимоги до кандидатів. Вакансії можуть включати детальний опис обов’язків та умов роботи. Роботодавці мають доступ до статистики переглядів та відгуків на вакансії. Система підтримує планування дат публікації та закриття вакансій. Вакансії можуть бути опубліковані в різних категоріях та регіонах. Роботодавці отримують сповіщення про нові відгуки.

Пошукова система забезпечує швидкий доступ до релевантних вакансій за різними критеріями. Фільтри дозволяють уточнювати пошук за спеціалізацією, локацією, рівнем досвіду та зарплатою. Результати пошуку оновлюються в реальному часі при зміні параметрів фільтрації. Система зберігає історію пошуків та дозволяє налаштовувати сповіщення. Користувачі можуть зберігати вакансії в обране для подальшого перегляду. Пошук підтримує сортування результатів за різними параметрами. Система враховує релевантність вакансій профілю користувача.

Система відгуків на вакансії забезпечує структурований процес комунікації між кандидатами та роботодавцями. Кандидати можуть прикріплювати супровідні листи та додаткові документи до відгуку. Роботодавці

отримують можливість встановлювати статуси розгляду відгуків. Система відстежує всі етапи розгляду кандидатури. Користувачі отримують сповіщення про зміну статусу відгуку. Відгуки зберігаються в історії активності користувача. Система забезпечує конфіденційність комунікації.

Месенджер для спілкування між користувачами реалізується з використанням SignalR для миттєвої доставки повідомлень. Чати підтримують обмін текстовими повідомленнями, файлами та посиланнями. Користувачі можуть створювати групові обговорення вакансій. Система зберігає історію листування та забезпечує пошук по повідомленнях. Месенджер підтримує статуси прочитання та доставки повідомлень. Користувачі можуть блокувати небажані контакти. Система забезпечує шифрування повідомлень.

Адміністративна панель надає інструменти для управління контентом та користувачами системи. Адміністратори можуть модерувати вакансії та профілі користувачів. Система надає доступ до статистики використання сервісу. Адміністратори можуть керувати категоріями та параметрами вакансій. Панель включає інструменти для розсилки системних повідомлень. Адміністратори мають доступ до логів системи. Панель забезпечує управління правами доступу.

Система аналітики збирає та аналізує дані про активність користувачів та ефективність вакансій. Роботодавці отримують доступ до метрик по своїх вакансіях. Система генерує звіти про тренди на ринку праці. Аналітика включає теплові карти активності користувачів. Система відстежує конверсію відгуків у працевлаштування. Роботодавці можуть порівнювати ефективність різних вакансій. Аналітичні дані використовуються для покращення рекомендацій.

Система сповіщень забезпечує інформування користувачів про релевантні події. Сповіщення можуть надсилатися через електронну пошту, браузер та мобільні пристрої. Користувачі можуть налаштовувати типи сповіщень та їх частоту. Система групує сповіщення за категоріями та пріоритетами. Сповіщення включають посилання на відповідні розділи системи. Користувачі можуть відмічати сповіщення як прочитані. Система зберігає історію сповіщень.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СЕРВІСУ ПОШУКУ РОБОТИ

2.1 Архітектура програмного забезпечення

Архітектура сервісу пошуку роботи базується на принципах чистої архітектури та мікросервісного підходу. Система розділяється на незалежні модулі, кожен з яких відповідає за конкретний функціонал. Взаємодія між модулями здійснюється через чітко визначені інтерфейси. Blazor Server забезпечує серверний рендеринг та обробку бізнес-логіки. База даних використовує Entity Framework Core для об’єктно-реляційного відображення. SignalR забезпечує двосторонню комунікацію в реальному часі. Система підтримує горизонтальне масштабування для забезпечення високої доступності.

Presentation Layer реалізується з використанням компонентного підходу Blazor. Компоненти інкапсулюють логіку відображення та обробки користувацьких дій. Модульна структура спрощує повторне використання компонентів. Стилізація здійснюється через CSS та SCSS препроцесор. JavaScript інтероперабельність забезпечує інтеграцію з браузерними API. Компоненти підтримують двостороннє зв’язування даних. Система маршрутизації забезпечує навігацію між сторінками.

Application Layer містить бізнес-логіку та координує взаємодію між різними частинами системи. Сервіси реалізують конкретні бізнес-операції та правила. Mediator pattern забезпечує слабке зв’язування між компонентами. Validation pipeline перевіряє коректність вхідних даних. Exception handling централізує обробку помилок. Command Query Responsibility Segregation розділяє операції читання та запису. Система логування фіксує критичні операції.

Domain Layer визначає основні сутності та бізнес-правила системи. Моделі даних реалізують доменну логіку та валідацію. Агрегати забезпечують цілісність пов’язаних сутностей. Value Objects інкапсулюють складні значення та правила.

Domain Events дозволяють реагувати на зміни стану. Специфікації визначають правила вибірки даних. Repository interfaces абстрагують доступ до даних.

Infrastructure Layer забезпечує технічну реалізацію взаємодії з зовнішніми системами. Репозиторії реалізують доступ до бази даних через Entity Framework. Email service відповідає за відправку повідомлень. File storage обробляє завантаження та зберігання файлів. Caching service оптимізує продуктивність запитів. Authentication provider забезпечує автентифікацію користувачів. API clients взаємодіють з зовнішніми сервісами.

Data Access Layer абстрагує роботу з базою даних через паттерн Repository. Unit of Work забезпечує атомарність транзакцій. Query specifications інкапсулюють логіку вибірки даних. Lazy loading оптимізує завантаження пов’язаних даних. Migration system керує версіями схеми бази даних. Connection pooling оптимізує використання підключень. Query caching покращує продуктивність читання.

Real-time Communication Module забезпечує миттєву взаємодію між користувачами. SignalR hubs координують обмін повідомленнями. Message queues забезпечують надійну доставку даних. Presence tracking відстежує онлайн-статус користувачів. Connection management обробляє підключення клієнтів. Message persistence зберігає історію повідомлень. Scaling support забезпечує масштабування WebSocket з’єднань.

Search Module реалізує функціонал пошуку та фільтрації даних. Elasticsearch забезпечує повнотекстовий пошук. Faceted search підтримує фасетну навігацію. Search suggestions надають підказки при введенні. Query boosting оптимізує релевантність результатів. Geo-search підтримує пошук за місцезнаходженням. Indexing service оновлює пошуковий індекс.

2.2 Проектування бази даних

База даних сервісу пошуку роботи проектується з використанням Microsoft SQL Server та Entity Framework Core. Структура бази даних забезпечує

зберігання всієї необхідної інформації про користувачів, вакансії та взаємодії між ними. Таблиці пов’язуються через зовнішні ключі з підтримкою цілісності даних. Індекси оптимізують швидкість виконання запитів. Збережені процедури забезпечують виконання складних операцій. Система тригерів підтримує актуальність даних. Схема бази даних підтримує версіонування через міграції.

Опис основних таблиць бази даних, їх функціонального призначення, ключових полів та міжтабличних зв’язків наведено в таблиці 2.1.

Таблиця 2.1 – Структура бази даних сервісу пошуку роботи

Таблиця	Призначення	Основні поля	Зв’язки
Users	інформація про користувачів	– UserID; – Email; – PasswordHash; – RoleID;	– Roles; – UserProfiles;
Vacancies	вакансії роботодавців	– VacancyID; – Title; – Description; – CompanyID;	– Companies; – Applications;
Applications	відгуки на вакансії	– ApplicationID; – VacancyID; – UserID; – Status;	– Users; – Vacancies;
Companies	профілі компаній	– CompanyID; – Name; – Description; – LogoURL;	– Users; – Vacancies;
Skills	навички кандидатів	– SkillID; – Name; – Category;	– UserSkills
UserSkills	зв’язок користувачів з навичками	– UserID; – SkillID; – Level;	– Users; – Skills;

Таблиця Users зберігає основну інформацію про користувачів системи. Поля таблиці включають базові дані: email, хеш паролю, ім’я, контактну інформацію. Зовнішній ключ RoleId пов’язує користувача з його роллю в системі. Статус користувача відстежується через поле IsActive. Поле CreatedAt фіксує дату реєстрації. Додаткові налаштування зберігаються в JSON форматі. Таблиця індексується за email та номером телефону.

CompanyProfiles зберігає розширену інформацію про компанії-роботодавці. Структура включає поля для опису компанії, історії, місії та цінностей. Логотип та медіа-файли зберігаються як посилання на файлове сховище. Контактна інформація включає адресу, телефони та соціальні мережі. Поле VerificationStatus відстежує перевірку компанії. Рейтинг компанії розраховується автоматично. Географічні координати офісів зберігаються для пошуку.

Таблиця Vacancies містить інформацію про відкриті позиції. Поля описують вимоги, обов'язки, умови роботи та компенсацію. Зовнішній ключ CompanyId пов'язує вакансію з компанією. Статус вакансії контролюється через поле Status. Дати публікації та закриття визначають життєвий цикл вакансії. Категорії та теги зберігаються в пов'язаних таблицях. Пріоритет вакансії впливає на порядок відображення. Перегляди та відгуки підраховуються через тригери.

Applications відстежує відгуки кандидатів на вакансії. Таблиця зберігає зв'язки між користувачами та вакансіями. Статус розгляду заявки контролюється через поле Status. Супровідні листи та додаткові документи зберігаються як посилання. Історія зміни статусів фіксується в окремій таблиці. Система сповіщень тригериться при зміні статусу. Комунікація між сторонами зберігається в повідомленнях. Оцінки кандидатів записуються в окремих полях [17].

2.3 Проектування користувацького інтерфейсу

Інтерфейс сервісу пошуку роботи розроблений з урахуванням сучасних принципів UX/UI дизайну. Система використовує компонентний підхід Blazor для створення інтерактивних елементів. Дизайн адаптується під різні розміри екранів через CSS Grid та Flexbox. Кольорова схема забезпечує контрастність та читабельність контенту. Типографіка базується на системних шрифтах для оптимальної продуктивності. Анімації додають плавності переходам між станами інтерфейсу. Компоненти підтримують теми оформлення та локалізацію.

Головна сторінка представляє пошукову систему з фільтрами та списком вакансій. Пошуковий рядок підтримує автодоповнення та підказки. Фільтри групуються за категоріями для зручної навігації. Список вакансій оновлюється динамічно при зміні параметрів пошуку. Картки вакансій відображають ключову інформацію про позиції. Пагінація забезпечує швидку навігацію по результатах. Сортування дозволяє упорядковувати вакансії за різними критеріями.

Сторінка реєстрації пропонує вибір типу облікового запису та форму введення даних. Валідація полів виконується в реальному часі з підказками. Паролі перевіряються на складність та надійність. Реєстрація через соціальні мережі спрощує процес створення акаунту. Підтвердження email реалізовано через відправку коду. Форма підтримує автозаповнення браузера. Прогрес-бар показує етапи реєстрації.

Профіль кандидата організований у розділи з різними типами інформації. Редагування даних відбувається через модальні вікна. Завантаження фото профілю включає попередній перегляд. Резюме можна створювати через вбудований редактор. Навички додаються через систему тегів з автодоповненням. Портфоліо підтримує різні формати медіа-файлів. Налаштування приватності контролюють видимість даних.

Сторінка компанії презентує інформацію про роботодавця та його вакансії. Галерея зображень демонструє офіс та команду. Опис компанії форматується через rich-text редактор. Карта показує розташування офісів компанії. Відгуки співробітників відображаються з рейтингами. Стрічка новин компанії оновлюється автоматично. Контактна інформація групується за типами зв’язку.

Створення вакансії реалізовано через покроковий процес заповнення даних. Текстовий редактор підтримує форматування та списки. Вибір навичок здійснюється через пошук з підказками. Зарплатна вилка встановлюється через подвійний слайдер. Попередній перегляд показує фінальний вигляд вакансії. Публікація може бути відкладена на певну дату. Шаблони вакансій прискорюють створення схожих позицій.

Система повідомлень забезпечує комунікацію між користувачами через чат. Повідомлення оновлюються в реальному часі через SignalR. Історія листування зберігається з можливістю пошуку. Файли можна прикріплювати через drag-and-drop. Емодзі та форматування тексту підтримуються редактором. Статуси прочитання відображаються індикаторами. Сповіщення про нові повідомлення показуються в браузері.

Панель адміністратора надає інструменти управління контентом та користувачами. Таблиці даних підтримують фільтрацію та експорт. Графіки показують статистику використання системи. Модерація контенту здійснюється через систему черг. Масові операції виконуються через чекбокси. Логи системи відображаються з можливістю пошуку. Налаштування сервісу доступні через форми конфігурації.

Мобільна версія оптимізована для зручного використання на смартфонах. Навігація здійснюється через нижнє меню. Жести підтримуються для основних операцій. Форми адаптовані під сенсорне введення. Зображення оптимізуються під розмір екрану. Push-сповіщення працюють через service worker. Офлайн-режим кешує основний контент.

Сторінка пошуку кандидатів надає роботодавцям інструменти підбору персоналу. Фільтри дозволяють уточнювати параметри пошуку спеціалістів. Картки кандидатів показують ключові навички та досвід. Порівняння кандидатів реалізовано через таблицю характеристик. Збереження пошуків дозволяє відстежувати нових кандидатів. Експорт результатів доступний у різних форматах. Система рекомендацій пропонує релевантних спеціалістів.

Аналітична панель візуалізує дані про ефективність вакансій та активність користувачів. Графіки будуються через бібліотеку Chart.js. Метрики групуються за часовими періодами. Експорт звітів доступний у PDF та Excel. Дашборди налаштовуються під потреби користувача. Сповіщення надсилаються при досягненні цільових показників. Прогнози будуються на основі історичних даних.

Система сповіщень інформує користувачів через різні канали комунікації. Спливаючі повідомлення показують термінові сповіщення. Email-розсилки формуються за шаблонами. Налаштування дозволяють обрати типи сповіщень. Групування повідомлень зменшує інформаційний шум. Історія сповіщень доступна для перегляду. Сповіщення позначаються як прочитані автоматично.

Сторінка налаштувань дозволяє персоналізувати роботу з сервісом. Параметри згруповані за категоріями функціоналу. Зміни зберігаються автоматично після модифікації. Експорт та імпорт налаштувань доступні через JSON. Скидання до значень за замовчуванням можливе для окремих груп. Попередній перегляд показує вплив змін. Підказки пояснюють призначення параметрів.

Форма зворотного зв’язку забезпечує комунікацію з підтримкою сервісу. Вибір теми звернення структурує запити користувачів. Файли можна прикріплювати для ілюстрації проблем. Статус обробки звернення відображається в особистому кабінеті. Історія звернень зберігається для довідки. Автоматичні відповіді надсилаються для типових питань. Оцінка якості підтримки доступна після вирішення запиту.

Сторінка помилок надає інформацію про проблеми та шляхи їх вирішення. Різні типи помилок мають унікальний дизайн сторінок. Кнопки швидкої навігації повертають до робочого стану. Технічна інформація доступна для служби підтримки. Автоматичне перенаправлення працює після відновлення сервісу. Форма звітування про помилки вбудована на сторінку. Статистика помилок збирається для аналізу.

2.4 Розробка діаграм UML

Проектування системи пошуку роботи починається з побудови діаграми варіантів використання (Use Case). Діаграма відображає взаємодію між користувачами та системою. Основні актори включають кандидатів, роботодавців та адміністраторів. Варіанти використання групуються за

функціональними модулями системи. Зв’язки між варіантами показують залежності та розширення. Передумови та постумови визначають контекст виконання операцій. Потоки подій описують сценарії взаємодії.

Структурне представлення основних компонентів системи та їх зв’язків реалізовано у діаграмі класів, що наведена на рисунку 2.1.

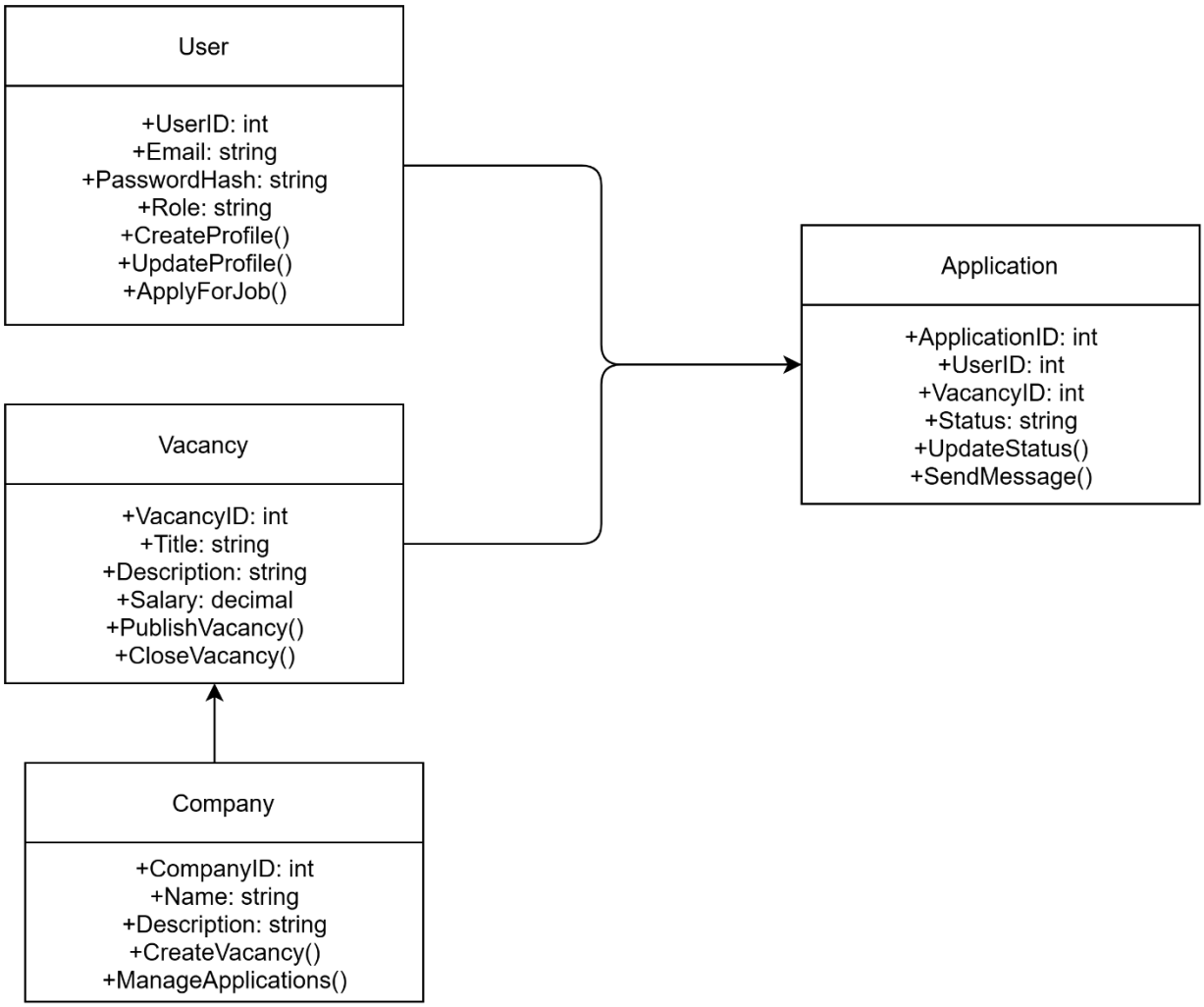

Рисунок 2.1 – Діаграма класів

Діаграма класів представляє структурну модель системи через об’єктно-орієнтований підхід. Класи розділяються на доменні сутності, сервіси та контролери. Атрибути класів визначають структуру даних. Методи описують поведінку об’єктів. Зв’язки між класами показують відношення успадкування та

агрегації. Інтерфейси визначають контракти взаємодії. Модифікатори доступу контролюють видимість елементів.

Як об’єкти взаємодіють під час виконання операцій, показано на рисунку 2.2.

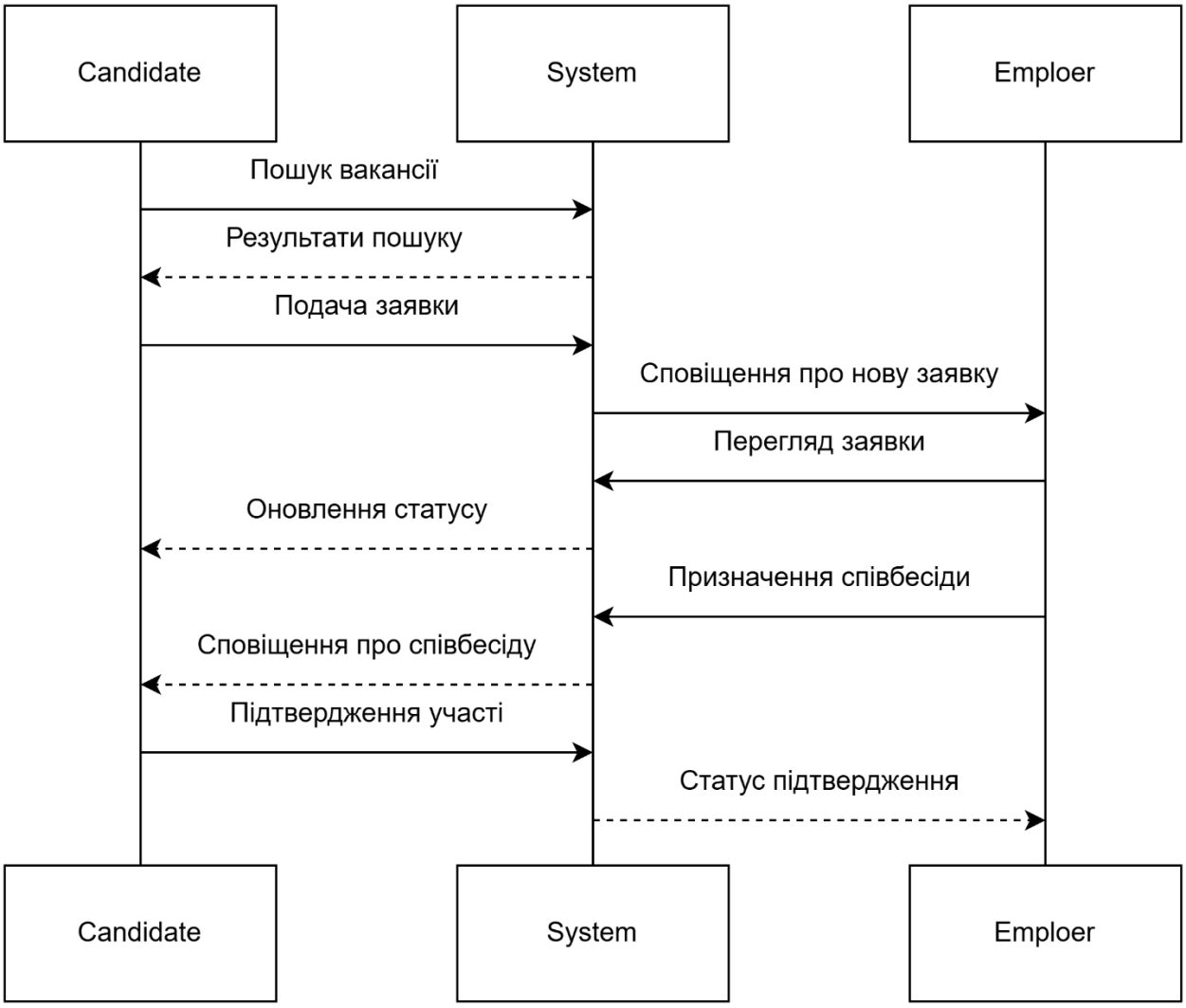

Рисунок 2.2 – Діаграма послідовності

Послідовність взаємодії об’єктів відображається через sequence діаграми. Часова шкала показує порядок виклику операцій. Об’єкти розміщуються вертикально з лініями життя. Повідомлення між об’єктами позначають методи взаємодії. Умовні блоки показують розгалуження логіки. Цикли відображають повторювані операції. Паралельні потоки демонструють асинхронність.

Sequence діаграми також допомагають виявити потенційні помилки у логіці виконання.

Життєвий цикл об’єктів у системі детально представлений на рисунку 2.3. Діаграма показує, як об’єкти змінюють свій стан залежно від подій, що відбуваються в системі.

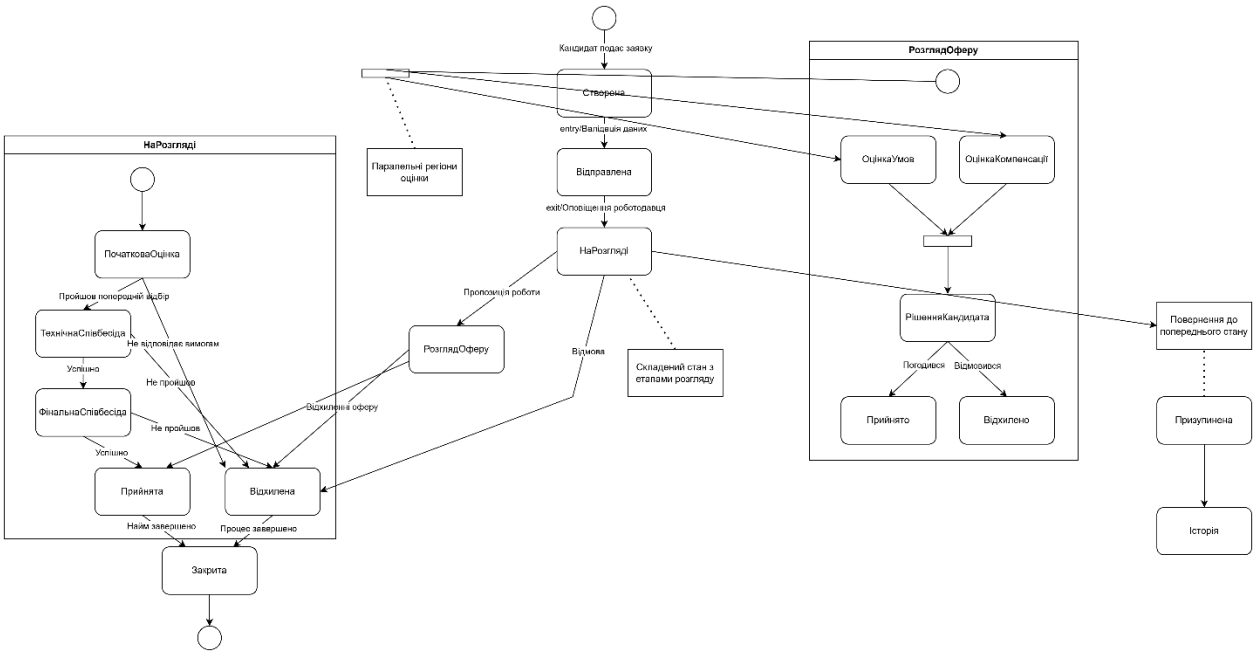

Рисунок 2.3 – Діаграма стану описує життєвий цикл об’єктів системи

Стани представляють можливі конфігурації об’єктів. Переходи показують зміни стану через події. Дії виконуються при вході та виході зі станів. Складені стани групують пов’язані конфігурації. Історичні стани зберігають попередні конфігурації. Паралельні регіони показують незалежні аспекти стану.

Профілі UML розширюють нотацію для конкретних доменів. Стереотипи визначають спеціалізовані типи елементів. Позначені значення додають специфічні атрибути. Обмеження профілю встановлюють правила використання. Піктограми надають візуальні позначення. Теги метаданих додають семантичну інформацію. Шаблони профілю визначають типові структури.

2.5 Проектування системи безпеки та авторизації

Система безпеки сервісу пошуку роботи базується на багаторівневій архітектурі захисту. ASP.NET Core Identity забезпечує базовий функціонал автентифікації та авторизації. JWT токени використовуються для безпечної передачі даних між клієнтом та сервером. Паролі зберігаються в базі даних у захешованому вигляді з використанням сучасних алгоритмів. Система ролей розділяє користувачів на кандидатів, роботодавців та адміністраторів. Кожна роль має чітко визначений набір прав та дозволів. Механізм claims забезпечує гнучке управління доступом.

Процес реєстрації користувачів включає декілька рівнів перевірки. Валідація email відбувається через відправку підтверджуючого посилання. Паролі перевіряються на відповідність політиці безпеки. Система запобігає створенню дублікатів облікових записів. CAPTCHA захищає від автоматизованої реєстрації. Двофакторна автентифікація доступна для додаткового захисту. IP-адреси та user-agent перевіряються на підозрілу активність.

OAuth 2.0 інтеграція дозволяє авторизацію через соціальні мережі. Підтримуються популярні провайдери: Google, Facebook, LinkedIn. Система зберігає токени доступу в захищеному форматі. Користувачі можуть керувати підключеними обліковими записами. Оновлення токенів відбувається автоматично. Профілі синхронізуються з даними соціальних мереж. Відключення зовнішніх провайдерів не впливає на локальний акаунт.

Сесії користувачів контролюються через систему управління токенами. Refresh токени забезпечують автоматичне продовження сесій. Система відстежує активні сесії користувача. Одночасні входи обмежуються налаштуваннями безпеки. Підозрілі сесії автоматично завершуються. Історія входів зберігається для аудиту. Користувачі можуть переглядати та завершувати активні сесії.

CORS політики обмежують доступ до API системи. Білий список дозволених доменів контролює крос-доменні запити. Preflight запити перевіряються на відповідність політикам. Заголовки безпеки встановлюються для всіх відповідей. OPTIONS запити фільтруються через middleware. Credentialed запити дозволяються вибірково. Політики кешування налаштовані для захисту даних.

Система антивірусного захисту сканує всі завантажені файли. Перевірка відбувається перед збереженням файлів у сховище. Підозрілі файли автоматично блокуються. Розміри та типи файлів обмежуються налаштуваннями. Сканування виконується асинхронно для продуктивності. Результати перевірок логуються для аналізу. Користувачі отримують сповіщення про заблоковані файли.

XSS захист реалізований через вбудовані механізми Blazor. Користувацький ввід екранується перед відображенням. Content Security Policy обмежує виконання скриптів. HTML санітайзер очищує небезпечний контент. Атрибути безпеки встановлюються для всіх форм. JavaScript код виконується в ізольованому середовищі. Фреймворк запобігає ін’єкціям шкідливого коду.

CSRF токени захищають форми від підробки запитів. Токени генеруються для кожної сесії користувача. Перевірка токенів відбувається на серверній стороні. Форми без валідних токенів відхиляються. Токени оновлюються при зміні сесії. Система протидіє повторному використанню токенів. Механізм працює прозоро для користувачів.

Логування безпеки відстежує всі критичні операції. Події безпеки зберігаються з детальним контекстом. Підозріла активність позначається для розслідування. Система агрегує логи з різних компонентів. Аналіз логів виявляє потенційні загрози. Сповіщення надсилаються при виявленні інцидентів. Логи архівуються для довготривалого зберігання.

Система резервного копіювання захищає дані користувачів. Бекапи створюються за розкладом з шифруванням. Різні типи даних мають окремі політики резервування. Відновлення даних тестується регулярно. Копії

зберігаються в географічно розподілених локаціях. Система відстежує цілісність резервних копій. Автоматичне очищення видаляє застарілі бекапи.

Шифрування даних застосовується на різних рівнях системи. Комунікації захищаються через TLS протокол. Конфіденційні дані шифруються перед збереженням. Ключі шифрування зберігаються окремо від даних. Система підтримує ротацію ключів шифрування. Алгоритми шифрування відповідають сучасним стандартам. Дешифрування виконується тільки при необхідності.

Управління доступом до API реалізовано через API ключі. Ключі генеруються з обмеженим терміном дії. Система відстежує використання ключів. Rate limiting запобігає зловживанню API. Доступ можна обмежувати за IP-адресами. Ключі можуть бути відкликані адміністратором. Різні рівні доступу контролюються через scope.

Моніторинг безпеки відстежує стан системи захисту. Метрики безпеки збираються в реальному часі. Аномалії детектуються через статистичний аналіз. Сповіщення налаштовані для різних типів загроз. Панель моніторингу показує поточний стан безпеки. Тренди аналізуються для прогнозування загроз. Звіти генеруються для аудиту безпеки.

Політики безпеки документують вимоги та процедури. Правила регулярно переглядаються та оновлюються. Користувачі інформуються про зміни в політиках. Навчальні матеріали доступні для різних ролей. Процедури реагування на інциденти чітко визначені. Відповідальність розподілена між командами. Аудит перевіряє дотри мання політик.

WAF захищає від типових веб-атак. Правила фільтрації блокують шкідливі запити. Сигнатури атак регулярно оновлюються. Легітимний трафік пропускається без затримок. Помилкові спрацювання аналізуються та коригуються. Географічна фільтрація обмежує доступ з підозрілих регіонів. Атаки логуються для подальшого аналізу.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка серверної частини

Серверна частина сервісу пошуку роботи реалізована на платформі ASP.NET Core 8.0. Архітектура побудована за принципами чистої архітектури з розділенням на шари. Domain Layer містить бізнес-моделі та основну логіку. Application Layer реалізує бізнес-правила та сценарії використання. Infrastructure Layer забезпечує взаємодію з зовнішніми системами. Presentation Layer реалізує API endpoints та контролери. WebAPI побудоване за принципами REST.

Доменні моделі описують основні сутності системи через C# класи. Entity Base Class забезпечує базову функціональність для всіх сутностей. Value Objects інкапсулюють складні значення та правила валідації. Агрегати забезпечують цілісність пов’язаних сутностей. Domain Events дозволяють реагувати на зміни стану. Repository interfaces визначають контракти доступу до даних. Domain Services реалізують складну бізнес-логіку.

Application сервіси реалізують основні сценарії використання системи. CQRS розділяє операції читання та запису даних. Mediator pattern забезпечує слабке зв’язування компонентів. Validation pipeline перевіряє вхідні дані запитів. Exception handling централізує обробку помилок. AutoMapper спрощує маппінг між різними моделями. Fluent Validation забезпечує гнучку валідацію.

Entity Framework Core реалізує доступ до бази даних Microsoft SQL Server. Code-First підхід автоматично генерує схему бази даних. Міграції забезпечують версіонування структури даних. Репозиторії реалізують шаблони доступу до даних. Unit Of Work забезпечує атомарність транзакцій. Lazy loading оптимізує завантаження зв’язаних даних. Query specifications інкапсулюють логіку вибірки.

Контролери API реалізують REST endpoints для клієнтських додатків. Swagger генерує автоматичну документацію API. Versioning забезпечує підтримку різних версій API. Rate limiting запобігає перевантаженню серверів.

Caching оптимізує часто запитувані дані. Response compression зменшує обсяг трафіку. Health checks відстежують стан сервісів.

SignalR забезпечує двосторонню комунікацію в реальному часі. Hub-класи реалізують методи обміну повідомленнями. Groups дозволяють групувати підключення користувачів. Connection management відстежує активні з’єднання. Message persistence зберігає історію повідомлень. Scaling підтримує роботу через Redis backplane. Authentication інтегрується з основною системою.

Background Service реалізує обробку фонових завдань. Hangfire керує виконанням відкладених задач. Job queues забезпечують надійну доставку завдань. Retry policies обробляють помилки виконання. Scheduling дозволяє планувати регулярні завдання. Progress tracking відстежує стан виконання. Resource management контролює навантаження.

Email сервіс забезпечує розсилку повідомлень користувачам. SendGrid використовується як SMTP провайдер. Templates керують шаблонами листів. Queue system забезпечує асинхронну відправку. Delivery tracking відстежує статус доставки. Bounce handling обробляє недоставлені листи. Analytics збирає статистику розсилок.

File Storage сервіс керує завантаженням та зберіганням файлів. Azure Blob Storage використовується як сховище. File validation перевіряє безпеку файлів. Image processing оптимізує зображення. CDN прискорює доставку статичного контенту. Cleanup видаляє тимчасові файли. Access control обмежує доступ до файлів.

Logging система збирає інформацію про роботу сервісу. Serilog використовується для структурованого логування. Elasticsearch зберігає логи для аналізу. Kibana візуалізує дані логів. Log levels контролюють деталізацію записів. Correlation ID відстежує пов’язані події. Retention policies керують зберіганням логів.

Caching система оптимізує продуктивність запитів. Redis використовується як розподілене сховище кешу. Cache invalidation підтримує актуальність даних. Memory cache прискорює часті запити. Distributed cache

синхронізує дані між серверами. Cache tags групують пов’язані елементи. Cache dependencies відстежують залежності даних.

Configuration система керує налаштуваннями сервісу. Azure Key Vault зберігає секретні дані. Environment variables визначають налаштування середовища. JSON файли містять базову конфігурацію. Reloading оновлює налаштування без перезапуску. Validation перевіряє коректність конфігурації. Hierarchical structure організує параметри.

Monitoring система відстежує стан та продуктивність сервісу. Application Insights збирає телеметрію. Metrics відстежують ключові показники. Alerts сповіщають про проблеми. Dashboards візуалізують стан системи. Tracing відстежує послідовність операцій. Performance counters вимірюють продуктивність.

Authentication система забезпечує безпеку доступу. JWT токени використовуються для автентифікації. OAuth підтримує зовнішніх провайдерів. Role-based access control керує правами. Claims-based authorization забезпечує гнучкі дозволи. Two-factor authentication підвищує безпеку. Session management контролює активні сесії.

API Gateway маршрутизує запити до внутрішніх сервісів. Load balancing розподіляє навантаження. Circuit breaker запобігає каскадним відмовам. Request aggregation оптимізує клієнтські запити. SSL termination централізує шифрування. Rate limiting захищає від перевантаження. Request tracking відстежує запити.

3.2 Реалізація клієнтської частини на Blazor

Клієнтська частина сервісу пошуку роботи реалізована з використанням Blazor WebAssembly. Компонентна архітектура забезпечує модульність та перевикористання коду. Стан додатку керується через вбудовані механізми Blazor. Маршрутизація реалізована через Router компонент. JavaScript інтероперабельність дозволяє використовувати браузерні API. LocalStorage

зберігає локальні дані користувача. PWA функціональність забезпечує офлайн-роботу.

Base компоненти реалізують загальну функціональність інтерфейсу. Layout компоненти визначають структуру сторінок. Shared компоненти використовуються в різних частинах додатку. Parameters дозволяють конфігурувати компоненти. EventCallback забезпечує комунікацію між компонентами. RenderFragment дозволяє передавати шаблони розмітки. Lifecycle hooks керують життєвим циклом.

Форми реалізовані через EditForm компонент Blazor. DataAnnotations забезпечують валідацію полів. ValidationMessage показує помилки валідації. InputBase розширює стандартні елементи вводу. Custom Validation реалізує складні правила перевірки. Model Binding зв’язує дані з моделями. Form Submission обробляє відправку даних.

State Management реалізований через сервіси з DI контейнера. Observable Pattern відстежує зміни стану. Cascading Parameters передають дані через ієрархію. Local Storage синхронізує стан між сесіями. State Container централізує управління даними. Redux Pattern реалізує передбачуваний потік даних. Event Aggregator забезпечує комунікацію компонентів.

HTTP клієнт реалізує взаємодію з серверним API. HttpClient Factory керує життєвим циклом клієнтів. Retry Policy обробляє тимчасові помилки. Authentication Handler додає токени до запитів. Request Caching кешує відповіді сервера. Response Interceptor обробляє помилки API. Request Cancellation скасовує неактуальні запити.

SignalR клієнт забезпечує реалтайм комунікацію. HubConnection керує підключенням до сервера. Reconnection відновлює з’єднання при розривах. Message Handlers обробляють вхідні повідомлення. Connection State відстежує стан підключення. Stream підтримує потокову передачу даних. Groups керують груповими повідомленнями.

Authentication реалізований через AuthenticationStateProvider. JWT токени зберігаються в LocalStorage. Login Form обробляє введення облікових даних.

OAuth реалізує вхід через соціальні мережі. Logout очищає дані автентифікації. AuthorizeView керує відображенням контенту. Redirect обробляє неавторизований доступ.

Image Processing оптимізує роботу з зображеннями. File Upload обробляє завантаження файлів. Image Resizing змінює розміри зображень. Lazy Loading відкладає завантаження. Preview генерує попередній перегляд. Drag and Drop підтримує перетягування файлів. Format Conversion конвертує формати зображень.

Localization забезпечує багатомовність інтерфейсу. Resource Files зберігають переклади. Culture Provider визначає поточну локаль. String Formatting форматує дані локально. Number Formatting локалізує числа. Date Formatting адаптує формати дат. Currency Formatting локалізує валюти.

Error Handling централізує обробку помилок. Error Boundary перехоплює помилки компонентів. Custom Error Pages показують інформативні повідомлення. Logging записує помилки для аналізу. Exception Filter фільтрує типи помилок. Recovery відновлює роботу після збоїв. User Notification інформує про проблеми.

Performance Optimization покращує швидкість роботи. Code Splitting розділяє код на чанки. Lazy Loading відкладає завантаження модулів. Prerendering генерує початковий HTML. Caching кешує статичні ресурси. Bundle Optimization мінімізує розмір файлів. Memory Management оптимізує використання пам’яті.

Testing реалізований через bUnit фреймворк. Component Testing тестує окремі компоненти. Integration Testing перевіряє взаємодію компонентів. Mocking імітує зовнішні залежності. Snapshot Testing порівнює розмітку компонентів. Event Testing перевіряє обробку подій. DOM Testing тестує маніпуляції з DOM.

Accessibility забезпечує доступність інтерфейсу. ARIA атрибути додають семантику. Keyboard Navigation підтримує керування з клавіатури. Screen Reader оптимізує читання контенту. Color Contrast забезпечує контрастність. Focus

Management керує фокусом елементів. Semantic HTML використовує правильні теги.

Progressive Enhancement покращує функціональність. Feature Detection перевіряє підтримку функцій. Fallback забезпечує базову роботу. Offline Support підтримує роботу без мережі. Push Notifications відправляє сповіщення. Background Sync синхронізує дані. Install Prompt пропонує встановлення PWA.

SEO оптимізація покращує видимість у пошуку. Meta Tags генеруються динамічно. Title Management керує заголовками сторінок. Canonical URLs визначають основні URL. Sitemap Generation створює карту сайту. Schema Markup додає семантичну розмітку. Robots.txt конфігурує пошукові роботи.

3.3 Інтеграція системи авторизації та автентифікації

Автентифікація в сервісі пошуку роботи реалізована на базі ASP.NET Core Identity. JWT токени використовуються для безпечної передачі даних авторизації. Система підтримує локальну реєстрацію через email та пароль. OAuth 2.0 забезпечує вхід через соціальні мережі. Refresh токени дозволяють автоматично продовжувати сесії. Bearer схема автентифікації застосовується для API запитів. Двофакторна автентифікація доступна для підвищеного захисту.

IdentityServer4 конфігурація налаштовує параметри автентифікації. Клієнтські додатки реєструються з унікальними ідентифікаторами. Scopes визначають доступні ресурси та права. Налаштування токенів включають термін дії та алгоритми підпису. Провайдери зовнішньої автентифікації конфігуруються через секрети. Політики паролів встановлюють вимоги до складності. Конфігурація сесій визначає параметри управління.

Реєстрація користувачів проходить через валідацію даних. Email підтверджується через відправку посилання активації. Паролі хешуються з використанням сучасних алгоритмів. Система запобігає створенню дублікатів облікових записів. Профілі користувачів створюються з базовими

налаштуваннями. Ролі призначаються відповідно до типу користувача. Сповіщення інформують про успішну реєстрацію.

Процес входу перевіряє облікові дані користувачів. Невдалі спроби входу обмежуються для захисту від атак. Успішна автентифікація генерує JWT токен. Claims містять інформацію про користувача та його права. Сесія користувача відстежується через токени. IP-адреса та браузер перевіряються на підозрілу активність. Історія входів зберігається для аудиту.

OAuth інтеграція підтримує популярні провайдери автентифікації. Google, Facebook та LinkedIn налаштовані як зовнішні провайдери. Процес OAuth реалізований через стандартний потік авторизації. Токени соціальних мереж конвертуються у внутрішні токени. Профілі синхронізуються з даними соціальних мереж. Користувачі можуть керувати підключеними обліковими записами. Відключення провайдерів не впливає на локальний акаунт.

Role-based authorization контролює доступ до ресурсів. Ролі визначають базові набори прав користувачів. Policies реалізують складні правила авторизації. Requirements перевіряють додаткові умови доступу. Authorization handlers обробляють перевірку вимог. Resource-based authorization захищає конкретні об’єкти. Claims-based authorization забезпечує гнучкі дозволи.

Управління паролями включає функції відновлення та зміни. Відновлення паролю відбувається через email посилання. Зміна паролю вимагає підтвердження поточного паролю. Історія паролів запобігає повторному використанню. Складність паролів перевіряється за встановленими правилами. Тимчасові паролі генеруються для нових користувачів. Блокування облікового запису відбувається після підозрілих операцій.

Token management керує життєвим циклом токенів. Refresh токени дозволяють оновлювати access токени. Revocation endpoints відкликають недійсні токени. Token storage безпечно зберігає токени на клієнті. Token validation перевіряє підпис та термін дії. Token introspection надає інформацію про токени. Rotation policies керують оновленням токенів.

Session management відстежує активні сесії користувачів. Concurrent sessions обмежуються налаштуваннями безпеки. Session tokens зберігаються в захищених cookies. Session termination завершує неактивні сесії. Session hijacking prevention запобігає перехопленню. Session fixation protection захищає від атак. Користувачі можуть переглядати активні сесії.

Account lockout захищає від брутфорс атак. Failed attempts tracking відстежує невдалі спроби. Lockout duration встановлює період блокування. Notification emails інформують про блокування. Admin unlock дозволяє розблокувати акаунти. IP-based blocking запобігає атакам з однієї адреси. Security logs фіксують спроби злому.

External authentication events обробляють зовнішню автентифікацію. Success events створюють локальні облікові записи. Failure events логують помилки автентифікації. Token validation events перевіряють зовнішні токени. Profile sync events оновлюють дані користувача. Link events пов’язують облікові записи. Unlink events відключають провайдерів.

Anti-forgery protection захищає від CSRF атак. Anti-forgery tokens генеруються для форм. Token validation middleware перевіряє токени. Token persistence зберігає токени між запитами. Token renewal оновлює застарілі токени. Configuration options налаштовують захист. Error handling обробляє помилки валідації.

Security headers забезпечують додатковий захист. CSP headers обмежують джерела контенту. HSTS enforces використання HTTPS. X-Frame-Options запобігає clickjacking. X-Content-Type-Options запобігає MIME-sniffing. Referrer-Policy контролює referrer information. Feature-Policy обмежує браузерні функції.

Audit logging відстежує операції безпеки. Security events логуються з деталями. User actions фіксуються для аналізу. Suspicious activities позначаються для розслідування. Log storage зберігає історію подій. Log analysis виявляє патерни атак. Alerting сповіщає про інциденти.

Compliance monitoring забезпечує відповідність стандартам. GDPR requirements відстежуються системою. Data protection забезпечує захист даних. Privacy policy enforcement контролює використання даних. Consent management керує дозволами користувачів. Audit trails підтверджують відповідність. Regular assessments перевіряють безпеку.

3.4 Реалізація основних функціональних модулів

3.4.1 Модуль реєстрації та авторизації

Реєстраційний модуль забезпечує процес створення нових облікових записів користувачів (рис. 3.1). Форма реєстрації реалізована через компонент RegistrationForm з використанням Blazor. Валідація полів виконується в реальному часі з використанням DataAnnotations. Система перевіряє унікальність email адреси перед створенням акаунту. Паролі хешуються з використанням сучасних криптографічних алгоритмів. Автоматична генерація userID забезпечує унікальність користувачів. Сповіщення про успішну реєстрацію надсилається на вказану електронну пошту.

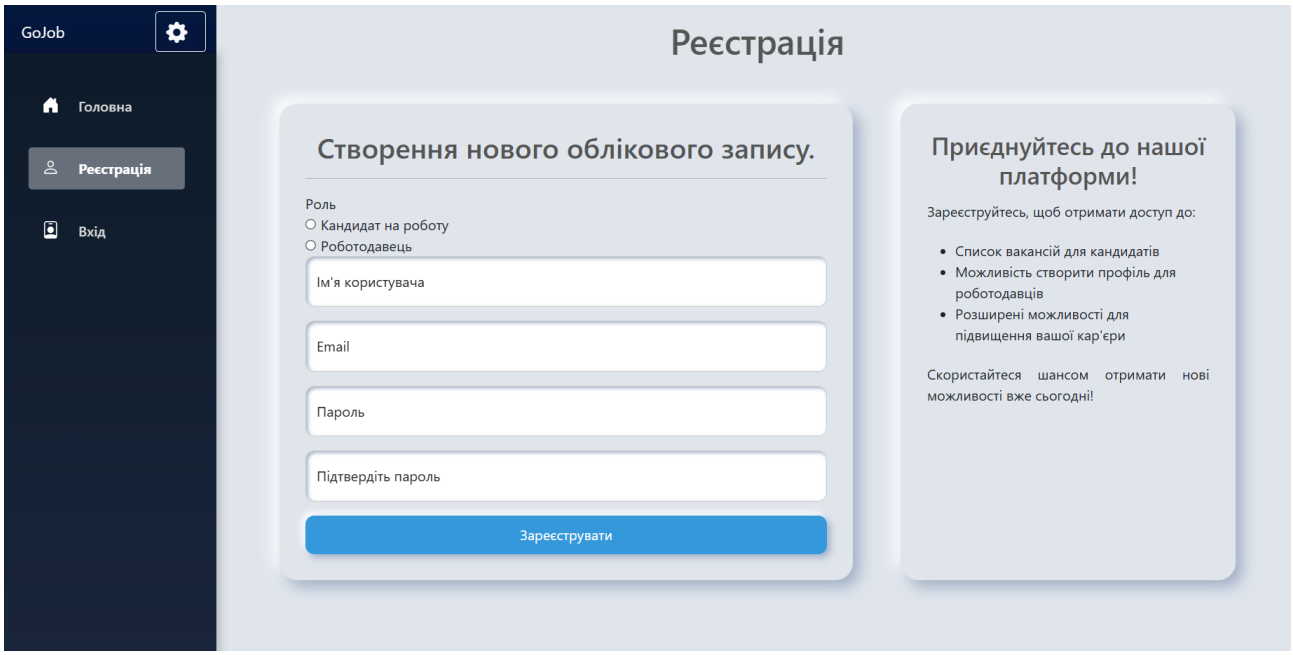

Рисунок 3.1 – Відображення реєстраційного модулю

Процес авторизації реалізований через компонент LoginForm з підтримкою різних методів входу. Система підтримує вхід через email та пароль або соціальні мережі. Перевірка облікових даних виконується на сервері через ASP.NET Core Identity. JWT токени генеруються після успішної автентифікації. Сесія користувача зберігається в локальному сховищі браузера. Механізм Refresh Token забезпечує автоматичне продовження сесії. Історія входів записується в базу даних для аудиту.

Компонент ForgotPassword дозволяє користувачам відновити доступ до облікового запису. Система генерує унікальне посилання для скидання паролю. Термін дії посилання обмежений для безпеки. Новий пароль перевіряється на відповідність політиці безпеки. Email з інструкціями відправляється через сервіс розсилки. Процес відновлення захищений від автоматизованих атак через CAPTCHA. Успішна зміна паролю завершує всі активні сесії користувача.

OAuth інтеграція реалізована через компонент ExternalLogin. Підтримуються популярні провайдери: Google, Facebook, LinkedIn. Процес автентифікації відповідає стандартним протоколам OAuth 2.0. Токени соціальних мереж валідуються на сервері. Базова інформація профілю імпортується при першому вході. Користувачі можуть пов’язати кілька соціальних акаунтів. Відключення зовнішнього провайдера не блокує локальний акаунт.

Компонент UserProfile дозволяє користувачам керувати особистими даними. Редагування профілю включає базову інформацію та налаштування. Завантаження аватара реалізовано через drag-and-drop інтерфейс. Зміна email вимагає повторної верифікації адреси. Налаштування приватності контролюють видимість даних профілю. Двофакторна автентифікація може бути активована користувачем. Історія змін профілю зберігається для аудиту.

Security компоненти забезпечують захист облікових даних. Система блокує акаунт після серії невдалих спроб входу. IP-адреси та браузери перевіряються на підозрілу активність. CSRF токени захищають форми від

підробки. Паролі зберігаються в базі даних у захешованому вигляді. XSS атаки блокуються через санітизацію введення. Сесії автоматично завершуються при бездіяльності.

ProfileVerification компонент реалізує процес підтвердження особи користувача. Верифікація email виконується через відправку коду підтвердження. Документи для верифікації завантажуються через захищений канал. Процес перевірки документів виконується модераторами системи. Статус верифікації відображається в профілі користувача. Верифіковані користувачі отримують додаткові можливості. Система зберігає історію верифікацій.

RoleManagement компонент керує правами доступу користувачів. Ролі визначають базовий набір дозволів для користувачів. Адміністратори можуть призначати та змінювати ролі. Система підтримує створення користувацьких ролей. Права доступу перевіряються при кожному запиті. Зміни ролей фіксуються в журналі аудиту. Конфлікти прав вирішуються за встановленими правилами.

SessionManagement відстежує та керує активними сесіями користувачів. Користувачі можуть переглядати список активних пристроїв. Підозрілі сесії позначаються для перевірки. Примусове завершення сесії доступне користувачу. Одночасні входи обмежуються налаштуваннями безпеки. Неактивні сесії автоматично завершуються. Дані сесій шифруються при зберіганні.

NotificationSystem інформує користувачів про операції з обліковим записом. Сповіщення надсилаються при зміні паролю або налаштувань безпеки. Push-повідомлення інформують про нові входи в акаунт. Email-сповіщення містять інформацію про критичні зміни. Користувачі можуть налаштувати типи сповіщень. Історія сповіщень зберігається в профілі. Шаблони повідомлень підтримують локалізацію.

3.4.2 Модуль управління профілями

Модуль управління профілями реалізує функціонал створення та редагування користувацьких профілів. Компонент ProfileEditor забезпечує інтерфейс для введення та оновлення даних. Система підтримує різні типи полів:

текстові, числові, дати, переліки. Завантаження фотографій реалізовано через drag-and-drop інтерфейс. Валідація даних виконується в реальному часі з відображенням підказок. Автозбереження захищає від втрати введених даних. Історія змін профілю зберігається для аудиту.

Блок персональних даних включає основну інформацію про користувача. Контактні дані проходять верифікацію через підтвердження email та телефону. Система підтримує множинні адреси та контакти. Геолокація автоматично визначає місцезнаходження користувача. Налаштування приватності контролюють видимість кожного поля. Мовні преференції впливають на локалізацію інтерфейсу. Часовий пояс налаштовується автоматично або вручну.

Професійний блок містить інформацію про досвід роботи та навички. Редактор досвіду роботи підтримує хронологічне впорядкування позицій. Навички додаються через систему тегів з автодоповненням. Рівні володіння навичками визначаються через шкалу оцінок. Сертифікати та дипломи можна завантажувати з попереднім переглядом. Портфоліо робіт організовано за категоріями. Рекомендації від колег підтверджуються через систему запитів.

Компонент налаштувань безпеки керує параметрами захисту профілю. Двофакторна автентифікація налаштовується через QR-код. Історія входів показує всі пристрої з доступом до акаунту. Зміна паролю вимагає підтвердження поточного паролю. Резервні коди генеруються для аварійного доступу. Сповіщення про безпеку налаштовуються користувачем. Блокування профілю доступне для термінового захисту.

Система сповіщень інформує про взаємодії з профілем. Push-повідомлення надсилаються через service worker. Email-сповіщення використовують HTML шаблони. SMS-сповіщення доступні для критичних подій. Налаштування дозволяють фільтрувати типи сповіщень. Групування повідомлень зменшує інформаційний шум. Архів сповіщень зберігає історію комунікацій.

Пошукові налаштування профілю визначають видимість для роботодавців. Ключові слова оптимізують пошукову видимість профілю. Приватність пошуку контролює доступність даних. Налаштування релевантності впливають на

позицію в результатах. Блокування компаній обмежує небажані контакти. Статистика переглядів профілю оновлюється щоденно. Режим активного пошуку підвищує видимість профілю.

Інтеграції з соціальними мережами розширюють профіль додатковими даними. LinkedIn синхронізує професійну інформацію. GitHub додає технічне портфоліо. Facebook імпортує соціальні зв’язки. Автоматичне оновлення підтримує актуальність даних. Налаштування конфіденційності контролюють імпорт. Відключення інтеграцій не видаляє імпортовані дані.

Модуль аналітики надає статистику активності профілю. Графіки показують динаміку переглядів профілю. Теплова карта відображає активність за часом. Джерела переходів аналізуються для оптимізації. Конверсія відгуків розраховується автоматично. Порівняння з середніми показниками оцінює ефективність. Експорт статистики доступний у різних форматах.

Система резервного копіювання захищає дані профілю. Автоматичні бекапи створюються щоденно. Експорт даних доступний у форматі JSON. Відновлення профілю можливе з будь-якої точки збереження. Шифрування захищає резервні копії. Вибіркове відновлення дозволяє повернути окремі дані. Архівація неактивних профілів економить ресурси.

Верифікація профілю підтверджує достовірність даних користувача. Документи перевіряються службою безпеки. Підтвердження освіти вимагає сканів дипломів. Рекомендації перевіряються через запити до роботодавців. Верифікований статус відображається спеціальною позначкою. Регулярні перевірки підтримують актуальність верифікації. Скасування верифікації можливе при порушеннях.

3.4.3 Модуль пошуку вакансій

Система пошуку вакансій базується на Elasticsearch для забезпечення швидкого повнотекстового пошуку. Компонент SearchBar реалізує інтерфейс введення пошукових запитів (рис. 3.2). Автодоповнення пропонує популярні пошукові терміни. Фільтри дозволяють уточнювати параметри пошуку, а результати оновлюються в реальному часі при їх зміні. Історія пошуку

зберігається для швидкого доступу до попередніх запитів. Релевантність результатів оптимізується через алгоритми ранжування.

Рисунок 3.2 – Відображення пошуку вакансій із введенням параметрів

Рисунок 3.3 демонструє параметри вибору компанії та відображення її опису, що доступне при перегляді вакансії.

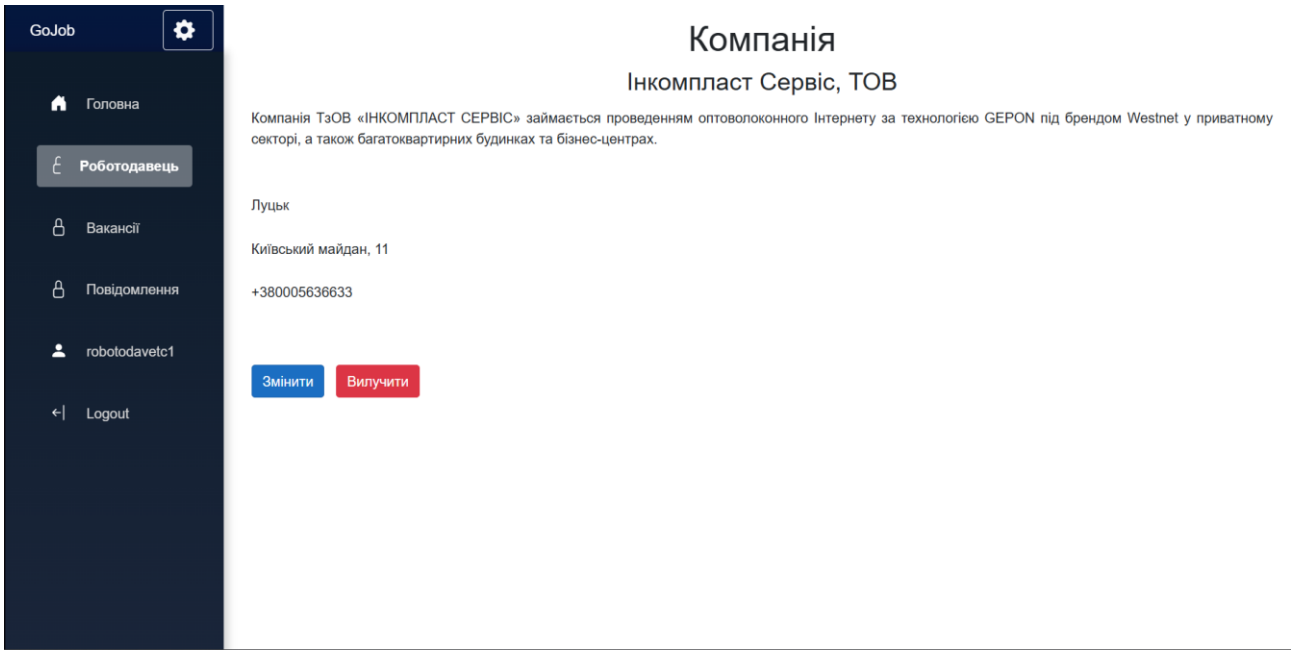

Рисунок 3.3 – Відображення параметрів та опису компанії

Після вибору конкретної компанії користувач може редагувати її дані.

Цей процес показано на рисунку 3.4, де продемонстровано виконання зміни інформації про компанію через інтерфейс.

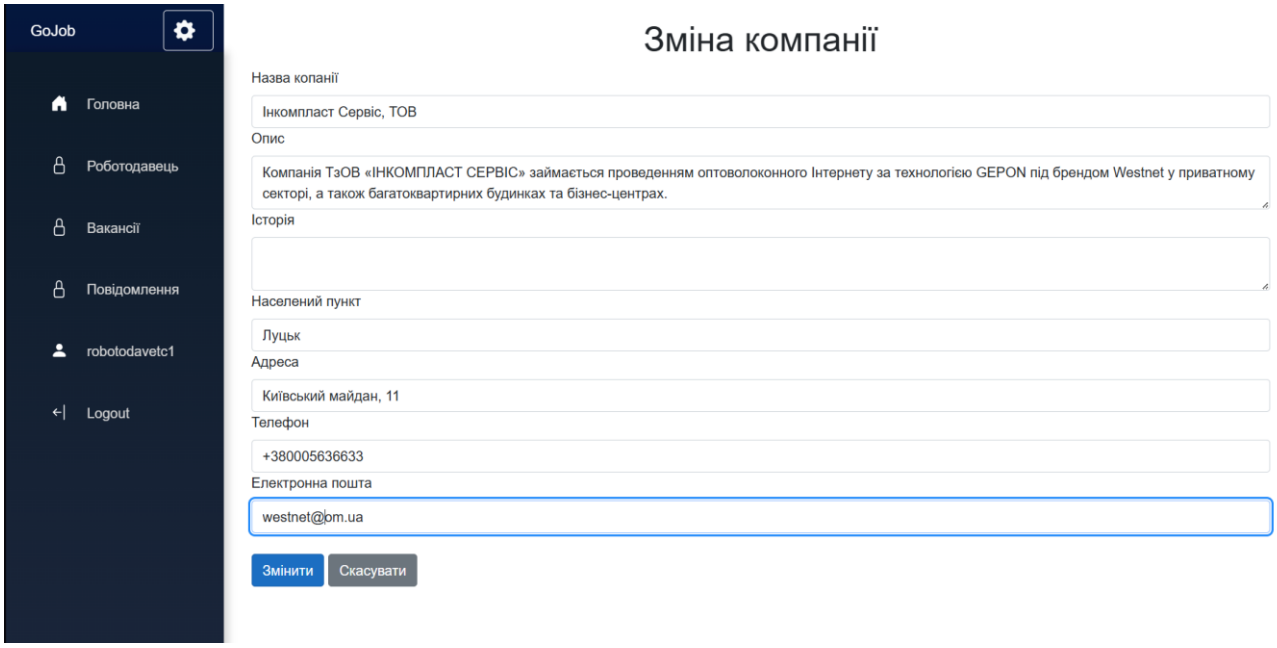

Рисунок 3.4 – Виконання зміни компанії

Фільтрація результатів здійснюється за множиною параметрів. Спеціалізація та категорії структурують вакансії. Географічне розташування визначається через геокодування адрес. Рівень заробітної плати фільтрується через діапазони. Тип зайнятості включає повну, часткову та віддалену роботу. Досвід роботи категоризується за рівнями. Додаткові умови включають соціальні пакети та бонуси.

Система сортування дозволяє упорядковувати результати пошуку. Дата публікації визначає актуальність вакансій. Рівень зарплати впливає на позицію в результатах. Рейтинг компаній враховується при сортуванні. Кількість відгуків показує популярність вакансії. Терміновість закриття позиції підвищує пріоритет. Релевантність профілю кандидата оптимізує порядок.

Картки вакансій відображають ключову інформацію про позиції (рис. 3.5). Заголовок та компанія виділяються візуально. Короткий опис надає огляд обов’язків. Вимоги та умови структуровані списками. Кнопки швидких дій

доступні для відгуку. Статистика переглядів оновлюється автоматично. Закладки дозволяють зберігати цікаві вакансії.

Рисунок 3.5 – Виконаний пошук актуальних вакансій

Детальна сторінка вакансії містить повну інформацію про позицію. Rich-text редактор форматує опис вакансії. Контакти рекрутера доступні для зв'язку. Карта показує розташування офісу. Схожі вакансії пропонуються на основі категорій. Кнопка швидкого відгуку відкриває форму заявки. Поділитися вакансією можна через соціальні мережі.

Система рекомендацій пропонує релевантні вакансії. Профіль кандидата аналізується для підбору позицій. Машинне навчання оптимізує точність рекомендацій. Поведінкові патерни враховуються в алгоритмі. Зворотний зв'язок покращує якість пропозицій. Персоналізовані рекомендації оновлюються щоденно. Email-розсилка інформує про нові matches.

Збережені пошуки автоматизують моніторинг вакансій. Користувачі можуть зберігати набори фільтрів. Сповіщення надсилаються при появі нових результатів. Періодичність оновлень налаштовується користувачем. Експорт

результатів доступний у різних форматах. Статистика показує динаміку ринку. Видалення пошуків не впливає на історію.

Аналітика пошуку надає інсайти про тренди ринку. Популярні запити аналізуються для оптимізації. Теплові карти показують географію вакансій. Сезонні тренди відстежуються автоматично. Зарплатні діапазони агрегуються за категоріями. Попит на навички оцінюється через частоту згадувань. Прогнози будуються на основі історичних даних.

Мобільна версія пошуку оптимізована для смартфонів. Інтерфейс адаптується під розмір екрану. Жести спрощують навігацію по результатах. Push-сповіщення інформують про нові вакансії. Офлайн-режим кешує останні результати. Голосовий пошук підтримується через WebSpeech API. Share API інтегрується з системними функціями.

Система геолокації покращує релевантність результатів. Визначення місцезнаходження працює через браузерний API. Радіус пошуку налаштовується користувачем. Маршрути до офісу розраховуються автоматично. Кластеризація показує щільність вакансій на карті. Фільтрація за транспортною доступністю. Часові зони враховуються при відображенні дат.

SEO оптимізація забезпечує видимість вакансій в пошукових системах. URL структура оптимізована для пошукових роботів. Meta-теги генеруються динамічно для кожної вакансії. Sitemap оновлюється при публікації нових позицій. Структуровані дані додаються через schema.org розмітку. Canonical URLs запобігають дублюванню контенту. AMP версії прискорюють завантаження на мобільних.

Система моніторингу відстежує продуктивність пошуку. Час відповіді вимірюється для оптимізації швидкості. Помилки пошуку логуються для аналізу. Навантаження на сервери балансується автоматично. Кешування результатів зменшує затримки. Метрики якості оцінюють релевантність результатів. Алерти сповіщають про проблеми продуктивності.

3.4.4 Модуль управління вакансіями

Модуль управління вакансіями забезпечує повний цикл створення та публікації вакансій роботодавцями. Компонент VacancyEditor реалізує інтерфейс для створення нових позицій. Rich-text редактор дозволяє форматувати опис вакансії. Система шаблонів прискорює створення типових вакансій. Попередній перегляд показує фінальний вигляд публікації. Автозбереження захищає від втрати даних. Валідація полів виконується в реальному часі.

Налаштування публікації визначають параметри розміщення вакансії. Термін публікації встановлюється через календар. Пріоритет впливає на позицію в результатах пошуку. Таргетинг дозволяє обмежити аудиторію показів. Бюджет рекламної кампанії розподіляється по днях. Статистика переглядів оновлюється щогодини. Статус публікації контролюється через панель управління.

Система модерації перевіряє вакансії перед публікацією. Автоматичні фільтри виявляють заборонений контент. Ручна перевірка виконується модераторами платформи. Зауваження надсилаються роботодавцю для виправлення. Відхилені вакансії потребують повторного розгляду. Термін модерації не перевищує 24 години. Історія модерації зберігається в системі.

Управління відгуками реалізовано через компонент ApplicationManager. Вхідні заявки сортуються за статусами розгляду. Профілі кандидатів доступні для швидкого перегляду. Система оцінок допомагає ранжувати кандидатів. Комунікація з кандидатами ведеться через вбудований месенджер. Шаблони відповідей прискорюють обробку заявок. Статистика конверсії розраховується автоматично.

Аналітика вакансій надає детальну статистику ефективності. Графіки показують динаміку переглядів та відгуків. Джерела трафіку аналізуються для оптимізації. Конверсія розраховується за різними метриками. Порівняння з середніми показниками оцінює ефективність. Прогнози будуються на основі історичних даних. Експорт звітів доступний у різних форматах.

Система тегів структурує вимоги до кандидатів. Навички додаються через автодоповнення з бази даних. Рівні володіння навичками визначаються через шкалу. Soft skills категоризуються окремим блоком. Мовні вимоги специфікуються за рівнями. Обов’язкові та бажані навички розділяються візуально. Пріоритетність навичок впливає на пошук.

Геолокація офісу реалізована через інтеграцію з картами. Адреса валідується через геокодування. Радіус пошуку встановлюється для віддаленої роботи. Маршрути транспорту розраховуються автоматично. Панорами вулиць доступні для перегляду. Кластеризація відображає множинні офіси. Часові пояси враховуються при вказанні графіку.

Система сповіщень інформує про активність по вакансії. Push-повідомлення надсилаються про нові відгуки. Email-дайджести агрегують статистику. SMS використовуються для термінових повідомлень. Налаштування дозволяють фільтрувати типи сповіщень. Шаблони повідомлень підтримують змінні. Історія сповіщень зберігається в системі.

Архівація вакансій зберігає історію публікацій. Закриті вакансії переміщуються в архів автоматично. Статистика зберігається для аналізу ефективності. Повторна публікація можлива з архівної версії. Очищення архіву виконується за розкладом. Експорт архівних даних доступний у CSV. Пошук працює по архівним вакансіям.

Інтеграції з job-бордами розширюють охоплення публікацій. API підтримує автоматичну синхронізацію вакансій. Статистика агрегується з різних платформ. Модерація враховує вимоги майданчиків. Бюджети рекламних кампаній розподіляються автоматично. Статуси публікацій оновлюються в реальному часі. Відгуки збираються в єдину систему.

3.4.5 Система відгуків та комунікації

Система відгуків реалізує повний цикл взаємодії між кандидатами та роботодавцями. Компонент ApplicationForm забезпечує структуроване подання заявок.

На рисунках 3.6 та 3.7 показано процес довантаження та завантаження резюме до кабінету через drag-and-drop інтерфейс, що дозволяє легко додавати файли для подальшого використання.

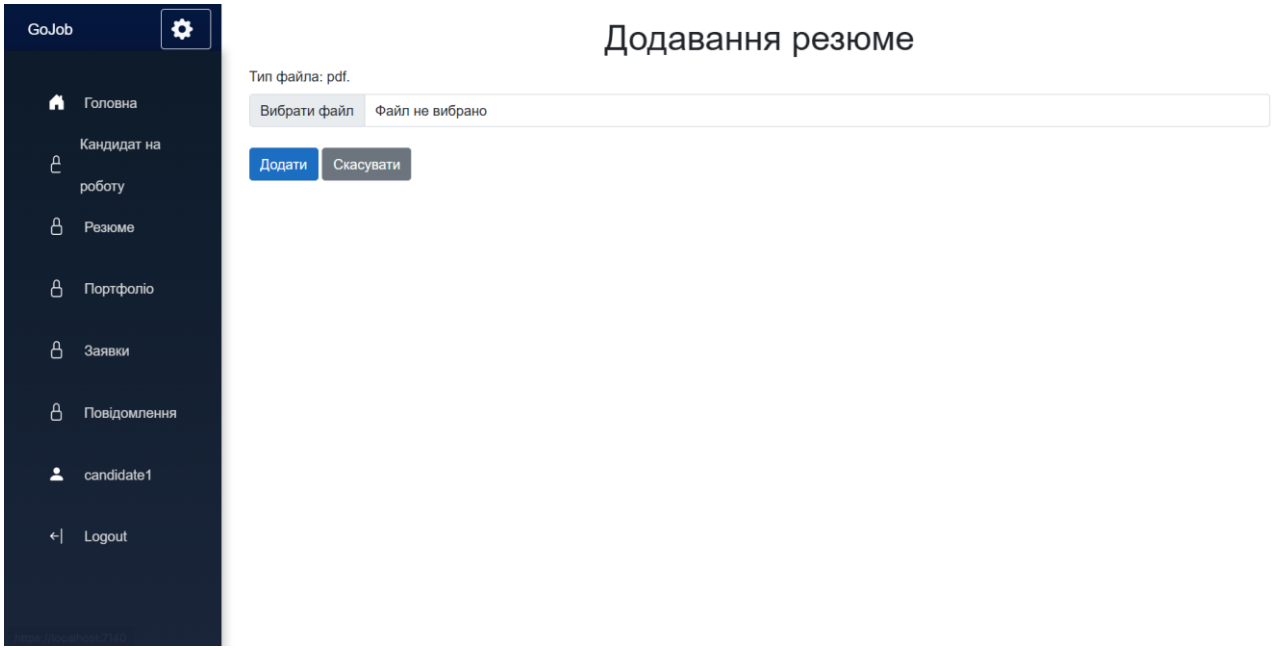

Рисунок 3.6 – Довантаження резюме до кабінету

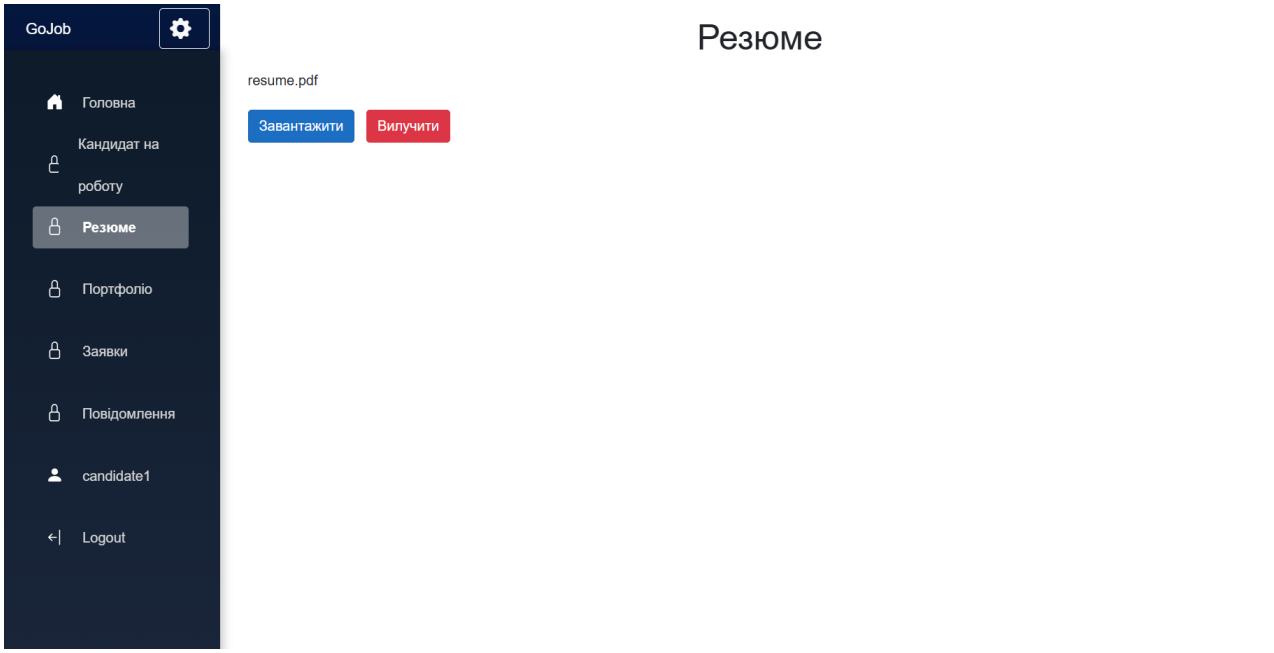

Рисунок 3.7 – Завантаження резюме до кабінету

Рисунок 3.8 демонструє подачу заявки на вакансію, де кандидат заповнює форму з особистими даними.

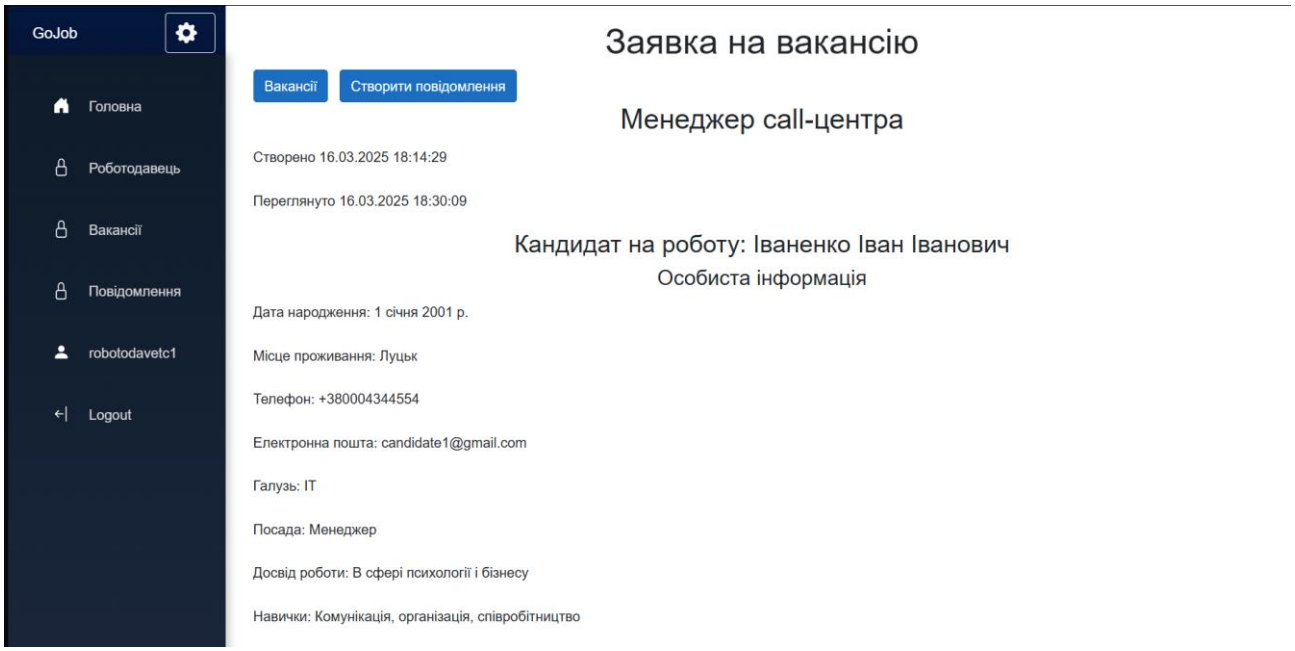

Рисунок 3.8 – Подача заявки на вакансію

Рисунок 3.9 показує вибір портфоліо для подачі заявки.

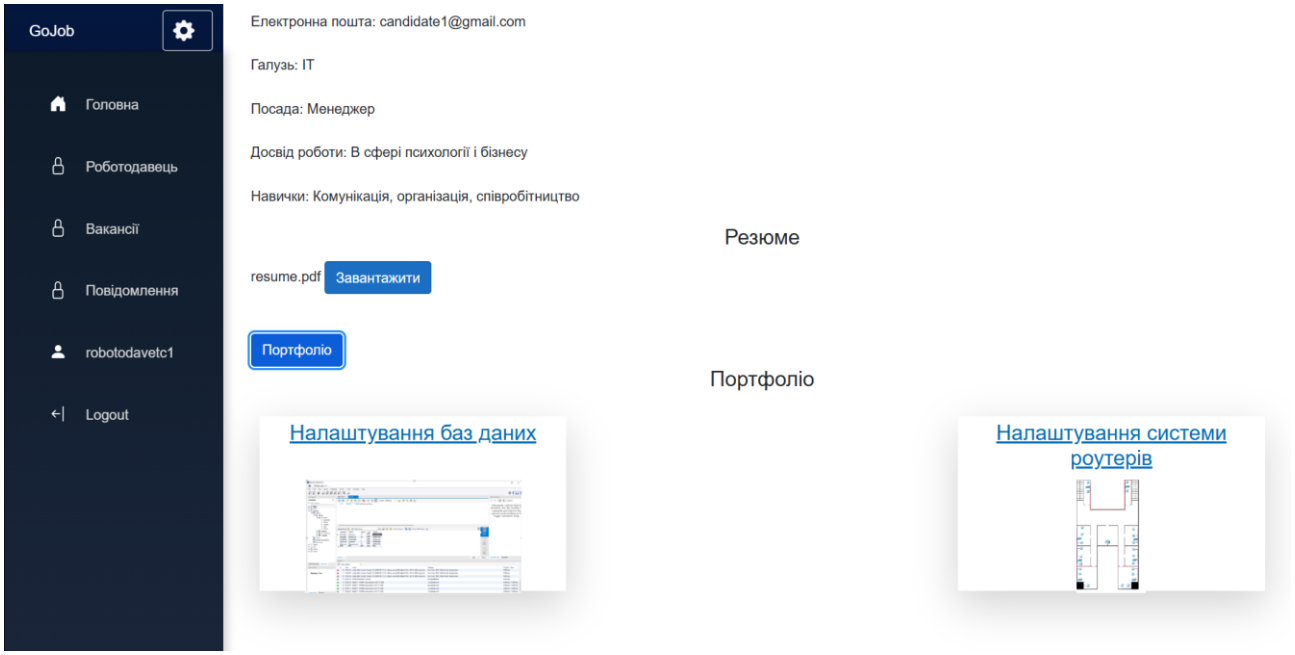

Рисунок 3.9 – Процес обирання портфоліо для подачі заявки

Супровідні листи форматуються через rich-text редактор. Файли резюме завантажуються через drag-and-drop інтерфейс. Попередній перегляд дозволяє перевірити коректність даних. Автозбереження захищає від втрати інформації. Валідація перевіряє повноту заповнення даних.

Статуси відгуків відображають етапи розгляду кандидатури. Система підтримує customization статусів під процеси компанії. Автоматичні переходи статусів налаштовуються через правила. Сповіщення надсилаються при зміні статусу. Коментарі фіксують причини зміни статусу. Часові мітки відстежують тривалість етапів. Дедлайни нагадують про необхідність оновлення статусу.

Месенджер забезпечує комунікацію між учасниками процесу. Чати підтримують обмін текстовими повідомленнями та файлами. Шаблони повідомлень прискорюють типові комунікації. Історія листування зберігається для аудиту. Статуси прочитання відстежують отримання повідомлень. Групові чати координують командні обговорення. Пошук працює по історії повідомлень.

Планувальник співбесід автоматизує процес узгодження зустрічей. Календар показує доступні слоти для інтерв'ю. Підтвердження зустрічей надсилається через email. Нагадування встановлюються автоматично для всіх учасників. Відеоконференції інтегруються через популярні сервіси. Перенесення зустрічей враховує зайнятість учасників. Статистика відвідуваності аналізується автоматично.

Система оцінювання кандидатів структурує зворотний зв'язок. Критерії оцінки налаштовуються під вимоги позиції. Вагові коефіцієнти визначають пріоритетність критеріїв. Порівняння кандидатів виконується через рейтинги. Коментарі пояснюють виставлені оцінки. Агреговані показники формують фінальний рейтинг. Експорт оцінок доступний для аналізу.

Тестові завдання перевіряють практичні навички кандидатів. Бібліотека завдань категоризується за спеціалізаціями. Автоматична перевірка доступна для технічних завдань. Часові обмеження контролюють тривалість виконання. Плагіат перевіряється через спеціальні алгоритми. Результати інтегруються в профіль кандидата. Статистика успішності аналізується по завданнях.

Відеозаписи співбесід зберігаються для подальшого аналізу. Система підтримує запис онлайн-інтерв'ю. Транскрипція генерується автоматично через Speech-to-Text. Пошук працює по тексту розмови. Закладки відмічають ключові моменти. Доступ до записів обмежується налаштуваннями приватності. Архівація виконується за розкладом.

Аналітика процесу найму надає метрики ефективності. Воронка рекрутингу показує конверсії по етапах. Час закриття вакансій розраховується автоматично. Джерела кандидатів аналізуються для оптимізації. Причини відмов категоризуються для аналізу. Прогнози будуються на основі історичних даних. Дашборди візуалізують ключові показники.

Система референсів автоматизує збір рекомендацій. Запити надсилаються автоматично після узгодження з кандидатом. Форми для відгуків структурують отриману інформацію. Нагадування відправляються при відсутності відповіді. Валідація перевіряє достовірність контактів. Результати інтегруються в профіль кандидата. Статистика відповідей аналізується автоматично.

Управління офертами автоматизує фінальний етап найму. Шаблони офертів налаштовуються під компанію. Узгодження умов відстежується через статуси. Електронне підписання доступне через інтеграції. Нагадування встановлюються для дедлайнів прийняття. Історія переговорів зберігається в системі. Аналітика прийняття оферів оцінює ефективність.

Onboarding процес інтегрується з системою найму. Чек-листи автоматизують адаптацію нових співробітників. Доступи до систем надаються автоматично. Навчальні матеріали призначаються відповідно до позиції. Прогрес адаптації відстежується через метрики. Зворотний зв'язок збирається регулярно. Статистика успішності аналізується по програмах.

Система оповіщень координує всіх учасників процесу. Push-сповіщення інформують про термінові події. Email-дайджести агрегують оновлення статусів. SMS використовуються для критичних повідомлень. Налаштування дозволяють фільтрувати типи сповіщень. Шаблони повідомлень підтримують персоналізацію. Статистика доставки аналізується автоматично.

Інтеграції з HR-системами синхронізують дані найму (рис. 3.10-3.11). API підтримує двосторонній обмін інформацією. Статуси оновлюються в реальному часі. Документи передаються в кадрові системи. Доступи налаштовуються через єдиний вхід. Аудит відстежує синхронізацію даних. Помилки інтеграції обробляються автоматично.

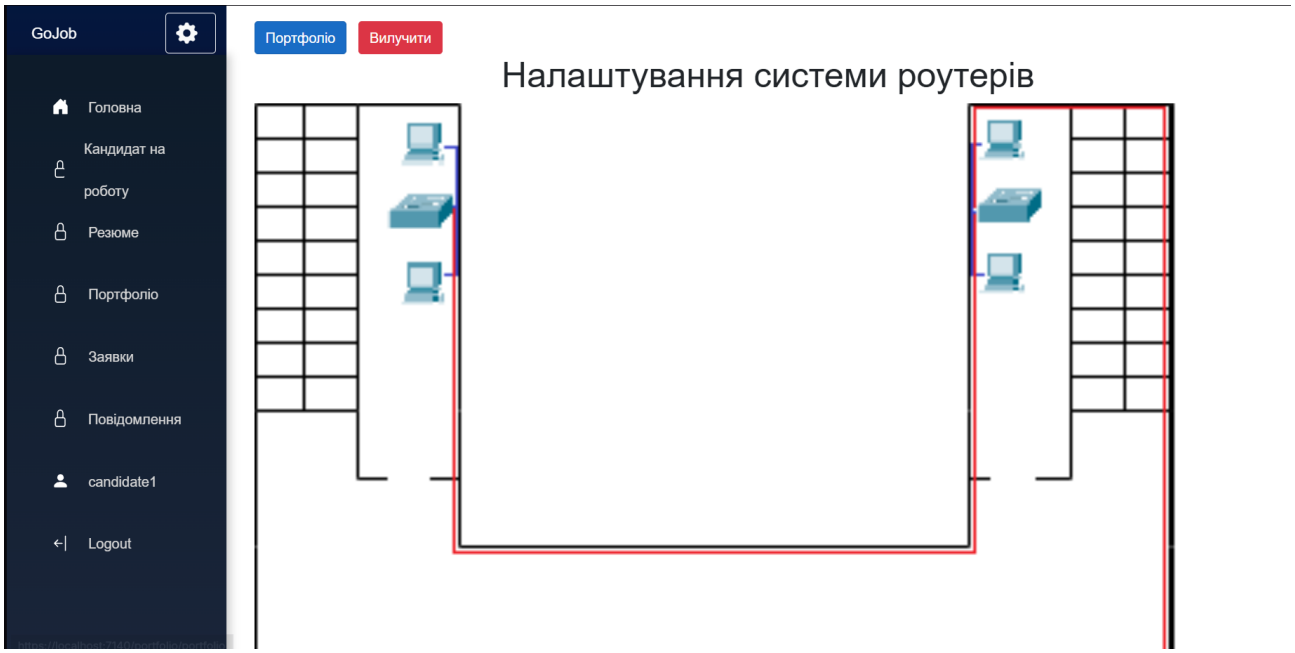

Рисунок 3.10 – Виконання дій з портфоліо та додавання рисунку

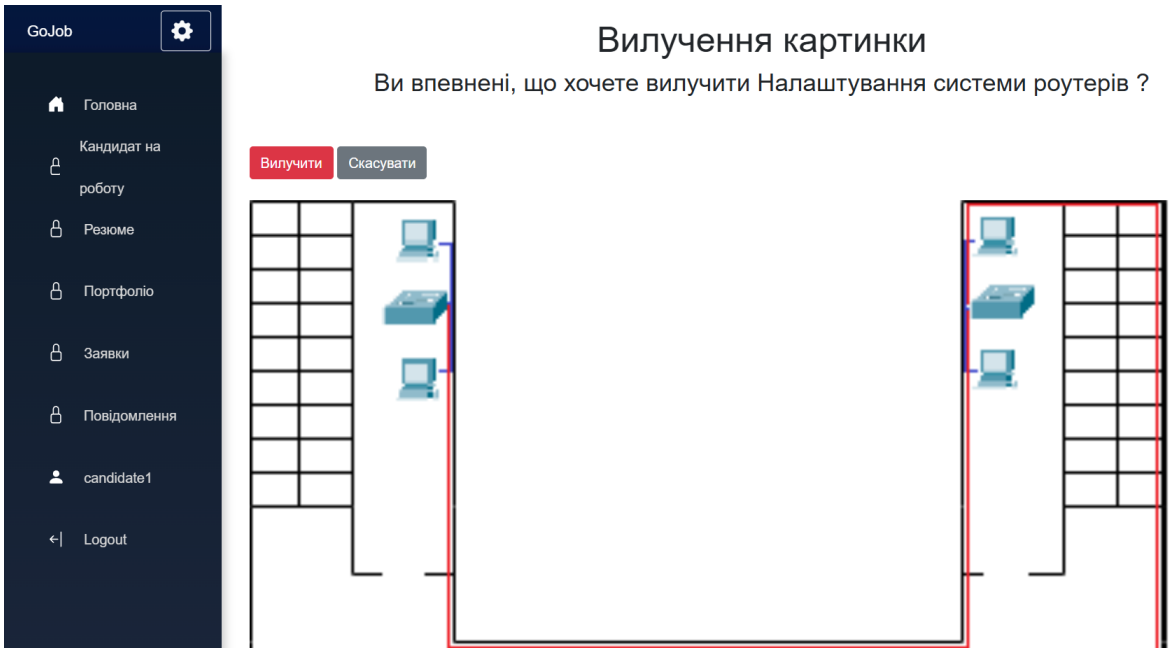

Рисунок 3.11 – Вилучення актуального портфоліо

3.5 Тестування розробленого програмного забезпечення

Модульне тестування компонентів Blazor виконується через фреймворк bUnit. Тести перевіряють коректність рендерингу компонентів. Події користувацького інтерфейсу симулюються через TestContext. Залежності компонентів замінюються моками для ізоляції. Асинхронні операції тестуються через спеціальні методи. Стан компонентів перевіряється після кожної дії. Покриття коду відстежується через тестові метрики.

Інтеграційне тестування перевіряє взаємодію між компонентами системи. Тестові сценарії охоплюють основні потоки користувачів. WebApplicationFactory створює тестове середовище. HTTP-запити симулюються через HttpClient. База даних замінюється тестовим контейнером. Автентифікація емулюється через тестові токени. Кешування тестується через mock-сервіси.

UI тестування автоматизується через Selenium WebDriver. Сценарії покривають критичні шляхи користувачів. Скріншоти фіксують візуальні регресії. Page Object патерн структурує тестовий код. Кросбраузерне тестування виконується в різних середовищах. Мобільні версії тестуються через емулятори. Продуктивність вимірюється через Performance API.

API тестування виконується через Postman колекції. Автоматизовані тести перевіряють всі endpoints. Схеми відповідей валідуються через JSON Schema. Авторизація тестується через різні ролі. Граничні випадки перевіряються через спеціальні набори даних. Продуктивність API вимірюється через навантажувальні тести. Моніторинг відстежує статуси відповідей.

Навантажувальне тестування проводиться через k6 фреймворк. Сценарії симулюють реальні патерни використання. Метрики збираються для аналізу продуктивності. Вузькі місця виявляються через профілювання. Масштабованість перевіряється через поступове збільшення навантаження. Стабільність оцінюється під тривалим навантаженням. Ресурси моніторяться через системні метрики.

Безпека тестується через спеціалізовані інструменти сканування. OWASP ZAP перевіряє поширені вразливості. SQL-ін’єкції тестуються через автоматизовані сценарії. XSS-атаки симулюються через тестові payload. CSRF-захист перевіряється через спеціальні тести. Шифрування даних тестується через криптографічні перевірки. Аутентифікація верифікується через різні сценарії.

Тестування локалізації перевіряє коректність перекладів. Ресурсні файли валідуються на повноту. Відображення перевіряється для різних мов. Форматування дат тестується для різних локалей. Спеціальні символи перевіряються в інтерфейсі. Напрямок тексту тестується для RTL мов. Переклади верифікуються через автоматизовані перевірки.

E2E тестування виконується через Cypress фреймворк. Бізнес-сценарії автоматизуються від початку до кінця. Взаємодії користувача записуються через тестовий рекордер. Асинхронні операції обробляються через спеціальні команди. Мережеві запити мокаються для стабільності. Відео записується для аналізу помилок. Звіти генеруються після кожного прогону.

Тестування доступності використовує спеціалізовані інструменти. WAVE перевіряє відповідність WCAG стандартам. Скрінрідери тестують навігацію інтерфейсу. Контраст кольорів перевіряється через автоматизовані тести. Keyboard navigation тестується через спеціальні сценарії. ARIA атрибути валідуються на коректність. Фокус елементів перевіряється через автоматизацію.

Регресійне тестування запускається після кожного релізу. Автоматизовані тести охоплюють критичну функціональність. Візуальні регресії відстежуються через скріншоти. Продуктивність порівнюється з попередніми версіями. Сумісність перевіряється через матрицю версій. Міграції даних тестуються через спеціальні сценарії. Откати перевіряються через відновлення бекапів.

Мутаційне тестування оцінює якість тестових наборів. PIT framework модифікує вихідний код. Виживання мутантів аналізується для покращення тестів. Покриття коду оцінюється через різні метрики. Складність тестів

аналізується через статичний аналіз. Дублювання тестів виявляється через спеціальні перевірки. Ефективність тестів оцінюється через статистику.

Тестування продуктивності фронтенду використовує Lighthouse. Метрики Core Web Vitals вимірюються автоматично. Завантаження ресурсів оптимізується через аналіз. JavaScript виконання профілюється для оптимізації. Розмір бандлів контролюється через спеціальні перевірки. Кешування перевіряється через симуляцію мережі. Офлайн-режим тестується через Service Worker. Автоматичні звіти допомагають швидко виявляти проблеми з продуктивністю. Регулярне тестування дозволяє підтримувати стабільну якість інтерфейсу.

Тестування PWA функціональності перевіряє мобільні можливості. Встановлення додатку тестується через емулятори. Push-повідомлення перевіряються через тестові сценарії. Офлайн-кешування валідується через Network throttling. Фонова синхронізація тестується через Background Sync API. Manifest перевіряється на коректність. Іконки тестуються для різних платформ.

Тестування безперервної інтеграції автоматизується через GitHub Actions. Тести запускаються після кожного пуша. Середовища розгортаються автоматично для тестування. Артефакти зберігаються для аналізу помилок. Нотифікації надсилаються про результати тестів. Метрики збираються для аналізу трендів. Паралельне виконання прискорює тестування. Відмови в тестах миттєво фіксуються для оперативного реагування.

Моніторинг тестового покриття здійснюється через Coverlet. Метрики збираються для різних типів тестів. Тренди аналізуються через історичні дані. Порогові значення контролюють якість покриття. Звіти генеруються в різних форматах. Інтеграція з CI/CD автоматизує збір метрик. Візуалізація допомагає аналізувати результати.

Документація тестів генерується автоматично через DocFX. Тестові сценарії описуються в markdown форматі. Приклади коду включаються через спеціальні теги. API документація оновлюється автоматично. Версіонування

відстежує зміни в документації. Пошук працює по всій документації. Інтеграція з CI/CD забезпечує актуальність.

3.6 Вимоги до апаратної частини

Для стабільного та ефективного функціонування сервісу на базі Blazor необхідно забезпечити відповідні апаратні ресурси. Мінімальні вимоги включають процесор із чотирьма ядрами, наприклад, Intel i5 або AMD Ryzen 5, оперативну пам’ять обсягом 16 ГБ, твердотільний накопичувач (SSD) на 100 ГБ для бази даних та логів, а також операційну систему Windows 7, 8 або Linux, наприклад, Ubuntu 20.04. У якості системи керування базами даних можна використовувати SQL Server Express.

Для кращої продуктивності рекомендується використовувати процесор із вісьмома ядрами, таким як Intel Xeon або AMD EPYC, 32 ГБ оперативної пам’яті, а також швидкий NVMe SSD обсягом 500 ГБ. Оптимальною операційною системою є Windows 10, 11 або Linux (Ubuntu 22.04, CentOS 8), а в якості бази даних краще використовувати SQL Server Standard у поєднанні з Redis для кешування.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено сервіс для пошуку роботи на базі фреймворку Blazor. Проведений аналіз існуючих рішень дозволив визначити основні функціональні вимоги до системи та обрати оптимальні технології розробки. За результатами дослідження було обрано фреймворк Blazor як основну технологію розробки, що забезпечує високу продуктивність, безпеку та зручність розробки завдяки використанню єдиної мови програмування C# як на сервері, так і на клієнті.

Архітектура системи спроектована на основі принципів чистої архітектури, що забезпечує модульність, масштабованість та легкість підтримки коду. Розроблені UML діаграми детально описують структуру та взаємодію компонентів системи, що спрощує розуміння архітектури та подальшу розробку. Система автентифікації та авторизації реалізована з використанням ASP.NET Core Identity та JWT токенів, забезпечуючи надійний захист користувацьких даних та гнучке розмежування прав доступу.

Модуль управління профілями користувачів надає кандидатам та роботодавцям розширені можливості для представлення себе на платформі. Система пошуку вакансій, реалізована на базі Elasticsearch, забезпечує швидкий та релевантний пошук з підтримкою складних фільтрів та сортування. Інтерфейс управління вакансіями для роботодавців включає функції публікації, модерації та аналітики ефективності розміщень.

Розроблена система відгуків та комунікації створює ефективне середовище для взаємодії між кандидатами та роботодавцями. Вбудований месенджер, планувальник співбесід та система оцінювання кандидатів оптимізують процес рекрутингу. Всебічне тестування програмного забезпечення, включаючи модульні, інтеграційні та навантажувальні тести, підтвердило надійність та стабільність системи.

Впроваджена система моніторингу та аналітики дозволяє відстежувати продуктивність системи та оптимізувати її роботу. Розроблений сервіс

відповідає сучасним вимогам до веб-додатків та надає користувачам повний набір інструментів для ефективного пошуку роботи та підбору персоналу. Система має значний потенціал для подальшого розвитку та масштабування.

Перспективними напрямками розвитку системи є впровадження алгоритмів машинного навчання для покращення релевантності пошуку та рекомендацій, розширення функціоналу аналітики та звітності, інтеграція з додатковими зовнішніми сервісами та job-бордами. Також планується розробка мобільних додатків для платформ iOS та Android та впровадження функціоналу відеоспівбесід безпосередньо на платформі.

Розроблений сервіс створює ефективну екосистему для взаємодії кандидатів та роботодавців, спрощуючи та оптимізуючи процес пошуку роботи та підбору персоналу. Використання сучасних технологій та архітектурних рішень забезпечує надійність, безпеку та високу продуктивність системи, а модульна структура дозволяє легко розширювати функціонал відповідно до потреб користувачів.

Практичне значення розробленої системи полягає у створенні зручного та ефективного інструменту для ринку праці, який допомагає кандидатам знаходити відповідні вакансії, а роботодавцям – підбирати кваліфікований персонал. Реалізовані функції аналітики та моніторингу дозволяють оптимізувати процеси рекрутингу та приймати обґрунтовані рішення на основі даних.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Official LinkedIn Blog. LinkedIn: Log In or Sign Up. URL: https://surl.li/dtbzjm (дата звернення: 24.02.2025).

2. Учасники проектів Вікімедіа. Indeed. Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Indeed (дата звернення: 24.02.2025).

3. Contributors to Wikimedia projects. Monster.com. Wikipedia. Wikipedia, the free encyclopedia. URL: https://surli.cc/pziwcl (дата звернення: 26.02.2025).

4. Glassdoor в Україні: огляд, вакансії та відгуки користувачів. Zarobitok. URL: https://surl.lu/ktydjw (дата звернення: 26.02.2025).

5. Робота в Києві і Україні. Work.ua. Work.ua. сайт пошуку роботи №1 в Україні. URL: https://www.work.ua/about-us/ (дата звернення: 28.02.2025).

6. Учасники проектів Вікімедіа. Djinni.co. Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Djinni.co (дата звернення: 05.03.2025).

7. Учасники проектів Вікімедіа. HeadHunter. Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/HeadHunter (дата звернення: 07.03.2025).

8. Why Join Dice. Find Jobs in Tech. Dice.com. URL: https://www.dice.com/why-join-dice (дата звернення: 09.03.2025).

9. ZipRecruiter. About. ZipRecruiter. URL: https://surl.li/aockol (дата звернення: 11.03.2025).

10. Учасники проектів Вікімедіа. Robota.ua. Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Robota.ua (дата звернення: 13.03.2025).

11. Про проет LinDeal (Універсальна Цифрова Платформа Знань). lindeal.com. URL: https://lindeal.com/about-us (дата звернення: 15.03.2025).

12. Getting Started. React. Wayback Machine. URL: https://surl.li/nkfxhs (дата звернення: 27.05.2025).

13. Introduction Vue.js. Wayback Machine. URL: https://surl.li/afsmnx (дата звернення: 17.03.2025).

14. Angular Docs. Surli.cc. Shortened links service from Surli. URL: https://surli.cc/dbsilq (дата звернення: 19.03.2025).

15. Docs. Node.js. Node.js. URL: https://surl.li/qllbda (дата звернення: 21.03.2025).

16. Учасники проектів Вікімедіа. ASP.NET. Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/ASP.NET (дата звернення: 23.03.2025).

17. ItProger. Спільнота програмістів! Short URL service. Surli. FREE Short Links. URL: https://surl.li/ixktmy (дата звернення: 30.03.2025).

18. Introduction to ASP.NET Core. Short URL service. Surli. FREE Short Links. URL: https://surl.li/enmree (дата звернення: 14.04.2025).

ДОДАТКИ

Додаток А

Програмний код для основного файлу Program.cs

```
using BlazorJobSearch.Components;
using BlazorJobSearch.Components.Account;
using BlazorJobSearch.Data;
using Microsoft.AspNetCore.Components.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using BlazorJobSearch.JobSearchServices;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorComponents()
    .AddInteractiveServerComponents();

builder.Services.AddCascadingAuthenticationState();
builder.Services.AddScoped<IdentityUserAccessor>();
builder.Services.AddScoped<IdentityRedirectManager>();
builder.Services.AddScoped<AuthenticationStateProvider,
IdentityRevalidatingAuthenticationStateProvider>();

builder.Services.AddAuthentication(options =>
    {
        options.DefaultScheme = IdentityConstants.ApplicationScheme;
        options.DefaultSignInScheme = IdentityConstants.ExternalScheme;
    })
    .AddIdentityCookies();
```

```
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ?? throw new InvalidOperationException("Connection string 'DefaultConnection' not found.");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString), optionsLifetime: ServiceLifetime.Singleton);
builder.Services.AddDbContextFactory<ApplicationDbContext>((DbContext OptionsBuilder options) =>
    options.UseSqlServer(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddIdentityCore<ApplicationUser>(options => options.SignIn.RequireConfirmedAccount = false)
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddSignInManager()
    .AddDefaultTokenProviders();
builder.Services.AddDistributedMemoryCache(); // Усі сесії працюють поверх об'єкта IDistributedCache. Додаю використання IDistributedCache.
builder.Services.AddSession(); // Додаю сервіси сесії.
builder.Services.AddScoped<AppData>();
//builder.Services.AddSingleton<AppData>();

builder.Services.AddSingleton<IEmailSender<ApplicationUser>, IdentityNoOpEmailSender>();

var app = builder.Build();

using (var scope = app.Services.CreateScope())
{
```

```
            var services = scope.ServiceProvider;
            try
            {
                ApplicationDbContext applicationIdentityDbContext =
services.GetRequiredService<ApplicationDbContext>();
                applicationIdentityDbContext.Database.Migrate();

                var userManager =
services.GetRequiredService<UserManager<ApplicationUser>>();
                var roleManager =
services.GetRequiredService<RoleManager<IdentityRole>>();
                string userRoleJobCandidate = "JobCandidate";
                string userRoleEmployer = "Employer";
                if (await roleManager.FindByNameAsync(userRoleJobCandidate) == null)
                {
                    await roleManager.CreateAsync(new
IdentityRole(userRoleJobCandidate));
                }
                if (await roleManager.FindByNameAsync(userRoleEmployer) == null)
                {
                    await roleManager.CreateAsync(new IdentityRole(userRoleEmployer));
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                var logger = services.GetRequiredService<ILogger<Program>>();
                logger.LogError(ex, ex.Message);
            }
        }
```

```
app.UseSession(); // Додаю middleware для роботи з сесіями.

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    app.UseExceptionHandler("/Error", createScopeForErrors: true);
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();

app.UseStaticFiles();
app.UseAntiforgery();

app.MapRazorComponents<App>()
    .AddInteractiveServerRenderMode();

// Add additional endpoints required by the Identity /Account Razor components.
app.MapAdditionalIdentityEndpoints();

app.Run();
```