НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок для автоматичного підбору вакансій на основі резюме та адаптації резюме за допомогою нейромереж для ІТ-галузі (комплексна тема). Вебзастосунок та агрегація вакансій

Виконав студент IV курсу, групи ІП-11
(шифр групи)
Рябов Юрій Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доц. каф.ІСТ, к.т.н., доц., Писаренко А. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____ _____
(підпис)

Київ – 2025

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Рябову Юрію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для автоматичного підбору вакансій на основі резюме та адаптації резюме за допомогою нейромереж для ІТ-галузі (комплексна тема). Вебзастосунок та агрегація вакансій

керівник проєкту Ліщук Катерина Ігорівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с

2. Термін подання студентом проєкту «16» червня 2025 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна компонентів програмного забезпечення _____

2) Схема структурна класів програмного забезпечення _____

_____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв’язання задачі	25.03.2025	
3	Постановка та формалізація задачі	05.04.2025	
4	Розробка інформаційного забезпечення	15.04.2025	
5	Алгоритмізація задачі	20.04.2025	
6	Обґрунтування вибору використаних технічних засобів	25.04.2025	
7	Розробка програмного забезпечення	15.05.2025	
8	Налагодження програми	20.05.2025	
9	Виконання графічних документів	26.05.2025	
10	Оформлення пояснювальної записки	28.05.2025	
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент _____ Юрій РЯБОВ

(підпис) (ініціали, прізвище)

Керівник _____ Катерина ЛІЩУК

(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з трьох розділів, містить 30 таблиць, 4 рисунки та 8 джерел – загалом 32 сторінки.

Дипломний проєкт присвячений розробці вебзастосунку для підбору вакансій на основі резюме та адаптації резюме.

Мета: створення вебзастосунку, котрий допомагає пошукачам роботи підбирати найбільш релевантні вакансії в ІТ-галузі та адаптувати зміст резюме користувача від вимоги обраних вакансій, що підвищує точність процесу працевлаштування для обох сторін на ринку праці.

В першому розділі сформульовано функціональні вимоги індивідуальної частини дипломного проєкту та сформовано матрицю трасування функціональних вимог для демонстрації відповідності вимог до варіантів використання проєкту, наведено постановку завдання на розробку програмного забезпечення.

В другому розділі описано архітектуру вебзастосунку, обґрунтування засобі розробки та наведено метод збору ІТ-вакансій з сайту dou.ua

В третьому розділі наведено оцінку якості розробленого вебзастосунку та описані результати функціонального тестування.

Програмне забезпечення впроваджено на хостингу Render.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, ВАКАНСІЇ, РЕЗЮМЕ, FIREBASE, .NET, ANGULAR

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 30 tables, 4 figures and 8 sources – in total 32 pages.

The diploma project is dedicated to the development of a web application for job search based on resumes and resume adaptation.

The purpose of the diploma project is to create a web application that helps job seekers find the most relevant vacancies in the IT industry and adapt the content of the user's resume to the requirements of the selected vacancies, which increases the accuracy of the employment process for both parties in the labour market.

The first section formulates the functional requirements of the individual part of the thesis project and forms a matrix for tracing functional requirements to demonstrate compliance with the requirements for project usage options, and provides a task statement for software development.

The second section describes the architecture of the web application, justifies the development tool, and presents a method for collecting IT vacancies from the dou.ua website.

The third section provides an assessment of the quality of the developed web application and describes the results of functional testing.

The software is hosted on Render.

KEYWORDS: WEB APPLICATION, VACANCIES, RESUME, FIREBASE, .NET, ANGULAR, FASTAPI.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-1122.045440.02.81	Пояснювальна записка	32	
3	А4	КПІ.ІП1122.045440.03.12	Текст програми	18	
4	А4	КПІ.ІП-1122.045440.04.51	Програма та методика тестування	6	
5	А3	КПІ.ІП-1122.045440.06.99	Графічний матеріал	2	

					КПІ .І П-1122.045440.00.90
Змі	Арк.	№ докум.	Пі дп.	Дата	
Розроб.		Рябов Ю І .			Ві домі сть дипломного проєкту
Переві р.		Лі щук К.І .			
Н.контр.		Лі щук К.І .			
Затв.		Жарі ков Е.В.			

Лі т.	Аркуш	Аркуші в
	1	

КПІ і м. І горя Сі корського ФІ ОТ
каф. І ПІ, гр. І П-11

Пояснювальна записка
до дипломного проєкту

на тему: Вебзастосунок для автоматичного підбору вакансій на основі резюме та адаптації резюме за допомогою нейромереж для ІТ-галузі (комплексна тема). Вебзастосунок та агрегація вакансій

КПІ.ІП-1122.045440.02.81

Київ – 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE – Integrated Development Environment – інтегроване середовище розробки.

API – Application programming interface, прикладний програмний Інтерфейс.

ІТ – Інформаційні технології.

БД – База даних.

ВСТУП

В умовах сучасного світу пошук роботи на IT ринку України є дуже трудомістким і може займати місяці, а деколи і роки. В рамках такого стану дуже важливим є пошук якнайбільш релевантних вакансій з усіх можливих сайтів роботи, та постійний їх моніторинг на предмет появи нових вакансій.

Дипломний проєкт присвячено розробці вебзастосунку для пошуку вакансій пошукачами роботи та отримання рекомендацій до адаптації резюме. Метою дипломного проєкту є створення вебзастосунку, котрий допомагає пошукачам роботи підбирати найбільш релевантні вакансії в IT-галузі та адаптувати зміст резюме користувача від вимоги обраних вакансій, що підвищує точність процесу працевлаштування для обох сторін на ринку праці.

Для досягнення поставленої мети дипломного проєкту в рамках індивідуальної частини роботи буде розроблено вебзастосунок для пошуку роботи та адаптації резюме, розроблено методи збору вакансій з різних сайтів пошуку роботи, а також методи автоматичного надсилання повідомлень про нові вакансії користувачам.

1 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Розроблення функціональних вимог

В таблиці 1.1 наведено загальну модель вимог до програмного забезпечення, а в таблиці 1.2 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити в таблиці 1.3.

Таблиця 1.1 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Система авторизації користувача.	FR-1	Високий	Високий
1.1	Реєстрація користувача.	FR-2	Високий	Високий
1.2	Авторизація користувача.	FR-3	Високий	Високий
1.3	Відновлення паролю	FR-4	Середній	Середній
1.4	Вихід із акаунту.	FR-5	Середній	Низький
2	Завантаження резюме	FR-6	Високий	Високий
3	Пошук вакансій	FR-7	Високий	Середній
3.1	Відображення характеристик вакансій	FR-8	Середній	Низький
3.2	Фільтрація вакансій	FR-9	Низький	Низький
4	Адаптація резюме	FR-10	Високий	Середній
4.1	Зчитування тексту вакансії за посиланням	FR-11	Середній	Низький
4.2	Відображення оцінки релевантності та рекомендацій	FR-12	Високий	Середній
5	Надсилання нотифікацій про нові вакансії	FR-13	Середній	Середній

Таблиця 1.2 – Перелік функціональних вимог

Назва	Опис
FR-1	Система авторизації користувача. При переході неавторизованим користувачем на вебсайт він переадресовується на сторінку входу в профіль. З неї він може перейти на сторінку реєстрації або відновлення паролю, а після входу користувач може вийти з профілю.
FR-2	Реєстрація користувача. Для реєстрації користувач має ввести імейл, ім’я та пароль та натиснути кнопку реєстрації. Якщо імейл не є валідним чи вже наявний в системі або одне з полів є порожнім, відображається відповідне повідомлення і кнопка не є активною. Після натиснення кнопки користувачу надсилається імейл про підтвердження імейл адреси, та на екрані відображається відповідне повідомлення. Після підтвердження імейлу користувач може увійти в профіль.
FR-3	Авторизація користувача. Після реєстрації і підтвердження імейлу користувач може увійти в профіль. Для цього він має ввести імейл та пароль та натиснути кнопку входу. Якщо імейл не є валідним або пароль є порожнім чи введено неправильно користувач отримує відповідне повідомлення і кнопка входу стає неактивною. Після успішного входу користувач переадресується на головну сторінку.
FR-4	Відновлення паролю. З сторінки входу в профіль можна перемкнутись на форму відновлення паролю. Там користувач має ввести свою імейл адресу, і якщо вона наявна в системі кнопка відновлення стає активною, інакше користувач отримує відповідне повідомлення про помилку. Після натискання кнопки користувачу надсилається імейл з посиланням, перейшовши по якому він може встановити новий пароль.

Продовження таблиці 1.2

FR-5	Вихід з акаунту. На усіх сторінках крім сторінки авторизації присутня кнопка виходу з профілю, після натиснення якої юзер виходить з акаунту та переадресується на сторінку авторизації.
FR-6	Завантаження резюме. На головній сторінці застосунку присутня кнопка завантаження резюме, після натиснення якої відкривається вікно вибору файлу, в якому можна вибрати pdf, docx або doc файл. Після вибору файлу і його збереження, користувачу відображається відповідне повідомлення.
FR-7	Пошук вакансій. На головній сторінці застосунку присутня кнопка пошуку вакансій, яка є активною якщо користувач має завантажене резюме в застосунку. Після натиснення кнопки користувач переадресується на сторінку результатів пошуку.
FR-8	Відображення характеристик вакансій. На сторінці результатів пошуку вакансії відображаються у вигляді списку, в якому присутня інформація про назву вакансії, оцінку релевантності до резюме користувача, її локацію та зарплатню якщо такі дані доступні. По кліку на назву вакансії користувач переадресовується на сайт вакансії.
FR-9	Фільтрація вакансій. Після отримання усіх результатів пошуку вакансій користувач може відфільтрувати їх за локацією, або наявністю інформації про зарплатню.

Продовження таблиці 2.2

FR-10	Адаптація резюме. На головній сторінці застосунку присутня кнопка пошуку вакансій, яка є активною якщо користувач має завантажене резюме в застосунку. Після натиснення кнопки відкривається вікно, в якому користувачу пропонується ввести посилання на вакансію або її текст. Після введення тексту вакансії кнопка отримання рекомендацій до резюме стає активною, і після її натиснення відображається вікно з результатом.
FR-11	Зчитування тексту вакансії за посиланням. У вікні завантаження вакансії для адаптації резюме присутнє поле вводу посилання на вакансії та кнопка “Зчитати”. Після натиснення кнопки текст вакансії зчитується з сайту та заповнюється в поле вводу тексту вакансії, після чого його можна відредагувати. Якщо текст не вдалось зчитати, відображається відповідне повідомлення, і користувачу пропонується ввести текст вручну або перевірити посилання.
FR-12	Відображення оцінки релевантності та рекомендацій. Після натиснення кнопки отримання рекомендацій до резюме відкривається вікно, я якому показується оцінка релевантності резюме користувача до введеної вакансії, та список ключових слів які варто додати в резюме.
FR-13	Надсилання нотифікацій про нові вакансії. Під час завантаження нових вакансій в базу даних застосунку відбувається перевірка їх релевантності до резюме користувача, і при співпадінні надсилається імейл з новими вакансіями.

Таблиця 1.3 – Матриця трасування функціональних вимог

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13
UC-01	+	+											
UC-02	+		+										
UC-03	+			+									
UC-04						+							
UC-05							+	+	+				
UC-06										+	+	+	
UC-07													+

1.2 Постановка завдання на розробку програмного забезпечення

Метою дипломного проєкту є створення вебзастосунку, котрий допомагає пошукачам роботи підбирати найбільш релевантні вакансії в IT-галузі та адаптувати зміст резюме користувача від вимоги обраних вакансій, що підвищує точність процесу працевлаштування для обох сторін на ринку праці.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

– проектування та розробка вебзастосунку;

– авторизація користувача;

– робота з резюме користувача (зчитування, збереження, підтримка можливості адаптації під вакансії);
– агрегація IT-вакансій зі сторонніх сайтів і їх збереження для подальшого аналізу;
– автоматизоване надсилання повідомлень користувачу з релевантними вакансіями.

Висновки до розділу

У першому розділі пояснювальної записки сформульовано функціональні вимоги індивідуальної частини дипломного проєкту та сформовано матрицю трасування функціональних вимог для демонстрації відповідності вимог до варіантів використання проєкту.

В результаті сформульовано постановку завдання на розробку індивідуальної частини програмного забезпечення.

2 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Архітектура програмного забезпечення

Для клієнтської сторони вебзастосунку було обрано використання Single-Page Application архітектури.

Серверна частина застосунку використовує трьохрівневу архітектуру, тобто складається з:

- API що надає інтерфейс для роботи з застосунком;
- Business logic layer що містить логіку застосунку;
- Data access layer для роботи з базою даних.

Серверна частина застосунку містить наступні компоненти:

– компонент користувачів: відповідає за збереження профілю користувача та його резюме та верифікацію юзера за допомогою запитів до Firebase [1]. При завантаженні резюме він також нотифікує API сервісу машинного навчання для препроцесінгу резюме;

– компонент нотифікацій: відповідає за відправлення імейлів користувачам, використовуючи запити до Mailjet [2];

– компонент скрапінгу: відповідає за збір описів вакансій за посиланням;

– компонент вакансій: відповідає за обробку запитів про отримання вакансій, їх описів та адаптацію резюме, використовуючи запити до API сервісу машинного навчання, дані з бази даних та компонент скрапінгу;

– компонент агрегації вакансій: відповідає за збір даних про вакансії та підтримання їх в актуальному стані за допомогою запитів до сайтів вакансій. Після отримання вакансій цей компонент передає нові вакансії API сервісу машинного навчання для препроцесінгу та отримання оцінок релевантності для користувачів, які направляє компоненту нотифікацій для направлення користувачам.

Діаграму компонентів в нотації C4 наведено на рисунку 2.1:

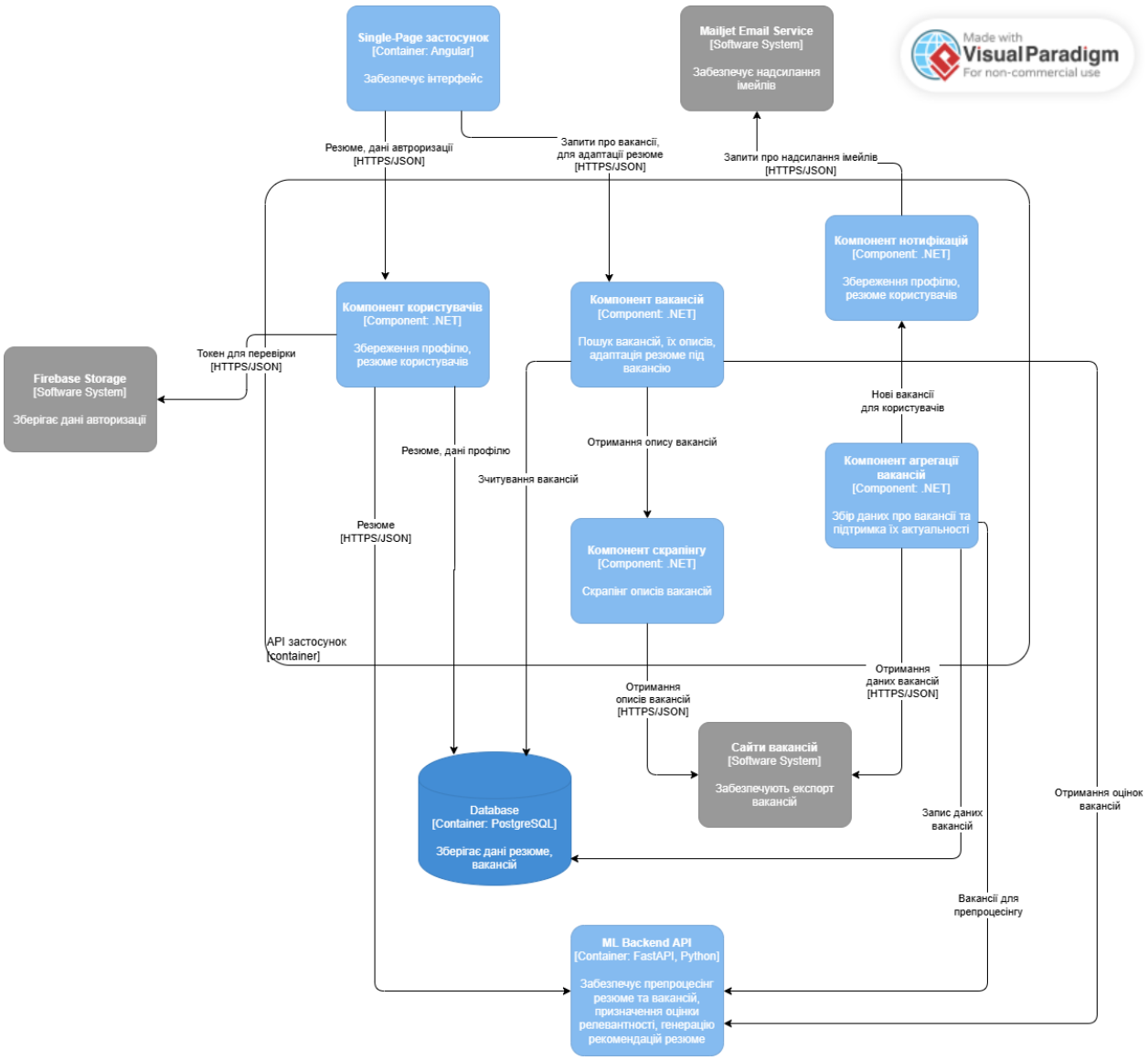


Рисунок 2.1 - Діаграма третього рівня С4 для API сервісу серверної частини вебзастосунку

Діаграму основних класів серверної частини застосунку наведено у графічному матеріалі, кресленні 2.

2.2 Архітектурні рішення та обґрунтування вибору засобів розробки

Для створення клієнтської сторони вебзастосунку було обрано використати фреймворк Angular [3], з використанням NgRx Store [4] для роботи з даними. NgRx Store ізолює логіку роботи з даними і виклики API сервісів від логіки представлення даних, що значно спрощує модифікацію коду та покращує його якість за рахунок чіткого розділення обов’язків.

Діаграма компонентів NgRx Store що використовуються для менеджменту стану клієнтської сторони застосунку наведена на рисунку 2.2:

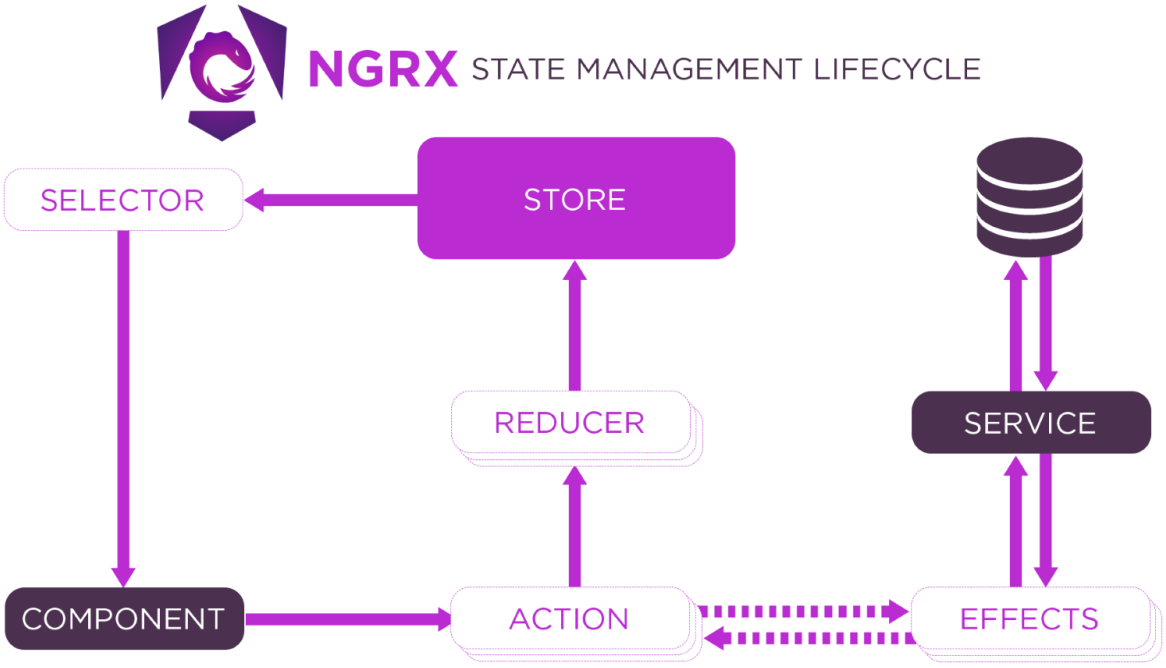


Рисунок 2.2 - Діаграма компонентів NgRx Store [4]

Для API сервісу застосунку було обрано використати веб-фреймворк ASP.NET [5]з використанням мови C# завдяки його продуктивності та простоті роботи з базою даних за допомогою Entity Framework [6].

Для авторизації користувачів було обрано використати сервіс Firebase завдяки безпеці даних користувачів, що він пропонує, його зручній інтеграції та гарній ціновій політиці, що дозволяє обмежене використання в рамках безкоштовної підписки. При реєстрації та авторизації користувача пароль та логін передаються Firebase за допомогою HTTPS запиту, після чого він надає токен користувача та його унікальний ідентифікатор, що використовуються для автентифікації запитів на API застосунку. Це забезпечує безпеку даних користувача, оскільки його пароль не зберігається ніде в самому застосунку, а повністю керується Firebase, який також надає функціонал підтвердження імейлу та відновлення паролю для додаткової безпеки.

Для надсилання імейлів користувачу було розглянуто використання одного з 2 сервісів: Mailjet та SendGrid [7]. Переваги SendGrid включають дуже

поширене використання та гарну підтримка в рамках інфраструктури .NET, але в безкоштовній його підписці дозволено лише 100 листів на день, впродовж 60 днів. Натомість безкоштовна підписка Mailjet дозволяє відправку до 200 листів на день без обмеження в часі підписки, і також має .NET бібліотеку для роботи з ним, хоч і менш популярну. Завдяки перевагам безкоштовної підписки для надсилання імейл повідомлень було обрано використання сервісу Mailjet.

В якості інструменту для розробки клієнтської частини застосунку було розглянуто Visual Studio Code та WebStorm. Перевагою WebStorm є те, що він є повноцінним IDE з нативною підтримкою Angular, що забезпечує зручні інструменти створення компонентів та рефакторингу. Значним недоліком Webstorm є швидкодія роботи і кількість ресурсів що він витрачає. Натомість, Visual Studio Code є дуже легковаговим редактором коду з дуже великою можливістю конфігурації за допомогою плагінів, завдяки чому було обрано саме його використання.

Для розробки серверної частини було розглянуто використання таких IDE як Rider та Visual Studio. Основною і практично єдиною суттєвою перевагою Visual Studio є наявність безкоштовної версії. Rider же у порівнянні з Visual Studio має більш зрозумілий інтерфейс, використовує менше пам’яті та має більш зручні та швидкі інструменти рефакторингу, завдяки чому було обрано саме його використання.

2.3 Конструювання програмного забезпечення

Опис основних методів серверної частини застосунку наведено в таблицях 3.1-3.13:

Таблиця 2.1 – Опис методів класу UserService

Назва методу	Призначення методу
GetCurrent	Отримання поточного користувача за токеном
Create	Створення користувача
CheckExisting	Перевірка існування користувача у базі даних з вказаним імейлом
AddResume	Створення резюме для користувача та надсилання його айді на апі машинного навчання для попередньої обробки
GetResumeId	Отримання айді резюме користувача
AddClaims	Додавання даних про айді користувача до токену авторизації

Таблиця 2.2 – Опис методів класу VacancyService

Назва методу	Призначення методу
Get	Отримання айді релевантних вакансій з апі машинного навчання та повернення повних моделей вакансій
AdaptResume	Отримання тексту вакансії за айді або як параметр в залежності від перегрузки, передача його та резюме користувача до апі машинного навчання та повернення результату
ParseVacancy	Отримання тексту вакансії за посиланням на неї, при наявності dou.ua в посиланні текст дістається з бази даних, при наявності postjobfree.com в посиланні використовується скрапер

Таблиця 2.3 – Опис методів класу PostJobFreeVacancyScrapper

Назва методу	Призначення методу
Scrape	Статичний метод для скрапінгу тексту вакансії з сайту postjobfree.com за посиланням

Таблиця 2.4 – Опис методів класу MachineLearningApiService

Назва методу	Призначення методу
NotifyResumeCreated	Надсилання повідомлення на апі машинного навчання про необхідність попередньої обробки резюме
NotifyVacanciesUpdated	Надсилання нових вакансій на апі машинного навчання з метою попередньої обробки та отримання їх оцінок релевантності для користувачів
GetVacancyScores	Надсилання айді резюме користувача на апі машинного навчання з метою отримання релевантних для нього вакансій
GetResumeAdaptation	Надсилання айді резюме та тексту вакансії з метою отримання рекомендацій до адаптації

Таблиця 2.5 – Опис методів класу TextExtractorService

Назва методу	Призначення методу
ReadPdf	Отримання тексту .pdf файлу
ReadDocx	Отримання тексту .docx файлу
ReadDoc	Отримання тексту .doc файлу

Таблиця 2.6 – Опис методів класу AggregatorJob

Назва методу	Призначення методу
Execute	Виклик агрегаторів вакансій з метою їх збору та збереження в базу даних, надсилання отриманих вакансій на апі машинного навчання, надсилання імейлів користувачам про нові релевантні для них вакансії

Таблиця 2.7 – Опис методів класу DouVacancyAggregatorService

Назва методу	Призначення методу
Aggregate	Ітерація по посиланням на експорт компаній dou.ua з метою збору з них вакансій, збереження їх у базі даних
GetVacancies	Отримання та мапінг вакансій за посиланням на експорт

Таблиця 2.8 – Опис методів класу PostJobVacancyAggregatorService

Назва методу	Призначення методу
Aggregate	Отримання вакансій з postjobfree.com по всім необхідним категоріям за допомогою скрапера, збереження їх в базі даних

Таблиця 2.9 – Опис методів класу PostJobVacanciesScrapper

Назва методу	Призначення методу
Scrape	Скрапінг вакансій з сторінки пошуку вакансій сайту postjobfree.com за текстом пошуку

Таблиця 2.10 – Опис методів класу EmailService

Назва методу	Призначення методу
SendAsync	Надсилання імейл повідомлень, використовуючи апі Mailjet

Таблиця 2.11 – Опис методів класу ScrapperJob

Назва методу	Призначення методу
StartAsync	Запуск збору посилань на компанії Dou при першому запуску серверу

Таблиця 2.12 – Опис методів класу CompanyService

Назва методу	Призначення методу
Create	Створення записів у таблиці компаній в базі даних
HasCompanies	Перевірка наявності компаній в базі даних

Таблиця 2.13 – Опис методів класу DouCompanyScrapper

Назва методу	Призначення методу
Scrape	Скрапінг посилань на топ1500 компаній dou.ua зі сторінки компаній

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.14.

Таблиця 2.14 - Опис утиліт та програмних бібліотек

№ п/п	Назва утиліти	Опис застосування
1	Jetbrains Rider	Головне середовище розробки серверні частини дипломного проєкту
2	Visual Sudio Code	Головне середовище розробки клієнтської частии дипломного проєкту
3	Entity Framework	ORM фреймворк для роботи з базою даних
4	ASP.NET Core	Фреймворк для побудови веб-апі застосунку
5	Selenium	Бібліотека для скрапінгу даних
6	PdfPig	Бібліотека для зчитування резюме у форматі pdf
7	NPOI	Бібліотека для зчитування резюме у форматі doc
8	OpenXML	Бібліотека для зчитування резюме у форматі docx
9	Quartz	Бібліотека для автоматичного запуску коду агрегації вакансій з заданою періодичністю
10	AutoMapper	Бібліотека для мапінгу моделей застосунку

2.3.1 Розробка методу збору вакансій

В рамках розробки програмного забезпечення дипломного проєкту постала задача зчитування вакансій з сайту dou.ua. Основними проблемами, які мав вирішувати метод, котрий розробляється, була відсутність єдиного API сайту для експорту вакансій та необхідність підтримки актуального списку вакансій у програмному забезпеченні.

Для збору вакансій було вирішено використовувати експорт вакансій компаній ДОУ, доступний за посиланням формату https://jobs.dou.ua/companies/{companyName}/vacancies/export/. За посиланням такого типу від ДОУ можна отримати JSON-файл, що містить всю інформацію про вакансії певної компанії. Таким чином задача зводиться до збору назв компаній, присутніх на ДОУ, та збір вакансій для кожної з них.

Для отримання списку назв компаній при першому запуску програмного забезпечення запускається вебскрапер, що переходить за посиланням https://jobs.dou.ua/companies/ та ініціює пагінацію, поки на сторінці не буде зображено не менше ніж 1500 компаній. Після цього він зчитує посилання на кожну з них у форматі https://jobs.dou.ua/companies/{companyName}/ та записує до бази даних програмного забезпечення.

Після цього та кожного дня протягом роботи програмного забезпечення запускається агрегатор вакансій, який для кожного отриманого посилання на компанію додає /vacancies/export та таким чином виконує експорт вакансій для кожної компанії, порівнює з наявними у базі даних та проводить синхронізацію. Таким чином в базі даних буде підтримуватись актуальний список більшості вакансій ДОУ з актуальністю на момент відпрацювання агрегатора раз на день.

Висновки до розділу

В рамках другого розділу пояснювальної записки було описано архітектуру індивідуальної частини дипломного проєкту та архітектурні рішення, пов’язані з нею. Після цього було наведено опис основних методів

серверної частини застосунку, а також утиліти та програмні бібліотеки, використані в розробці програмного забезпечення і описано алгоритм збору вакансій.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Для статичного аналізу коду серверної частини вебзастосунку було обрано використання вбудований в Jetbrains Rider статичний аналізатор Qodana [8]. Він дозволяє легко проаналізувати кодову базу на платформі .NET та виявити потенційні проблеми. Слід зауважити, що статичні аналізатори не можуть проаналізувати код, що використовує рефлексію, тому часто серед валідних зауважень надає помилки про не використання типів чи їх членів, хоча ті і використовуються неявно.

Звіт Qodana про кодову базу наведено на рисунку 3.1:

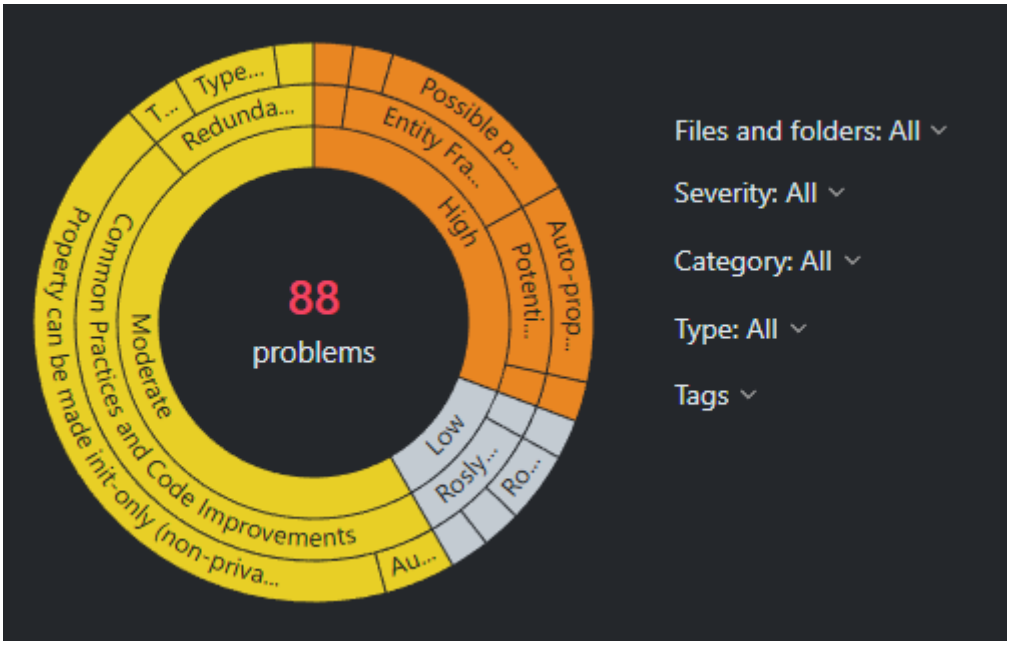


Рисунок 3.1 – Звіт Qodana

Звіт показує наявність 88 проблем із кодом, серед яких 56 середньої суворості і 26 – високої, що вже є непоганим показником. Проте як було сказано вище, серед цих проблем присутні багато таких, що пов’язані з відсутністю явного використання елементів коду, і не є валідними. Результат перевірки після вилучення таких помилок зі звіту наведено на рисунку 3.2:

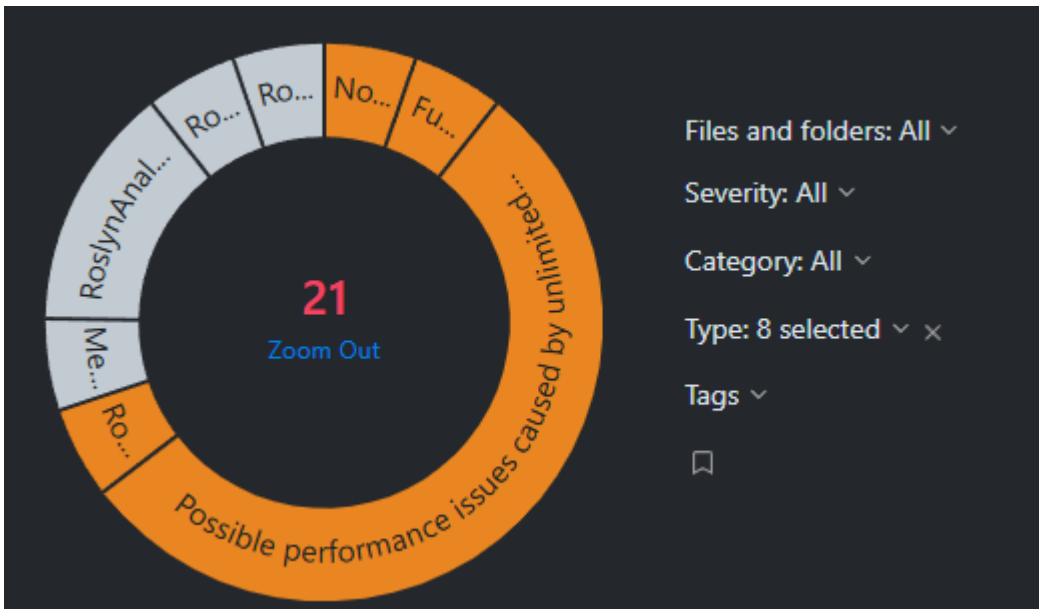


Рисунок 3.2 – Звіт Qodana після фільтрації типів проблем

Як бачимо, усі помилки середньої суворості були пов’язані з вказаною вище проблемою статичних аналізаторів, і після фільтрації типів помилок залишилось лише 15 проблем високої суворості, з яких 12 – потенційні проблеми з продуктивністю застосунку через необмежені за довжиною строки в базі даних, що безумовно є потенційною проблемою на яку варто звернути увагу, але в рамках предметної області роботи з текстами вакансій та резюме не обмежені за довжиною строки є необхідністю.

Загалом за звітом статичного аналізатора можна зробити висновок про якісне написання коду, з невеликою кількістю проблем.

3.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Основним методом тестування програмного забезпечення було мануальне тестування. Опис проведених тестів наведено у таблицях 3.1-3.15

Таблиця 3.1 – Тест 1.1 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, ім’я, пароль
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, яка не наявна у базі даних, ім’я користувача та пароль, користувач натискає кнопку реєстрації
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему, відображається повідомлення про надісланий
Фактичний результат	Збігається з очікуваним.

Таблиця 3.2 – Тест 1.2 Спроба входу в профіль до верифікації імейлу

Початковий стан системи	Користувач знаходиться на сторінці реєстрації, користувач зареєструвався але не підтвердив імейл
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, коректний пароль, користувач натискає кнопку входу
Очікуваний результат	Відображається повідомлення про те що користувач має підтвердити імейл перед входом
Фактичний результат	Збігається з очікуваним.

Таблиця 3.3 – Тест 1.3 Вхід в профіль

Початковий стан системи	Користувач знаходиться на сторінці реєстрації, користувач зареєструвався та підтвердив імейл
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, коректний пароль, користувач натискає кнопку входу
Очікуваний результат	Користувач переадресовується на головну сторінку застосунку і відображається повідомлення про успішну авторизацію
Фактичний результат	Збігається з очікуваним.

Таблиця 3.4 – Тест 1.4 Відновлення паролю

Початковий стан системи	Користувач знаходиться на сторінці реєстрації, користувач хоче відновити пароль
Вхідні дані	Електронна пошта
Опис проведення тесту	У відповідне поле вводиться коректна електронна пошта, що присутня у системі, користувач натискає кнопку відновлення паролю
Очікуваний результат	Відображається повідомлення про надісланий імейл відновлення паролю, користувачу надсилається імейл з посиланням, після переходу по якому він вводить новий пароль, за допомогою якого може увійти в профіль
Фактичний результат	Збігається з очікуваним.

Таблиця 3.5 – Тест 1.5 Завантаження резюме у форматі PDF

Початковий стан системи	Користувач знаходиться на головній сторінці, користувач не має завантаженого резюме
Вхідні дані	Резюме у форматі PDF
Опис проведення тесту	Користувач натискає кнопку завантаження резюме, обирає резюме у форматі PDF
Очікуваний результат	Відображається повідомлення про успішне завантаження резюме, кнопки пошуку вакансій та адаптації стають активними
Фактичний результат	Збігається з очікуваним.

Таблиця 3.6 – Тест 1.6 Завантаження резюме у форматі .doc

Початковий стан системи	Користувач знаходиться на головній сторінці, користувач не має завантаженого резюме
Вхідні дані	Резюме у форматі .doc
Опис проведення тесту	Користувач натискає кнопку завантаження резюме, обирає резюме у форматі .doc
Очікуваний результат	Відображається повідомлення про успішне завантаження резюме, кнопки пошуку вакансій та адаптації стають активними
Фактичний результат	Збігається з очікуваним.

Таблиця 3.7 – Тест 1.7 Завантаження резюме у форматі .docx

Початковий стан системи	Користувач знаходиться на головній сторінці, користувач не має завантаженого резюме
Вхідні дані	Резюме у форматі .docx
Опис проведення тесту	Користувач натискає кнопку завантаження резюме, обирає резюме у форматі .docx
Очікуваний результат	Відображається повідомлення про успішне завантаження резюме, кнопки пошуку вакансій та адаптації стають активними
Фактичний результат	Збігається з очікуваним.

Таблиця 3.8 – Тест 1.8 Пошук вакансій

Початковий стан системи	Користувач знаходиться на головній сторінці, користувач завантажив резюме
Вхідні дані	-
Опис проведення тесту	Користувач натискає кнопку пошуку вакансій
Очікуваний результат	Користувач переадресовується на сторінку пошуку вакансій, на ній присутні вакансії за резюме користувача, відсортовані за оцінкою релевантності. На кожній вакансії присутня інформація про назву позиції та локацію, оцінку релевантності, та зарплатню за наявністю такої інформації
Фактичний результат	Збігається з очікуваним.

Таблиця 3.9 – Тест 1.9 Фільтрація вакансій

Початковий стан системи	Користувач знаходиться на сторінці результатів пошуку
Вхідні дані	Текст для пошуку вакансій, локація
Опис проведення тесту	Користувач вводить дані у поле пошуку вакансій за назвою, обирає локацію з переліку
Очікуваний результат	Відображаються тільки вакансії, в назві яких є введений в поле пошуку текст, а в переліку локацій є обрана користувачем
Фактичний результат	Збігається з очікуваним.

Таблиця 3.10 – Тест 1.10 Адаптація вакансій з пошуку

Початковий стан системи	Користувач знаходиться на сторінці результатів пошуку
Вхідні дані	Вакансія
Опис проведення тесту	Користувач натискає на кнопку адаптації резюме для однієї з вакансій в пошуку.
Очікуваний результат	Відображається вікно з оцінкою співпадіння ключових слів резюме користувача та обраної вакансії, ключові слова пов’язані з навичками що наявні у вакансії але відсутні у резюме
Фактичний результат	Збігається з очікуваним.

Таблиця 3.11 – Тест 1.11 Зчитування тексту вакансії за посиланням

Початковий стан системи	Користувач знаходиться на головній сторінці, має завантажене резюме
Вхідні дані	Посилання на вакансію з сайту dou або postjobfree
Опис проведення тесту	Користувач натискає на кнопку адаптації резюме. Користувач вводить у поле посилання на вакансію та натискає кнопку зчитування тексту вакансії
Очікуваний результат	Поле тексту вакансії заповнюється згідно даних вакансії за введеним посиланням
Фактичний результат	Збігається з очікуваним.

Таблиця 3.12 – Тест 1.12 Зчитування тексту вакансії за не валідним посиланням

Початковий стан системи	Користувач знаходиться на головній сторінці, має завантажене резюме
Вхідні дані	Не валідне посилання на вакансію
Опис проведення тесту	Користувач натискає на кнопку адаптації резюме. Користувач вводить у поле текст що не є посиланням на вакансію dou, postjobfree і натискає на кнопку зчитування тексту
Очікуваний результат	Відображається повідомлення про не валідність посилання
Фактичний результат	Збігається з очікуваним.

Таблиця 3.13 – Тест 1.13 Адаптація резюме за текстом вакансії

Початковий стан системи	Користувач знаходиться на головній сторінці, має завантажене резюме
Вхідні дані	Текст вакансії або посилання
Опис проведення тесту	Користувач відкриває вікно адаптації резюме, вводить посилання на вакансію і натискає зчитування або вводить текст вручну, натискає кнопку адаптації резюме
Очікуваний результат	Відображається вікно з оцінкою співпадіння ключових слів резюме користувача та обраної вакансії, ключові слова пов’язані з навичками що наявні у вакансії але відсутні у резюме
Фактичний результат	Збігається з очікуваним.

Висновки до розділу

В третьому розділі пояснювальної записки до індивідуальної частини дипломного проєкту було проаналізовано якість програмного забезпечення за допомогою статичного аналізатора Qodana та зроблено висновки про якість програмного забезпечення, та наведено тести які були використані для мануального тестування вебзастосунку.

ВИСНОВКИ

В індивідуальній частині дипломного проєкту описано аріхтектуру вебзастосунку для автоматичного пошуку вакансій та адаптації резюме під вакансії, розроблено алгоритми агрегації вакансій з сайтів пошуку роботи та автоматичного надсилання користувачам імейлів про нові релевантні вакансії за їх резюме.

В першому розділі сформульовано функціональні вимоги індивідуальної частини дипломного проєкту та сформовано матрицю трасування функціональних вимог для демонстрації відповідності вимог до варіантів використання проєкту, наведено постановку завдання на розробку програмного забезпечення.

В лругому розділі описано архітектуру вебзастосунку, обґрунтування засобі розробки та наведено метод збору IT-вакансій з сайту dou.ua

В третьому розділі наведено оцінку якості розробленого вебзастосунку та описані результати функціонального тестування.

Завдяки реалізації збору вакансій з різних сайтів було досягнуто мету дипломного проєкту - пришвидшення пошуку роботи користувачами, оскільки їм не потрібно переглядати декілька сайтів для пошуку роботи, а також було усунено необхідність моніторингу таких сайтів завдяки реалізації автоматичних повідомлень про нові вакансії, що підходять до резюме користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Firebase. URL: https://firebase.google.com/ (дата звернення: 28.05.2025)

2) Mailjet. URL: https://www.mailjet.com/ (дата звернення: 28.05.2025)

3) Angular. URL: https://angular.dev/ (дата звернення: 28.05.2025)

4) NgRx Store. URL: https://ngrx.io/guide/store (дата звернення: 28.05.2025)

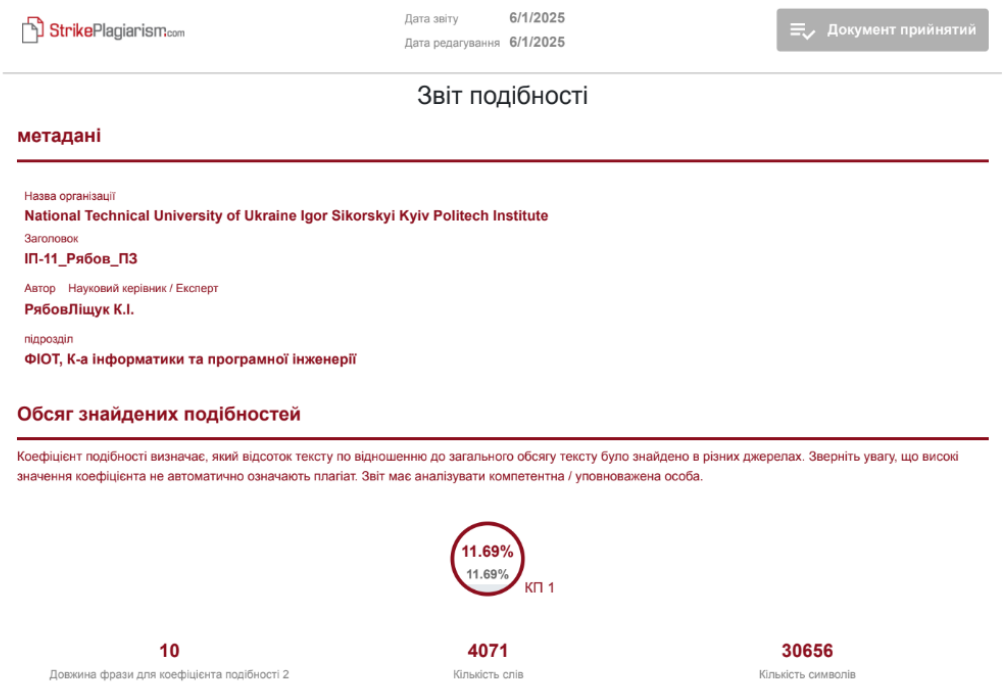
5) ASP.NET. URL: https://dotnet.microsoft.com/en-us/apps/aspnet (дата звернення: 28.05.2025)

6) Entity Framework. URL: https://learn.microsoft.com/en-us/aspnet/entity-framework (дата звернення: 28.05.2025)

7) SendGrid. URL: https://sendgrid.com/en-us (дата звернення: 28.05.2025)

8) JetBrains Qodana. URL: https://www.jetbrains.com/qodana/ (дата звернення: 28.05.2025)

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



StrikePlagiarism.com

Дата звіту 6/1/2025
Дата редагування 6/1/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-11_Рябов_ПЗ
Автор Науковий керівник / Експерт
РябовЛіщук К.І.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

11.69%
11.69%
КП 1

10	4071	30656
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2025 р.

Вебзастосунок для автоматичного підбору вакансій на основі резюме та адаптації резюме за допомогою нейромереж для IT-галузі (комплексна тема).

Вебзастосунок та агрегація вакансій

Текст програми

КПІ.ІП-1122.045440.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Юрій РЯБОВ

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду
https://github.com/YuraRiabov/KolybaResume

Файл DouVacancyAggregatorService.cs
Реалізація функціональної задачі збору вакансій з сайту dou.ua

```
using System.Net;
using System.Net.Http.Json;
using AutoMapper;
using KolybaResume.BLL.Models;
using KolybaResume.BLL.Services.Abstract;
using KolybaResume.BLL.Services.Base;
using KolybaResume.BLL.Services.Utility;
using KolybaResume.Common.Enums;
using KolybaResume.DAL.Context;
using KolybaResume.DAL.Entities;
using Microsoft.EntityFrameworkCore;

namespace KolybaResume.BLL.Services.Aggregators;

public class DouVacancyAggregatorService(KolybaResumeContext context,
IMapper mapper) : BaseService(context, mapper), IAggregator
{
   public async Task<List<Vacancy>> Aggregate()
   {
      var isFirstRun = !_context.Vacancies.Any(v => v.Source ==
VacancySource.Dou);
      var companyLinks = await _context.Companies.Select(c =>
c.Url).Take(1500).ToListAsync();
      var addedVacancies = new List<Vacancy>();
      var allVacanciesIds = new List<int>();

      try
      {
         foreach (var link in companyLinks)
         {
            var vacancies = await GetVacancies($"{link}vacancies/export/",
isFirstRun, allVacanciesIds);
            await _context.Vacancies.AddRangeAsync(vacancies);
            await _context.SaveChangesAsync();
            addedVacancies.AddRange(vacancies);
         }
      }
      finally
```

```
        {
            var vacanciesToDelete = (await _context.Vacancies.ToListAsync())
                .Where(v => v.Source == VacancySource.Dou &&
!allVacanciesIds.Contains(DouVacancyIdExtractor.GetId(v.Url)));

            _context.Vacancies.RemoveRange(vacanciesToDelete);
            await _context.SaveChangesAsync();
        }

        return addedVacancies;
    }

    private async Task<Vacancy[]> GetVacancies(string url, bool isFirstRun,
List<int> allVacanciesIds)
    {
        var handler = new HttpClientHandler
        {
            UseCookies = true,
            CookieContainer = new CookieContainer(),
            AutomaticDecompression = DecompressionMethods.GZip |
DecompressionMethods.Deflate
        };

        using var client = new HttpClient(handler);

        client.DefaultRequestHeaders.UserAgent.ParseAdd(
            "Mozilla/5.0 (Windows NT 10.0; Win64; x64) " +
            "AppleWebKit/537.36 (KHTML, like Gecko) " +
            "Chrome/114.0.0.0 Safari/537.36"
        );

        client.DefaultRequestHeaders.Accept.ParseAdd("application/json,
text/javascript, */*; q=0.01");
        client.DefaultRequestHeaders.TryAddWithoutValidation("X-Requested-
With", "XMLHttpRequest");

        using var request = new HttpRequestMessage(HttpMethod.Get, url);

        var response = await client.SendAsync(request);
        response.EnsureSuccessStatusCode();

        var vacancies = await
response.Content.ReadFromJsonAsync<VacancyModel[]>();
        allVacanciesIds.AddRange(vacancies?.Select(v =>
```

```
DouVacancyIdExtractor.GetId(v.Link)) ?? []);
        return _mapper.Map<Vacancy[]>(vacancies?.Where(v => isFirstRun || v.Date
> DateTime.Today.AddDays(-1)));
    }
}
```

Файл PostJobVacancyAggregatorService.cs
Реалізація функціональної задачі збору вакансій з сайту postjobfree.com

```
using AutoMapper;
using KolybaResume.BLL.Services.Abstract;
using KolybaResume.BLL.Services.Base;
using KolybaResume.BLL.Services.Scrappers;
using KolybaResume.Common.Enums;
using KolybaResume.DAL.Context;
using KolybaResume.DAL.Entities;
using Microsoft.EntityFrameworkCore;

namespace KolybaResume.BLL.Services.Aggregators;

public class PostJobVacancyAggregatorService(KolybaResumeContext context,
IMapper mapper) : BaseService(context, mapper), IAggregator
{
    private static readonly string[] CategoryQueries = [
        "\"mobile developer\"",
        "\"customer support\"",
        "\"project manager\"",
        "devops",
        "\"data analyst\"",
        "qa",
        "\"sales manager\"",
        "\"ux designer\"",
    ];

    public async Task<List<Vacancy>> Aggregate()
    {
        var isFirstRun = !_context.Vacancies.Any(v => v.Source ==
VacancySource.PostJob);
        var addedVacancies = new List<Vacancy>();
        var allVacanciesLinks = new List<string>();

        try
        {
            foreach (var query in CategoryQueries)
            {
```

```
            var vacancies = PostJobFreeVacanciesScraper.Scrape(query);
            allVacanciesLinks.AddRange(vacancies.Select(v => v.Link));

            var vacanciesToAdd = _mapper.Map<Vacancy[]>(vacancies.Where(v
=> isFirstRun || v.Date > DateTime.Today.AddDays(-1)));
            await _context.Vacancies.AddRangeAsync(vacanciesToAdd);
            await _context.SaveChangesAsync();
            addedVacancies.AddRange(vacanciesToAdd);
        }
    }
    finally
    {
        var vacanciesToDelete = (await _context.Vacancies.ToListAsync())
            .Where(v => v.Source == VacancySource.PostJob &&
                    !allVacanciesLinks.Contains(v.Url));

        _context.Vacancies.RemoveRange(vacanciesToDelete);
        await _context.SaveChangesAsync();
    }

    return addedVacancies;
  }
}
```

Файл PostJobFreeVacanciesScrapper.cs
Реалізація функціональної задачі скрапінгу вакансій з сайту postjobfree.com за фразою пошуку

```
using System.Globalization;
using KolybaResume.BLL.Models;
using KolybaResume.Common.Enums;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace KolybaResume.BLL.Services.Scrappers;

public static class PostJobFreeVacanciesScraper
{
  private const string BaseUrl = "https://www.postjobfree.com/jobs";

  public static List<VacancyModel> Scrape(string query)
  {
    var vacanciesList = new List<VacancyModel>();
    var options = new ChromeOptions();
```

```
        options.AddArgument("--headless");
        var driver = new ChromeDriver(options);
        var wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));

        try
        {
            var firstPageUrl =
$"{BaseUrl}?q={Uri.EscapeDataString(query)}&r=100&p=1";
            driver.Navigate().GoToUrl(firstPageUrl);
            wait.Until(drv => drv.FindElements(By.CssSelector(".pager")).Count !=
0);

            var pagerLinksCount = driver
                .FindElements(By.CssSelector(".pager"))
                .Where(a =>
                {
                    var paginationText = a.Text.Trim();
                    return !string.Equals(paginationText, "Previous",
StringComparison.OrdinalIgnoreCase)
                        && !string.Equals(paginationText, "Next",
StringComparison.OrdinalIgnoreCase);
                })
                .Count();

            int totalPages = Math.Min(pagerLinksCount, 5);

            for (int page = 1; page <= totalPages; page++)
            {
                var listUrl =
$"{BaseUrl}?q={Uri.EscapeDataString(query)}&r=100&p={page}";
                driver.Navigate().GoToUrl(listUrl);

                wait.Until(drv => drv.FindElements(By.CssSelector("h3 > a")).Count !=
0);

                var jobLinks = driver
                    .FindElements(By.CssSelector("h3 > a"))
                    .Select(elem => elem.GetAttribute("href")!)
                    .Where(link => !string.IsNullOrEmpty(link) &&
link.Contains("postjobfree.com/job"))
                    .ToList();

                if (jobLinks.Count == 0)
                {
                    break;
```

```
            }

            foreach (var link in jobLinks)
            {
               try
               {
                  var vacancy = new VacancyModel
                  {
                     Link = link,
                     Source = VacancySource.PostJob,
                     Category = query
                  };

                  driver.Navigate().GoToUrl(link);
                  wait.Until(drv => drv.FindElement(By.TagName("h1")));

                  vacancy.Title = driver.FindElement(By.TagName("h1")).Text;
                  vacancy.Location =
SafeFindText(By.CssSelector(".colorLocation"), driver);
                  vacancy.Salary = SafeFindText(By.CssSelector(".colorSalary"),
driver);
                  vacancy.Date = DateTime.ParseExact(
                     SafeFindText(By.CssSelector(".colorDate"), driver),
                     "MMMM d, yyyy",
                     CultureInfo.InvariantCulture
                  );
                  vacancy.Description =
driver.FindElement(By.CssSelector(".normalText")).Text.Trim();

                  vacanciesList.Add(vacancy);
               }
               catch (Exception)
               {
                  Console.WriteLine($"Unable to get vacancy, link: {link}");
               }
            }
         }
      }
      finally
      {
         driver.Quit();
      }

      return vacanciesList;
   }
```

```
    private static string SafeFindText(By by, ChromeDriver driver)
    {
        try
        {
            return driver.FindElement(by).Text.Trim();
        }
        catch (NoSuchElementException)
        {
            return string.Empty;
        }
    }
}
```

Файл PostJobFreeVacancyScrapper.cs
Реалізація функціональної задачі скрапінгу тексту вакансій за посиланням з postjobfree.com

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace KolybaResume.BLL.Services.Scrappers;

public class PostJobFreeVacancyScrapper
{
    public async Task<string> Scrape(string url)
    {
        var options = new ChromeOptions();
        options.AddArgument("--headless");
        using var driver = new ChromeDriver(options);

        try
        {
            await driver.Navigate().GoToUrlAsync(url);

            var wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
            var description = wait.Until(d =>
d.FindElement(By.CssSelector(".normalText")));

            return description.Text.Trim();
        }
        finally
        {
            driver.Quit();
```

```
        }
    }
}
```

Файл DouCompanyScrapper.cs

Реалізація функціональної задачі збору посилань на компанії з сайту dou.ua

```
using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using SeleniumExtras.WaitHelpers;

namespace KolybaResume.BLL.Services.Scrappers;

public static class DouCompanyScrapper
{
    public static string[] Scrape()
    {
        var options = new ChromeOptions();
        options.AddArgument("--headless");
        using var driver = new ChromeDriver(options);

        driver.Navigate().GoToUrl("https://jobs.dou.ua/companies/");

        var wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));

        while (true)
        {
            try
            {
                var moreButton =
wait.Until(ExpectedConditions.ElementToBeClickable(
                    By.LinkText("Більше компаній")
                ));

                var count = driver.FindElements(By.CssSelector("a.cn-a")).Count;

                if (count > 1500)
                {
                    break;
                }

                moreButton.Click();

                wait.Until(drv =>
```

```
                drv.FindElements(By.CssSelector("a.cn-a")).Count > count
            );
        }
        catch (WebDriverTimeoutException)
        {
            break;
        }
        catch (NoSuchElementException)
        {
            break;
        }
    }

    return driver.FindElements(By.CssSelector("a.cn-a"))
        .Select(e => e.GetAttribute("href"))
        .Distinct()
        .ToArray()!;
  }
}
```

Файл TextExtractorService.cs
Реалізація функціональної задачі зчитування тексту резюме

```
using System.Text;
using DocumentFormat.OpenXml.Packaging;
using NPOI.HWPF;
using NPOI.HWPF.Extractor;
using UglyToad.PdfPig;

namespace KolybaResume.BLL.Services;

public static class TextExtractorService
{
  public static string ReadPdf(Stream stream)
  {
    var sb = new StringBuilder();
    using (var document = PdfDocument.Open(stream))
    {
      foreach (var page in document.GetPages())
      {
        sb.AppendLine(page.Text);
      }
    }
```

```
        return sb.ToString();
    }

    public static string ReadDocx(Stream stream)
    {
        using var document = WordprocessingDocument.Open(stream, false);
        return document.MainDocumentPart!.Document.Body!.InnerText;
    }

    public static string ReadDoc(Stream stream)
    {
        var document = new HWPFDocument(stream);
        var extractor = new WordExtractor(document);
        return extractor.Text;
    }
}
```

Файл UserService.cs
Реалізація функціональної задачі управління користувачами і їх резюме

```
using AutoMapper;
using FirebaseAdmin.Auth;
using KolybaResume.BLL.Extensions;
using KolybaResume.BLL.Services.Abstract;
using KolybaResume.BLL.Services.Base;
using KolybaResume.Common.DTO.User;
using KolybaResume.DAL.Context;
using KolybaResume.DAL.Entities;
using Microsoft.AspNetCore.Http;
using Microsoft.EntityFrameworkCore;

namespace KolybaResume.BLL.Services;

public class UserService(KolybaResumeContext context, IMapper mapper,
FirebaseAuth firebaseAuth, IHttpContextAccessor httpContextAccessor,
IMachineLearningApiService apiService) : BaseService(context, mapper),
IUserService
{
    public async Task<UserDto> GetCurrent()
    {
        var currentUser = await GetCurrentInternal();

        await AddClaims(currentUser.Uid, currentUser.Id);
```

```
        var currentUserDto = _mapper.Map<UserDto>(currentUser);
        return currentUserDto;
    }

    public async Task<bool> CheckExisting(string email)
    {
        return await _context.Users.AnyAsync(u => u.Email == email);
    }


    public async Task<UserDto> Create(NewUserDto userDto)
    {
        if (userDto is null)
        {
            throw new ArgumentNullException(nameof(userDto), "New user cannot be
null");
        }

        var userEntity = await _context.Users.FirstOrDefaultAsync(u =>
u.Email.Equals(userDto.Email));
        if (userEntity is not null)
        {
            return _mapper.Map<UserDto>(userEntity);
        }

        var newUser = _mapper.Map<NewUserDto, User>(userDto);
        var user = (await _context.Users.AddAsync(newUser)).Entity;
        await _context.SaveChangesAsync();

        await AddClaims(user.Uid, user.Id);

        return _mapper.Map<User, UserDto>(user);
    }

    public async Task AddResume(string text)
    {
        var userId = (await GetCurrentInternal()).Id;
        var existingResume = await _context.Resumes.FirstOrDefaultAsync(r =>
r.UserId == userId);

        if (existingResume is not null)
        {
            _context.Resumes.Remove(existingResume);
        }
```

```
    var resume = new Resume
    {
        Text = text,
        UserId = userId
    };

    await _context.Resumes.AddAsync(resume);
    await _context.SaveChangesAsync();

    if (await apiService.NotifyResumeCreated(resume.Id))
    {
        return;
    }

    _context.Resumes.Remove(resume);
    await _context.SaveChangesAsync();

    throw new Exception("Could not add resume");
}

public async Task<long> GetResumeId()
{
    var user = await GetCurrentInternal();
    return user.Resume?.Id ?? 0;
}

private async Task AddClaims(string? uid, long? id)
{
    if (uid is null || id is null)
    {
        return;
    }

    var userRecord = await firebaseAuth.GetUserAsync(uid);

    if (userRecord.CustomClaims.ContainsKey("id"))
    {
        return;
    }

    var userClaims = new Dictionary<string, object>
    {
        { "id", id }
    };
```

```
        await firebaseAuth.SetCustomUserClaimsAsync(uid, userClaims);
    }

    private async Task<User> GetCurrentInternal()
        => await _context.Users.Include(u => u.Resume).FirstOrDefaultAsync(u =>
u.Uid == GetCurrentId())
            ?? throw new KeyNotFoundException("User doesn't exist");


    private string? GetCurrentId()
    {
        var userId = httpContextAccessor.HttpContext.User.GetUid();
        return userId;
    }
}
```

Файл VacancyService.cs
Реалізація функціональної задачі отримання вакансій та адаптації резюме

```
using AutoMapper;
using KolybaResume.BLL.Models;
using KolybaResume.BLL.Services.Abstract;
using KolybaResume.BLL.Services.Base;
using KolybaResume.BLL.Services.Scrappers;
using KolybaResume.BLL.Services.Utility;
using KolybaResume.Common.DTO.Vacancy;
using KolybaResume.Common.Enums;
using KolybaResume.DAL.Context;
using Microsoft.EntityFrameworkCore;

namespace KolybaResume.BLL.Services;

public class VacancyService(
    KolybaResumeContext context,
    IMapper mapper,
    IMachineLearningApiService apiService,
    IUserService userService) : BaseService(context, mapper), IVacancyService
{
    public async Task<VacancyTextDto> ParseVacancy(string vacancyUrl)
    {
        if (vacancyUrl.Contains("jobs.dou.ua"))
        {
            var vacancies = await _context.Vacancies.Where(v => v.Source ==
```

```
VacancySource.Dou).ToListAsync();
            var vacancy = vacancies.FirstOrDefault(v =>
DouVacancyIdExtractor.Compare(v.Url, vacancyUrl));

            if (vacancy != null)
            {
                return new VacancyTextDto
                {
                    Text = vacancy.CleanedText,
                };
            }
        }

        if (vacancyUrl.Contains("www.postjobfree.com/job"))
        {
            return new VacancyTextDto
            {
                Text = await new PostJobFreeVacancyScrapper().Scrape(vacancyUrl)
            };
        }

        throw new ArgumentException("Invalid URL");
    }

    public async Task<VacancyDto[]> Get()
    {
        var resumeId = await userService.GetResumeId();

        var scores = await apiService.GetVacancyScores(resumeId);

        var vacancies =
            (await _context.Vacancies.ToListAsync()).Where(v => scores.Any(score
=> score.VacancyId == v.Id));
        var dtos = _mapper.Map<VacancyDto[]>(vacancies);

        foreach (var dto in dtos)
        {
            dto.Score = scores.First(score => score.VacancyId == dto.Id).Score;
        }

        return dtos.OrderByDescending(d => d.Score).ToArray();
    }

    public async Task<AdaptationResponseDto> AdaptResume(string vacancyText,
bool shouldClean = true)
```

```
    {
        var resumeId = await userService.GetResumeId();

        return _mapper.Map<AdaptationResponseDto>(await
apiService.GetResumeAdaptation(new ResumeAdaptationRequest
        {
            ResumeId = resumeId,
            VacancyText = vacancyText,
            Clean = shouldClean
        }));
    }

    public async Task<AdaptationResponseDto> AdaptResume(long vacancyId)
    {
        var vacancyText = (await _context.Vacancies.FirstOrDefaultAsync(v => v.Id
== vacancyId))?.CleanedText;

        if (vacancyText == null)
        {
            throw new ArgumentException("Vacancy not found");
        }

        return await AdaptResume(vacancyText, false);
    }
}
```

Файл AggregatorJob.cs
Реалізація функціональної задачі збору вакансій та надсилання імейлів

```
using KolybaResume.BLL.Models;
using KolybaResume.BLL.Services.Abstract;
using KolybaResume.DAL.Context;
using KolybaResume.DAL.Entities;
using Microsoft.EntityFrameworkCore;
using Quartz;

namespace KolybaResume.Jobs;

public class AggregatorJob(IEnumerable<IAggregator> aggregators,
KolybaResumeContext dbContext, IEmailService emailService,
IMachineLearningApiService apiService) : IJob
{
```

```
    public async Task Execute(IJobExecutionContext context)
    {
        var addedVacancies = new List<Vacancy>();

        foreach (var aggregator in aggregators)
        {
            addedVacancies.AddRange(await aggregator.Aggregate());
        }

        var scores = new List<VacancyScoreResponse>();

        foreach (var batch in addedVacancies.Chunk(96))
        {
            scores.AddRange(await apiService.NotifyVacanciesUpdated(batch.Select(v
=> v.Id).ToArray()));
        }

        foreach (var userScore in scores.GroupBy(s => s.UserId))
        {
            var user = await dbContext.Users.FirstOrDefaultAsync(u => u.Id ==
userScore.Key);

            var relevantVacancies = userScore
                .Where(us => us.Score > 60)
                .Select(us => addedVacancies.First(v => v.Id ==
us.VacancyId)).ToArray();

            if (relevantVacancies.Length != 0 && user != null)
            {
                await emailService.SendAsync(
                    user.Email,
                    user.Name,
                    "New relevant vacancies",
                    string.Join(Environment.NewLine, relevantVacancies.Select(v =>
$"{v.Title}: {v.Url}")));
            }
        }
    }
}
```

Файл ScrapperJob.cs

Реалізація функціональної задачі збору посилань на компанії при першому запуску серверу

```
using KolybaResume.BLL.Services.Abstract;
using KolybaResume.BLL.Services.Scrappers;

namespace KolybaResume.Jobs;

public class ScrapperJob(IServiceProvider services) : IHostedService
{
    public async Task StartAsync(CancellationToken cancellationToken)
    {
        try
        {
            using var scope = services.CreateScope();
            var companyService =
scope.ServiceProvider.GetRequiredService<ICompanyService>();

            if (await companyService.HasCompanies())
            {
                return;
            }

            var companyLinks = DouCompanyScrapper.Scrape();
            await companyService.Create(companyLinks);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }
    }

    public Task StopAsync(CancellationToken cancellationToken) =>
Task.CompletedTask;
}
```

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2025 р.

Вебзастосунок для автоматичного підбору вакансій на основі резюме та адаптації резюме за допомогою нейромереж для IT-галузі (комплексна тема).

Вебзастосунок та агрегація вакансій

Програма та методика тестування

КПІ.ІП-1122.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Юрій РЯБОВ

Київ – 2025

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є тестування веб-застосунку Kolyba Resume, а саме коректності роботи авторизації, завантаження резюме користувача, пошуку та фільтрації вакансій, та адаптації резюме до вакансії з пошуку та за текстом вакансії.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
– знаходження проблем, помилок і недоліків з метою їх усунення;

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Під час проведення тестування будуть використовуватись допоміжний засіб Qodana для статичного аналізу коду.

Порядок проведення тестування буде наступним:

– тестування інтерфейсу користувача;
– статичний аналіз коду
– тестування на виведення повідомлень про помилку, коли це необхідно;
– функціональне тестування на відповідність функціональним вимогам;

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2025 р.

Вебзастосунок для автоматичного підбору вакансій на основі резюме та адаптації резюме за допомогою нейромереж для ІТ-галузі (комплексна тема). Вебзастосунок та агрегація вакансій

Графічний матеріал

КПІ.ІП-1122.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

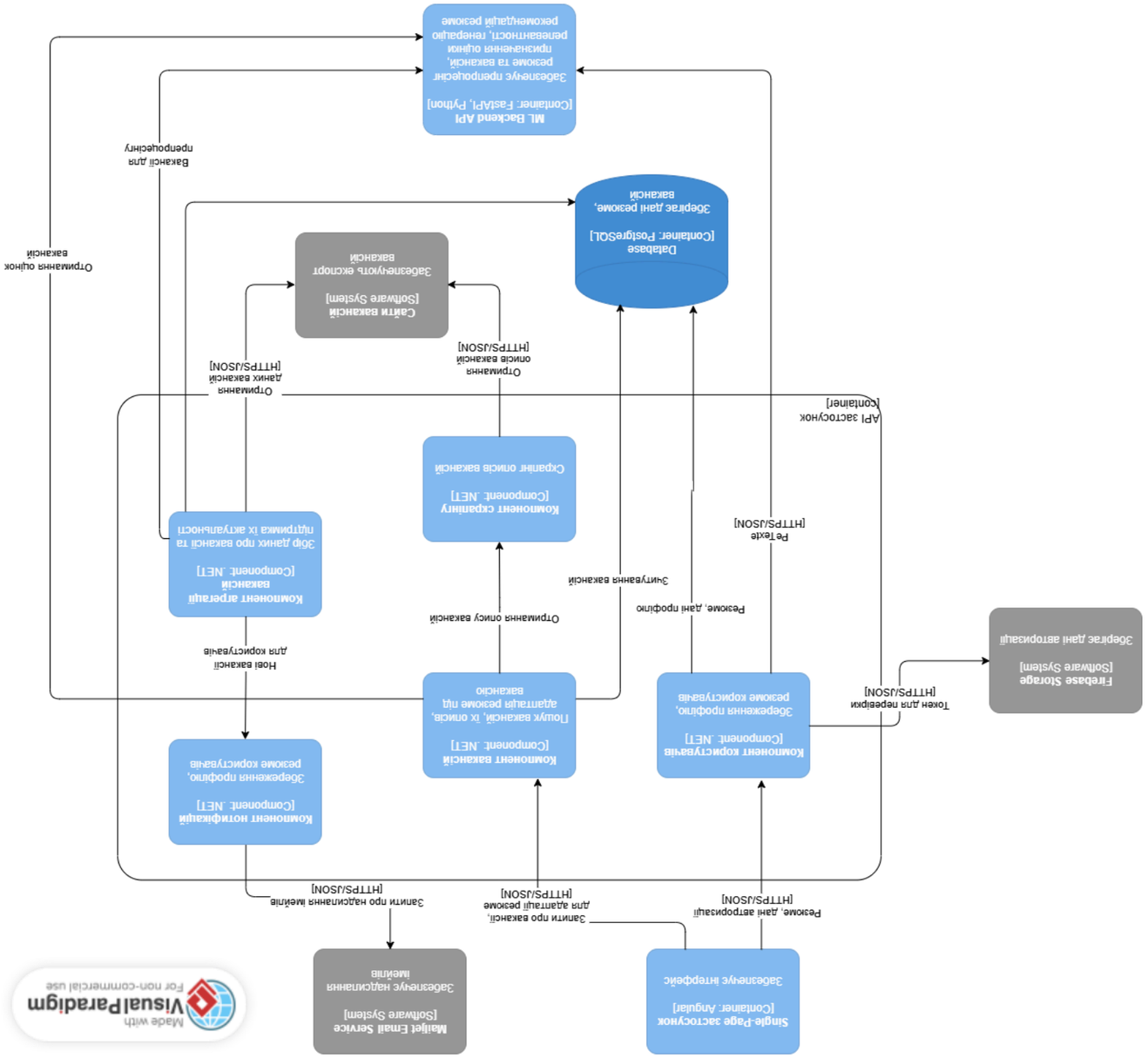
Нормоконтроль:

_____ Катерина ЛІЩУК

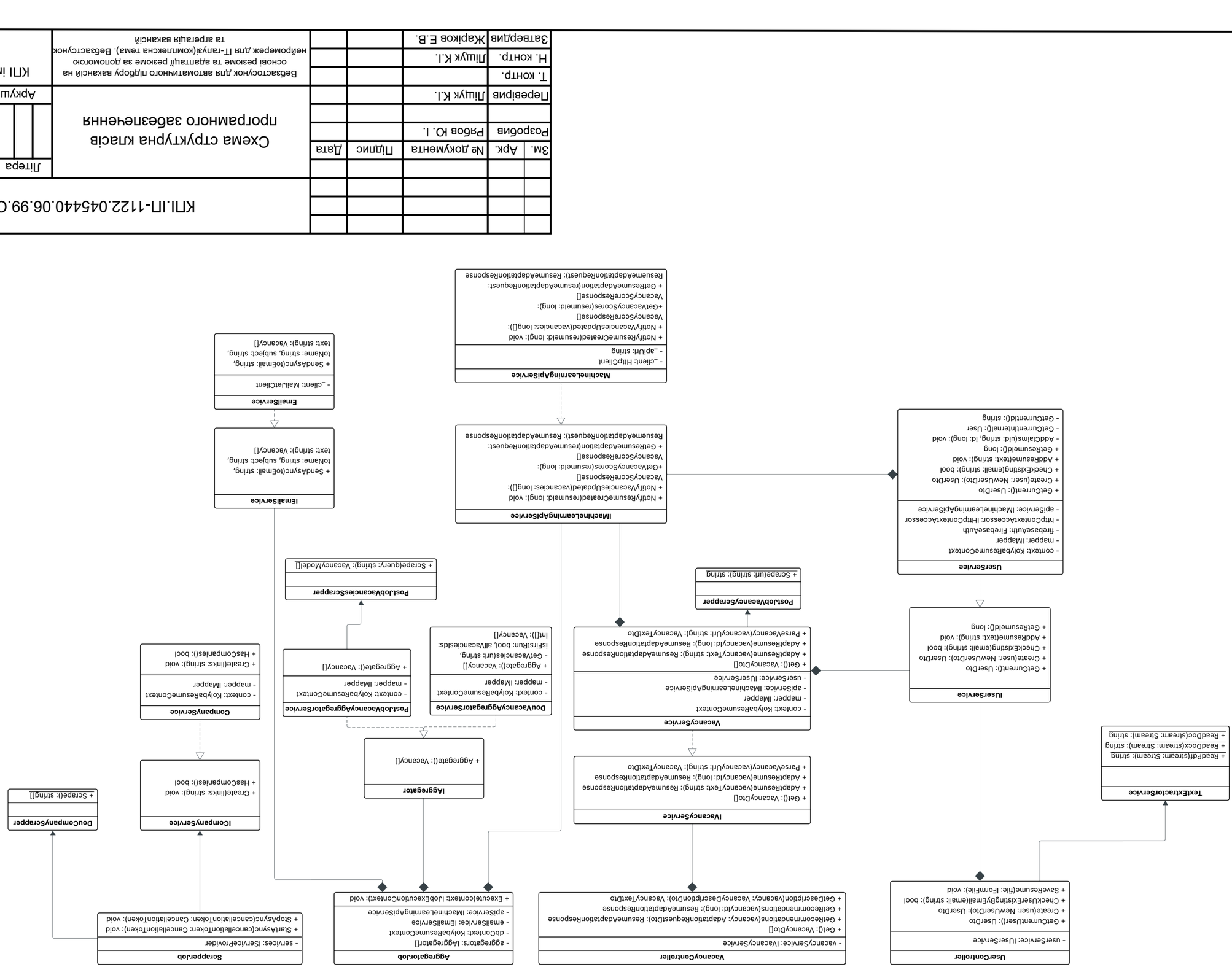
Виконавець:

_____ Юрій РЯБОВ

Київ – 2025



									КПІ.ІП-1122.045440.06.99.ССМ	
3м.	Арк.	№ документа	Підпис	Дата	Схема структура компонентів програмного забезпечення			Літера	Маса	Масштаб
Розробив		Рябов Ю. І.								
Перевірив		Лішук К.І.						Аркуш 1		Аркушів 2
Т. контр.					Безбастучок для автоматичного підбору вакансій на основі резюме та адаптацій резюме за допомогою нейромережі для IT-галузі (комплексна тема). Безбастучок та агрегація вакансій			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11		
Н. контр.		Лішук К.І.								
Затвердив		Жарікова Е.В.								



КПІ.ІП-1122.045440.06.99.ССК

					КПІ.ІП-1122.045440.06.99.ССК
Зм.	Арк.	№ докумeнта	Підпис	Дата	Схема структура класів програмного забезпечення
Розробив		Рябов Ю. І.			
Перевірив		Лящук К.І.			
Т. контр.					Арк. 2 / Аркушів 2
Н. контр.		Лящук К.І.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11
Затвердив		Жаріков Е.В.			Веб-застосунок для автоматичного підбору вакансій на основі резюме та додаткової інформації від користувача. Веб-застосунок інтегрується для IT-галузі(комп'ютерна тема). Веб-застосунок перевіряє дані та агрегує вакансії
Літера	Маса	Масштаб			