Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова
Херсонський навчально-науковий інститут

Кафедра інформаційних технологій та фізико-математичних дисциплін

«Допущений до захисту»
Завідувач кафедри

д.т.н., доцент Гучек П.Й.
«22» червня 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи на здобуття ступеня вищої освіти «бакалавр»

на тему: “ Розробка програмного забезпечення веб-додатку для ранжування резюме в рекрутингу» ”

Виконав: студент III курсу, групи 3157ст3
спеціальності
121 - Інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми
Інженерія програмного забезпечення
(назва освітньої програми)

Желобатий А.В.
(прізвище та ініціали)

Керівник Гайдаєнко О.В.
(прізвище та ініціали)

Рецензент Партас В.К.
(прізвище та ініціали)

м. Миколаїв-Херсон - 2024 р.

Форма № Н-9.02

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
імені адмірала Макарова
ХЕРСОНСЬКИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

Кафедра інформаційних технологій та фізико-математичних дисциплін
Галузь знань 12 "Інформаційні технології"
(шифр і назва)
Спеціальність 121 "Інженерія програмного забезпечення"
(шифр і назва)
Освітня програма "Інженерія програмного забезпечення"
(назва)

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ Літвінова М.Б.
" 17 " березня 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студентки Желобатий Анатолій Валентинович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи " Розробка програмного забезпечення веб-додатку для" ранжування резюме в рекрутингу»

Керівник роботи Гайдаєнко Оксана Володимирівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені розпорядженням по інституту від "04" березня 2024 року № 01

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

4.1 Титульний аркуш, ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ (ДИПЛОМНУ) РОБОТУ, АНОТАЦІЯ (українською, англійською), ЗМІСТ, ПЕРЕЛІК УМОВНИХ ОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ (при необхідності).

4.2 ВСТУП

4.3 ОПИС ПРЕДМЕТНОЇ ГАЛУЗІ, ПОСТАНОВКА ЗАДАЧІ

4.4 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (ПЗ)(зовнішні специфікації ПЗ, стани програми при виконанні, структура програми, структура даних, потоки даних та управління, проект ПЗ на рівні програмних модулів, випробування ПЗ)

4.5 РЕЗУЛЬТАТИ РОЗРОБКИ

4.6 РОЗДІЛ З ОХОРОНИ ПРАЦІ (ОХОРОНИ ОТОЧУЮЧОГО СЕРЕДОВИЩА) _____

4.7 ВИСНОВКИ

4.8 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

4.9 ДОДАТКИ (технічне завдання, опис програми, текст програми, програма та методика випробувань ПЗ, інструкція користувача)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

5.1 Діаграма варіантів використання веб-додатку для ранжування резюме в рекрутингу.

5.2 Діаграма компонентів і розгортання веб-додатку для ранжування резюме в рекрутингу.

5.3 Діаграма активності сеансу користувача на веб-додатку для ранжування резюме в рекрутингу.

5.4

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4.1-4.3	Дудченко О.М., декан		
4.4-4.5	Латанська Л.О., доцент		
4.6	Щедролосєв О.В., професор		

7. Дата видачі завдання 25 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Опис та аналіз предметної галузі	28.03.2024 р.	
2.	Постановка та формалізація задачі	06.04.2024 р.	
3.	Ескізний проєкт програмного забезпечення	13.04.2024 р.	
4.	Технічний проєкт програмного забезпечення	20.04.2024 р.	
5.	Робочий проєкт програмного забезпечення	27.04.2024 р.	
6.	Випробування програмного забезпечення	04.05.2024 р.	
7.	Розробка робочої документації	11.05.2024 р.	
8.	Виконання графічних документів	13.05.2024 р.	
9.	Вирішення питань охорони праці	18.05.2024 р.	
10.	Оформлення пояснювальної записки	25.05.2024 р.	
11.	Подання кваліфікаційної роботи на попередній захист	03.06.2024 р.	
12.	Подання кваліфікаційної роботи рецензенту	06.06.2024 р	
13.	Подання на кафедру ІТ та ФМД тексту остаточного варіанту роботи, підписаного її керівником, разом з заявою щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи	08.06.2024 р.	
14.	Подання на кафедру ІТ та ФМД електронних версії наступних документів у форматі pdf: кваліфікаційної (дипломної) роботи; файлу-опису кваліфікаційної роботи (згідно Застосунку до наказу ректора НУК від 19.05.2020 р. за №287-уч); презентації доповіді; письмового відгуку та рецензії на кваліфікаційну роботу	14.06.2024 р.	
15	Подання на кафедру ІТ ФМД письмового відгуку та рецензії на кваліфікаційну роботу	22.06.2024 р.	

Студент _____ Желобатий А.В.
Підпис Прізвище та ініціали

Керівник роботи _____ Гайдаєнко О.В
Підпис Прізвище та ініціали

Анотація

Дана дипломна робота присвячена розробці програмного забезпечення за темою дипломного проектування «Розробка програмного забезпечення веб-додатку для ранжування резюме в рекрутингу». Метою розробки є створення веб-додатку, що забезпечить рекрутерам можливість ранжування резюме, та буде зручною та зрозумілою у використанні.

Дипломна робота містить стадії та етапи розробки програмного забезпечення, опис призначення, структуру, функції, основні компоненти, задачі та цілі розроблюваного ПЗ. У роботі представлено аналіз процесу підбору кадрів, розділи з охорони праці, міститься технічне завдання на розробку програмного забезпечення, текст та опис програми, інструкція користувача, тестування та методика випробувань. Для розробки програмного забезпечення була обрана об’єктно-орієнтована методологія.

Дипломна робота викладена на 118 сторінках друкованого тексту та містить 23 рисунка, 25 таблиць, 3 додатків і список використаних джерел з 9 найменувань.

Дипломна робота викладена українською мовою.

Ключові слова: веб-додаток, програмне забезпечення, ранжування резюме, рекрутинг, автоматизація підбору персоналу, обробка даних, оцінка кандидатів, HR-технології, алгоритми сортування, штучний інтелект.

Abstract

This diploma paper is devoted is dedicated to the analysis of the subject area and the formulation of the problem relating to diploma projects "Development of the web application for ranking of resumes at recruiting". The goal of the development is to create a web application that will provide recruiters with the opportunity to rank the resume, and it will be convenient and understandable to use.

The diploma paper contains stages of software development, description of the purpose, structure, functions, main components, tasks and goals of the software being developed. The paper presents an analysis of the process of recruitment, sections on labor protection, contains a technical specification, the text and description of the program, user's instruction, testing and test methodology. For the development of software, an object-oriented methodology was chosen.

The diploma paper is presented on 118 pages of printed text and contains 23 figures, 25 tables, 3 appendixes and a list of references with 9 titles.

Diploma paper is written in Ukrainian.

Keywords: web application, software, resume ranking, recruitment, recruitment automation, data processing, candidate evaluation, HR technologies, sorting algorithms, artificial intelligence.

Зміст

ВСТУП

Підбор персоналу (рекрутинг) завжди був важливим і необхідним процесом для будь-якої компанії. Адже роботу, необхідну компанії, виконує її персонал і якість роботи напряму залежить від підібраних кадрів. Особливо цей аспект відчувається зараз з розвитком технологій, який спонукав і спонукає створення нових технологічних компаній. Створення продуктів яких, потребує високої кваліфікації від працівників і, нерідко, вимагає повної відповідності всім вимогам, навіть самим специфічним. Але розвиток технологій з шаленою швидкістю також створює проблему з підбором кадрів, яка полягає у відсутності необхідних знань у кандидатів. Саме ці проблеми мають вирішувати рекрутери (HR department).

Підбор персоналу - це система цілеспрямованих дій щодо залучення до роботи кандидатів, які мають якості, необхідні для досягнення цілей, поставлених організацією [1].

В зв’язку з вище названими причинами робота рекрутера ускладнюється і не завжди вдається підібрати необхідного кандидата і не сплутати кваліфікації. Рекрутер найчастіше приймає рішення щодо кваліфікації кандидата на основі його резюме. І в процесі вибору, відповідного до вимог кандидата, рекрутеру, нерідко, необхідно опрацювати сотні або, навіть, тисячі резюме. Невірний вибір має дорогий вплив на бізнес-процеси компанії. Так, процес підбору персоналу включає багато методик і підзавдань й одним з найперших – є аналіз представлених резюме, що полягає у первинному відборі претендентів, які відповідають профілю посади.

Існує низка програмних рішень (декілька з яких розглянуто в підрозділі 1.1), які спрямовані на підвищення ефективності аналізу представлених резюме і можуть бути або вже впроваджені компаніями, але, як правило, вони або не повністю задовольняють потребам компаній, або коштують немало грошей, зібрання яких вимагає багато клопоту, або навіть є неможливим. Та й більшість таких розробок – є суто корпоративними рішеннями, які не підійдуть широкому загалу.

Окрім цього, загально відомо, що гігантські корпоративні компанії та кадрові агентства контролюють процес та керують тисячами резюме, які будуть автоматично оброблені системою вилучення інформації. Витягнута інформація може зберігатися як структуровані дані в базі даних, які можуть бути використані в різних сферах. На відміну від багатьох неструктурованих типів документів, інформація в резюме має

трохи структуровану форму, де інформація зберігається в окремих блоках, що й дозволить автоматизувати цей процес.

Автоматизація процесу ранжування резюме в рекрутингу має базуватися на впровадженні певних математичних методів, зокрема впровадження методів машинного навчання та аналізу текстів в різноманітні сфери. Реалізація яких дозволяє програмним системам не тільки оброблювати великі об’єми інформації, а ще й в процесі підвищувати ефективність та якість обробки наступної інформації. Головною метою даних впроваджень – є зменшення витрат людино-годин.

Метою роботи є підвищення ефективності роботи рекрутерів компаній та зменшення ймовірності виникнення помилок в роботі через вплив людського фактору, скорочення часу на обробку даних о кандидатах на вакансії, шляхом розоробки та впровадження програмного забезпечення для ранжування резюме в рекрутингу. Програмне забезпечення повинне мати інтуїтивно-зрозумілий інтерфейс та не вимагати додаткового навчання персоналу фірми, який буде працювати з ним.

Для досягнення поставленої мети в ході дипломної роботи необхідно вирішити наступні задачі:

- на підставі аналізу сучасних програмних продуктів для автоматизації рекрутингової діяльності та особливостей процесу підбору персоналу виконати постановку завдання на розробку відповідного програмного забезпечення;
- сформулювати вимоги, що пред’являються до розробки програмного забезпечення і оформити їх у вигляді документа «Технічне завдання»;
- розглянути основні питання охорони праці та запропонувати заходи, щодо забезпечення сприятливих умов праці при роботі з персональним комп’ютером;
- виконати всі етапи проектування програмного забезпечення (ескізний, технічний, робочий) та його реалізацію;
- провести випробування та тестування програмного забезпечення;
- оформити необхідну програмну документацію;
- розробити спеціальний розділ з охорони праці, проаналізувати шкідливі чинники та вплив небезпечних факторів на робочому місці користувача ПК, розробити заходи техніки безпеки при роботі з комп’ютером.

РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ ПІДБОРУ ПЕРСОНАЛУ ТА ПОСТАНОВКА ЗАДАЧІ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБ-ДОДАТКУ ДЛЯ РАНЖУВАННЯ РЕЗЮМЕ В РЕКРУТИНГУ

На стадії аналізу ми приділяємо увагу опису предметної галузі функціонування продукту. Знання того як відбувається процес підбору кадрів, та аналіз організації введення та зберігання даних, сприятимуть чіткому розумінню суті існуючих проблем та правильному визначенні завдання.

1.1 Опис та аналіз процесу підбору персоналу

Рекрутинг – підбір персоналу в штат компанії, або під замовлення клієнта у випадку рекрутингового агентства; основна функція і обов'язок менеджерів по персоналу та рекрутерів.

Рекрутинг – обов'язковий та необхідний процес у кожній організації будь-якої сфери діяльності: від якості та ефективності даного процесу напряму залежить її фінансова результативність.

Пошук персоналу, або рекрутинг, здійснюють використовуючи різні підходи та джерела інформації, основними джерелами є такі [2]:

- внутрішня база даних компанії або агенції;
- вищі навчальні заклади, тобто залучення молодих спеціалістів з вищих навчальних закладів;
- сайти по пошуку роботи;
- соціальний капітал (або пошук кандидатів по знайомству);
- сасоби масової інформації;
- соціальні мережі, форуми, блоги та ін.;
- працівники компаній-конкурентів, тобто переманювання спеціалістів;
- співпраця з рекрутинговими агенціями.

Відбір претендентів на позицію проводиться відповідно до профілю посади в якому обов'язково враховуються біографічні характеристики (стать, вік, освіта).

У процесі підбору кадрів (рекрутинговим агентством або відділом персоналу) можна виділити такі етапи [3]:

1 визначення потреби компанії в персоналі, відкриття відповідних вакансій. У разі якщо пошуком займається рекрутингове агентство - заповнення заявки на підбір персоналу і узгодження її умов. Підписання договору на надання послуг.

2 аналіз представлених резюме. Первинний відбір претендентів, що відповідають профілю посади.

3 попередня співбесіда по телефону. На цьому етапі можна докладніше дізнатися про освіту, досвід роботи, скласти первинне уявлення про комунікативні навички. Відібрані кандидати запрошуються на особисту співбесіду.

4 оціночна інтерв'ю з претендентом. Інтерв'ю направлено на оцінку ключових компетенцій, зазначених в профілі посади. За його підсумками заповнюються анкети на кожного претендента.

5 тестування (психологічний або професійне) для визначення рівня розвитку професійних знань, навичок і особистісних особливостей.

6 перевірка рекомендацій. Найчастіше така перевірка допомагає з'ясувати важливу інформацію про претендента, виключити можливі зони ризику і потенційні проблеми.

7 передача клієнтові (зовнішньому чи внутрішньому) розгорнутого резюме претендентів, які пройшли попередні етапи відбору.

8 проведення замовником (зовнішнім або внутрішнім) співбесід з відібраними претендентами.

9 аналіз результатів. При необхідності - проведення другого туру.

10 прийняття рішення про «закриття» вакансії. Тобто, прийом рішення на користь одного з кандидатів.

11 узгодження з здобувачем дати передбачуваного виходу на роботу, підготовка, обговорення та укладання з ним проекту трудового договору.

12 адаптація кандидата (за це відповідає HR-відділ, навіть якщо підбір здійснювало агентство) і супровід його протягом випробувального терміну. У разі необхідності - здійснення гарантійної заміни. Так, так, гарантійної заміни кандидата, ви все правильно прочитали. Професійні рекрутингові агентства дійсно її забезпечують.

Особливу увагу слід виділити саме аналізу представлених резюме, адже це найперший фільтр кадидатів і, як результат, слід відібрати кандидатів, кваліфікація яких відповідає потребам компанії.

У резюме можна виділити наступну інформацію, яка впливає на вибір:

- досвід роботи;
- технічні навики;
- освіта;
- допоміжні навики та інше.

Найбільш поширеними атрибутами рейтингу кандидатів в загальному випадку є:

1 поточна компенсація
2 очікувана компенсація
3 освіта
4 спеціалізація
5 посада
6 найдавніша дата початку роботи
7 загальний досвід
8 відповідний досвід
9 зв'язок
10 поточний роботодавець
11 перерва в освіті
12 перерва в роботі.

Також слід врахувати, що на відміну від багатьох неструктурованих типів документів, дані в резюме мають частково структуровану форму, де інформація зберігається в окремих блоках.

Таким чином, користувачу просто необхідно вказати, які ключові слова він шукає, і веб-додаток відображатиме всі відповідні резюме на основі ключових слів. Користувач може отримати доступ до цього веб-додатку з будь-якої робочої станції, а також завантажити резюме на свою локальну робочу станцію. Все це має відбуватися в режимі реального часу. Якщо щойно додане резюме є релевантним, то воно має бути негайно відібране веб-додатком.

1.2 Алгоритм ранжування резюме для підбору кадрів

В даний час існує безліч методів інтеллектуального аналізу даних [6 - 7]. Більшість цих методів грунтуються на одному з 3-х основних підходів: ймовірносному підході, штучних нейронних мережах чи деревах рішень.

Крім того слід пам'ятати, що дані частіше зашумлені лишнями інформація, яка може створювати додаткові проблеми для аналізу [7].

В даній роботі пропонується застосувати векторні методи. Такі методи використовують векторну модель представлення тексту. Як правило, для класифікації використовується скалярне множення векторов. Вектор документа чітко скалярно множиться з векторами категорій і чим більше скалярне множення, тим більше вірогідність, що документ потрапляє в цю категорію.

Розглянемо моделі представлення даних у задачах підбору кадрів.

Терм-документная матриця представляє собою математичну матрицю, що описує частоту термінів, які зустрічаються в колекції документів. В терм-документній матриці строки відповідають документам в коллекции, а столбцы соответствуют термінам. Існують різні схеми визначення значення кожного елемента матриці. Однією з таких є схема TF-IDF. Вони корисні в області обробки природньої мови, особливо в методах латентно-семантичного аналізу. Рогзлянемо більш детально статистичний показник TF-IDF.

TF (term frequency — частота слова) — відношення числа входжень обраного слова до загальної кількості слів документа. Таким чином, оцінюється важливість слова в межах обраного документа.

$$\mathrm{tf}(t,d) = \frac{n_i}{\sum_k n_k} \quad (1)$$

де n_i є число входжень слова в документ, а в знаменнику — загальна кількість слів в документі.

IDF (inverse document frequency — обернена частота документа) — інверсія частоти, з якою слово зустрічається в документах колекції. Використання IDF зменшує вагу широковживаних слів.

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|(d_i \supset t_i)|} \quad (2)$$

де

|D| — кількість документів колекції;

$\overline{|(d_i \supset t_i)|}$ — кількість документів, в яких зустрічається слово t_i (коли t_i не равен 0)

Вибір основи логарифму у формулі не має значення, адже зміна основи призведе до зміни ваги кожного слова на постійний множник, тобто вагове співвідношення залишиться незмінним.

Іншими словами, показник TF-IDF це добуток двох множників: TF та IDF.

Більшу вагу TF-IDF отримають слова з високою частотою появи в межах документа та низькою частотою вживання в інших документах колекції.

Векторна модель - в інформаційному пошуку представлення колекції документів векторами одного загального для всієї колекції векторного простору.

Векторна модель є основою для вирішення багатьох завдань інформаційного пошуку таких як:

- пошук документа за запитом,
- класифікація документів,
- кластеризація документів.

Баготьма способами можно визначити масу терма в документі - «Важливість» слова для ідентифікації даного тексту. Наприклад, можна просто розрахувати кількість входжень терма в документі, так звану частоту терма, - чим частіше слово зустрічається в документі, тим більша у нього буде маса. Якщо терм не зустрічається в документі, то його маса в документі дорівнює нулю.

Більш формально це твердження можна представити у вигляді формули:

$$dj = (w_{1j}, w_{2j}, \ldots, w_{nj}), \quad (3)$$

де dj - векторне представлення j-го документа, wij - вага i-го терма в j-м документі, n - загальна кількість різних термов у всіх документах колекції.

1.3 Аналіз існуючих програмних продуктів з автоматизації рекрутингу

Розглянемо існуючі на сьогодні програмні засоби, які вирішують подібні і схожі завдання та проблеми, проаналізуємо їх переваги та недоліки.

Після пошуку та аналізу, було знайдено значну кількість програмних продуктів. Більшість з них – є платними і поширюються за моделлю SaaS (програмне забезпечення як послуга), що іноді не є прийнятним. (Clustree, Cebglobal та інші)

Також є продукти для кінцевого користувача (E-Staff Рекрутер, Microsoft Dynamic CRM, Oracle Taleo, 1С: Предприятие 8. Кадровое агентство, SAP та інші**)**, але вони являють собою лише автоматизоване робоче місце рекрутера, що спростовує його роботу, але не виконує необхідних задач.

Перевагами продуктів які поширюються за моделлю SaaS є:

- швидке створення автоматизованого робочого місця рекртутера;
- створити робоче місце рекрутера з повністю структурованим сховищем даних, без професійних знань та навичок;
- наявність демо-версії, яка дає можливість скласти враження щодо програмного забезпечення;
- відсутність накладних витрат щодо підтримки й забезпечення необхідним обладнанням.

Але б наскільки привабливими не були переваги, існують наступні недоліки:

- націлені лише на англомовні компанії і їх сервіси;
- передбачають підписання довгострокового контракту;
- передбачають підписку для сервісу (щомісячні, щоквартальні витрати);
- мають обмежені функціональні можливості (для роширення можливостей необхідно доплачувати);
- часто не мають об’єктивних оцінок ефективності, а суто суб’єктивні;
- відсутня інтелектуальна складова (класифікація, ранжування).

Перевагами продуктів для кінцевого користувача:

- повне налаштування програмного забезпечення;
- дозволяє повністю або частково автоматизувати процеси рекрутингу;
- підтримка з боку розробника;
- купівля всієї програмної системи з усіма можливостями.

Існують наступні недоліки продуктів для кінцевого користувача:

- накладні витрат щодо підтримки й забезпечення необхідним обладнанням;
- відсутня інтелектуальна складова (класифікація, ранжування);
- вирішення продуктом тільки декількох специфічних процесів рекрутингу.

Для більш детального розуміння переваг та недоліків існуючих програмних рішень розглянемо по одному рішенню з кожної групи.

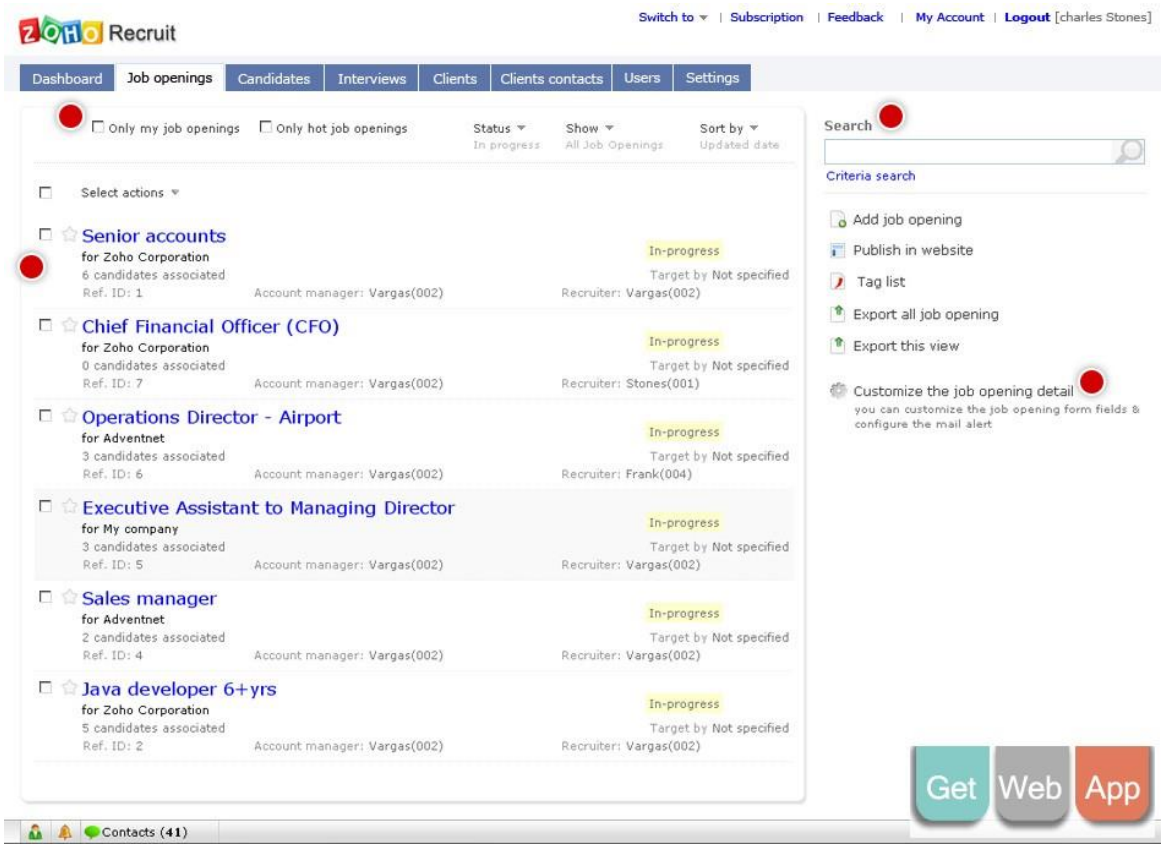

Рисунок 1.1 – Інтерфейс головної сторінки модулю рекрутингу програмного забезпечення Zoho

Zoho, програмне забезпечення як послуга:

- інтеграція з великою кількістю сервісів комунікації;
- інтеграція з великою кількість сервісів пошуку кандидатів, розміщення вакансій;
- зручний інтерфейс;
- передвизначені сутності та бізнес-процеси рекрутингу.

Крім того Zoho має ще й такі недоліки:

- відсутність можливість кастомізації;
- ця системи не гнучка, оскільки кандидат повинен завантажити там резюме в певному форматі, але для даного типу продуктів це скорішу виключення;
- відсутність фільтрації резюме у відповідності до вакансії.

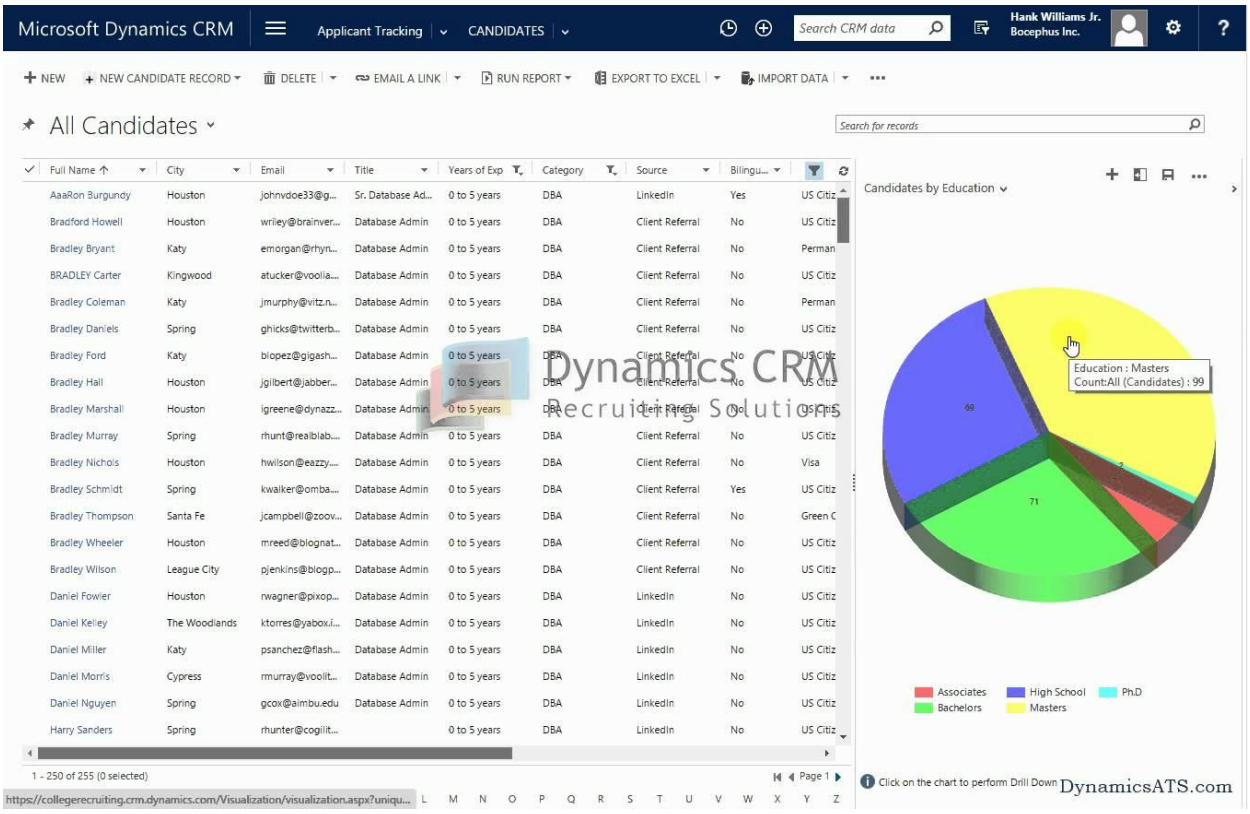

Рисунок 1.2 – Інтерфейс головної сторінки модулю рекрутингу програмного забезпечення Microsoft Dynamic CRM

Microsoft Dynamic CRM, як продукт для кінцевого користувача має такі переваги:

- інтеграція з іншими сервісами компанії Майкрасофт;
- інтеграція з Yammer і Skype;
- створення звітності у вигляді файлів офісного пакету програмного забезпечення від компанії Майкрасофт;

- оргінізаційна структура підприємства;
- власна база даних, яка оперує метаданими, що створює широкі можливості кастомізації;
- функціональність настройки бізнес-процесу побудований на Windows Workflow Foundation і забезпечує можливість програмування моделі, підвищення продуктивності та інструменти для швидкої побудови бізнес-процесів.
- в даній системі гарантується цілісність і збереження даних.

Крім того Microsoft Dynamic CRM має ще й такі недоліки:

- відсутність модуля Рекрутер з передвизначеними сутностями та бізснес-процесами характерними для процесу пошуку кадрів;
- в даній системі команда може публікувати вакансії, після резюме будуть отримані системою і відсортовані в список, далі резюме необхідно перевести в інший статус для проведення наступних раундів інтерв'ю. Такий шлях потребує багато часу та зусиль;
- для вирішення першого недоліку є можливим створення сервісу-інтеграції на основі існуючого програмного інтерфейсу. Але даний шлях потребує втручання кваліфікованих розробників, що не завжди є ліпшим рішенням;
- існує можливість кастомізації, але даний процес потребує капіталовкладень і часу на освоєння з продуктом;
- існує веб-додаток, який є більш зручним для використання користувачем, але на даний момент функціональність веб-додатку значно обмежена;
- ця система не гнучка, оскільки кандидат повинен завантажити резюме в певному форматі. Формати змінюються від системи до системи.

Отже, робимо висновок, що у розроблюваного програмного забезпечення відсутні аналоги, які змогли б повністю задовольнити вимогам щодо якісного процесу підбору персоналу. Крім того, розглянуті програмні продукти є комерційними.

1.4 Постановка задачі на розробку програмного забезпечення веб-додатку для ранжування резюме в рекрутингу

Виходячи з розглянутих особливостей організації процесів підбору кадрів та сучасних програмних продуктів, що використовуються в рекрутингу, сформулюємо наступну постановку задачі: розробити програмне забезпечення веб-додатку для

ранжування резюме кандидатів в рекрутингу, що дозволить підвищити ефективність роботи рекрутерів компаній шляхом зменшення ймовірності виникнення помилок в роботі через вплив людського фактору та скорочення часу на обробку даних о кандидатах на вакансії.

Програмне забезпечення повинно виконувати наступні функції:

- перегляд доступних резюме в системі: резюме зберігають по визначеному шляху, з якого користувач має можливість завантажувати нові резюме;
- ранжування резюме кандидатів на основі наданого профілю роботи;
- пошук найбільш релевантних резюме: система має виконати ранжування існуючих резюме на основі наданого профілю роботи і відобразити список відповідних;
- додавання / модифікація / видалення профілей роботи для пошуку релевантного співробітника: користувач може додавати / модифікавати / видаляти профілі роботи для пошуку найбільш релевантних резюме;
- додавання / видалення резюме в систему: резюме зберігають по визначеному шляху, до якого користувач має можливість завантажувати або видаляти резюме;
- аутентифікація: можливість використання програмного забезпечення зареєстрованими користувачами;
- управління рекрутерами: додавання / модифікація / видалення рекрутерів в системі.

До складу технічних засобів висуваються вимоги не нижче наступних:

- центральний процесор - Intel або AMD від 2.0 GHz і вище;
- жорсткий диск – об’єм не менший 40 Гб;
- оперативна пам’ять – об’єм не менший 2 Гб;
- мережевий адаптер або інші засоби доступу до мережі Internet.

Вихідні тексти програми повинні бути реалізовані на мові Java з використанням фреймворку Spring. В якості інтегрованої середовища розробки програми повинна бути використане середовище IntelliJ IDEA 2018.

У даному проекті необхідно використати такі технології:

- HTML 5, CSS 3 – розмітка та стилізація веб-сторінок;
- JavaScript (AJAX, Angular) – клієнтські скрипти, асинхронне завантаження даних, реактивність користувацького інтерфейсу;
- PostgreSQL –обробка та збереження даних у БД;
- Java, Spring Framework – серверна-розробка.

Детальний виклад вимог оформлений у вигляді документа «Технічне завдання» (відповідно до ЄСПД ГОСТ 19.201-78) і розміщений у Додатку до дипломної роботи[4-6]

РОЗДІЛ 2. ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБ-ДОДАТКУ ДЛЯ РАНЖУВАННЯ РЕЗЮМЕ В РЕКРУТИНГУ

2.1Ескізний проект

Ескізний проект представляє результати зовнішнього проектування програмного забезпечення: розробляються структури вхідних та вихідних даних, потоки і інтерфейси їхнього введення і виведення, загальна технологія розв'язання задачі з використанням обчислювальної техніки і необхідна для цього інформаційна модель. Спочатку розробляється діаграма варіантів використання, виконується структуризація варіантів використання та їх специфікація, проводиться уточнення варіантів використання на діаграмах діяльності. Також створюється логічний проект користувальницького інтерфейсу. Для розробки діаграм було використано UML – уніфіковану мова моделювання, що використовується у парадигмі об'єктно-орієнтованого програмування. Для формалізації представлення сценаріїв варіантів використання побудовано діаграму діяльності, спроектований логічний проект користувальницького інтерфейсу, а також розроблена інформаційна модель за допомогою діаграми класів.

2.1.1 Моделювання варіантів використання веб-додатку для ранжування резюме в рекрутингу та їх специфікація

Діаграма прецедентів (діаграма варіантів використання) – діаграма, на якій зображено відношення між акторами та прецедентами в системі. Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи, асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання. Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою варіантів використання. Варіант використання використовують для описання послуг, які система надає актору.

Основними кандидатами в актори є:

− рекрутер - потенційний користувач, який бажає виконати пошук релевантного резюме;

− адміністратор - людина, відповідальна за додавання користувачів до веб-додатку.

Розробимо діаграму варіантів використання веб-додатку для ранжування резюме в рекрутингу. Діаграма варіантів використання представлена на рисунку 2.1.

Рисунок 2.1 – Діаграма варіантів використання веб-додатку для ранжування резюме в рекрутингу

2.1.2 Специфікація варіантів використання веб-додатку для ранжування резюме в рекрутингу

Використовуючи розроблену діаграму варіантів використання веб-додатку для ранжування резюме в рекрутингу, опишемо специфікацію створених варіантів використання, що наведена нижче.

Специфікація варіантів використання

1. Перегляд завантажених резюме

Основні дійові особи: Рекрутер, адміністратор.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання

Розширюється «Завантаження, видалення резюме», «Завантаження резюме». Включається «Пошук релевантних резюме».

Передумова

Варіант використання «Аутентифікація»

Короткий опис

Даний варіант використання дає можливість користувачу переглянути каталог завантажених резюме.

Потік дій: даний варіант використання починається відразу після аутентифікації користувача.

Базовий потік – Перегляд завантажених резюме.

Послідовність дій:

- Користувач запускає сервіс, з'являється головна сторінка;
- Користувач проходить аутентифікацію;
- Користувач обирає пункт меню Резюме, з’являється сторінка з каталогом завантажених резюме.

Альтернативний потік – Завантаження, видалення резюме

Послідовність дій:

- Користувач запускає сервіс, з'являється головна сторінка;
- Користувач проходить аутентифікацію;
- Користувач обирає пункт меню Резюм, з’являється сторінка з каталогом завантажених резюме та можливими діями щодо кожного з них.

Альтернативний потік – Завантаженя резюме.

Послідовність дій:

- Користувач запускає сервіс, з'являється головна сторінка;
- Користувач проходить аутентифікацію;

- Користувач тисне кнопку Завантажити резюме. Система збереже на сервері завантажений файл. Користувач може переглянути користувач може переглянути каталог завантажених резюме та побачити щойно додане резюме.

Постумова

При успішному завершенні варіанту використання «Перегляд завантажених резюме» користувач може перейти до інших варіантів використання.

2. Пошук релевантних резюме

Основні дійові особи: Рекрутер, адміністратор.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання

Включає «Перегляд завантажених резюме», «Ранжування завантажених резюме»

Передумова

Варіант використання «Аутентифікація».

Короткий опис

Даний варіант використання дає можливість користувачу виконати ранжування завантажених резюме згідно обраних критеріїв.

Потік дій: даний варіант використання починається відразу після аутентифікації

Базовий потік – Пошук релевантних резюме.

Послідовність дій:

- Користувач переходить на сторінку пошуку.
- Користувач вводить критерії пошуку.
- Користувач тисне кнопку Шукати і переходить на сторінку з найбільш релевантиними резюме, що завантажені в систему.

Альтернативний потік – відсутній.

Постумова

При успішному завершенні варіанту використання «Перегляд завантажених резюме» користувач може перейти до інших варіантів використання.

3. Ранжування резюме

Основні дійові особи: Рекрутер, адміністратор.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання

Відсутні.

Передумова

Варіант використання «Аутентифікація».

Короткий опис

Даний варіант використання дає можливість виконати ранжування резюме, що завантажені до системи, згідно обраних критеріїв з використанням статистичного критерію TF-IDF.

Потік дій: даний варіант використання починається відразу після вибору критеріїв ранжування.

Базовий потік – Ранжування резюме

Послідовність дій:

- Користувач переходить на сторінку пошуку.
- Користувач вводить критерії пошуку.
- Користувач тисне кнопку Шукати. Система анлалізує і виконує ранжування резюме згідно обраних критеріїв. Відбувається перехід на сторінку з найбільш релевантиними резюме, що завантажені в систему.

Альтернативний потік – Ранжування резюме згідно збереженого профілю

Послідовність дій:

- Користувач переходить на сторінку збережених профілей пошуку.
- Користувач обирає необхідний профіль.
- Користувач тисне кнопку Шукати. Система анлалізує і виконує ранжування резюме згідно обраних критеріїв. Відбувається перехід на сторінку з найбільш релевантиними резюме, що завантажені в систему.

Постумова

При успішному завершенні варіанту використання «Ранжування резюме» користувач може перейти до інших варіантів використання.

4. Додавання / модифікація / видалення профілей роботи для пошуку релевантного співробітника

Основні дійові особи: Рекрутер, адміністратор.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання

Включає «Ранжування резюме»

Передумова

Варіант використання «Аутентифікація».

Короткий опис

Даний варіант використання дає можливість переглянути сторінку профілей роботи та виконати додавання / модифікацію / видалення профілей роботи для пошуку релевантного співробітника.

Потік дій: даний варіант використання починається відразу після аутентифікації.

Базовий потік - Додавання / модифікація / видалення профілей роботи для пошуку релевантного співробітника.

Послідовність дій:

- Користувач виконує аутентифікацію;
- Користувач обирає пункт меню Профілі роботи;
- Після буде відображена сторінка профілей роботи і користувач може виконати додавання / модифікацію / видалення профілей роботи для пошуку релевантного співробітника.

Альтернативний потік – відсутній.

Постумова

При успішному завершенні варіанту використання «Додавання / модифікація / видалення профілей роботи для пошуку релевантного співробітника» користувач може перейти до інших варіантів використання.

5. Управління рекрутерами

Основні дійові особи: Адміністратор.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання:

Відсутні.

Передумова

Варіант використання «Аутентифікація». Обов’язкова ідентифікація користувача, як адміністратора.

Короткий опис

Даний варіант використання дає можливість редагувати список рекрутерів, зареєстрованих у системі.

Потік дій: даний варіант використання починається відразу після аутентифікації корстувача, як адміністратора.

Базовий потік – Редагування кошику

Послідовність дій:

- Адмінстратор перходить у пункт меню Рекрутери;
- Відображається список рекрутерів, який можна редагувати за допомогою інтуєтивно зрозумілих кнопок: додати нового рекрутера, видалити рекрутера, оновити інформацію по рекрутеру.

Альтернативний потік - відсутній.

Постумова

При успішному завершенні варіанту використання «Управління рекрутерами» користувач може перейти до інших варіантів використання.

6. Аутентифікація

Основні дійові особи: Рекрутер, адміністратор.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання

Відсутні.

Передумова

Користувач має бути зареєстрований у системі.

Короткий опис

Варіант використання призначений для аутентифікації у клієнтському веб-сайті.

Потік дій: даний варіант використання починається відразу після перегляду клієнтського веб-сайту.

Базовий потік – Аутентифікація.

Послідовність дій:

- користувач відкриває сторінку аутентифікації;

- система формує сторінку та відображає її;
- користувач заповнює дані форми та натискає Увійти;
- система аутентифікує користувача;
- система формує головну сторінку та відображає її.

Альтернативний потік – відсутній.

Постумова

При успішному завершенні варіанту використання «Аутентифікації» користувач може перейти до інших варіантів використання.

7. Додавання/видалення резюме

Основні дійові особи: Рекрутер, адміністратор.

Інші учасники прецеденту: відсутні.

Зв'язки з іншими варіантами використання: відсутні.

Передумова

Для’ додавання/видалення резюме користувачу обов'язково необхідно пройти перегляд завантажених резюме.

Короткий опис

Даний варіант використання дає можливість виконати додавання/видалення резюме.

Потік дій: даний варіант використання починається відразу після перегляду завантажених резюме.

Базовий потік – Додавання резюме.

Послідовність дій:

- «Перегляд завантажених резюме»;
- Користувач виконує клік на кнопку «Додати»;
- У списку обирає «Резюме»;
- Після буде відображена форма з можливістю завантаження файлу резюме з локального сховища. У систему буде завантажено файл з резюме.

Альтернативний потік – Видалення резюме.

Послідовність дій:

- «Перегляд завантажених резюме»;
- Користувач виконує клік на кнопку «Видалити» у списку завантажених резюме. З системи буде видалено обране резюме.

Постумова

При успішному завершенні варіанту використання «Додавання/видалення резюме» модератор може перейти до інших варіантів використання.

2.1.3 Розробка діаграм діяльності прецедентів програмного забезпечення

Діаграма діяльності – це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. «Дія» є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Для формалізації представлення варіантів використання були створені діаграми діяльності для деяких з основних варіантів використання, які зображені на рисунках представлених нижче.

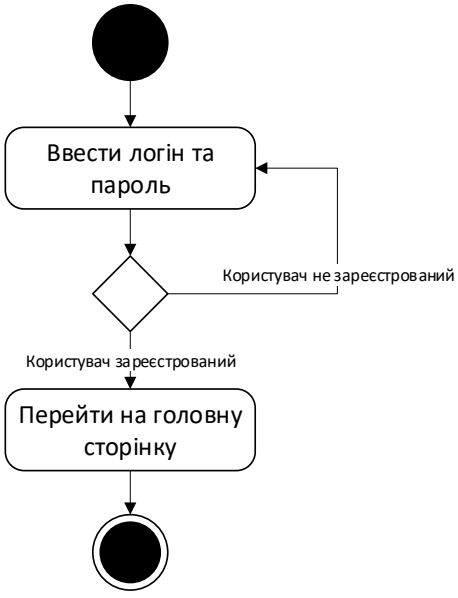

Рисунок 2.2 – Діаграма діяльності варіанту використання «Аутентифікація» веб-додатку для ранжування резюме

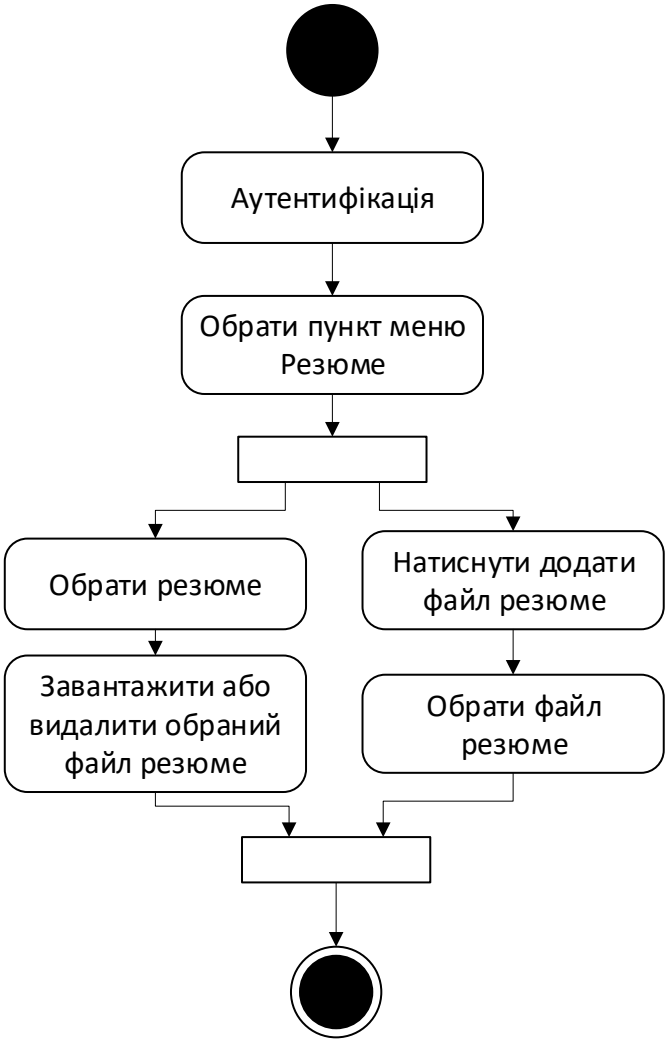

Рисунок 2.3 – Діаграма діяльності варіанту використання «Перегляд завантажених резюме» веб-додатку для ранжування резюме

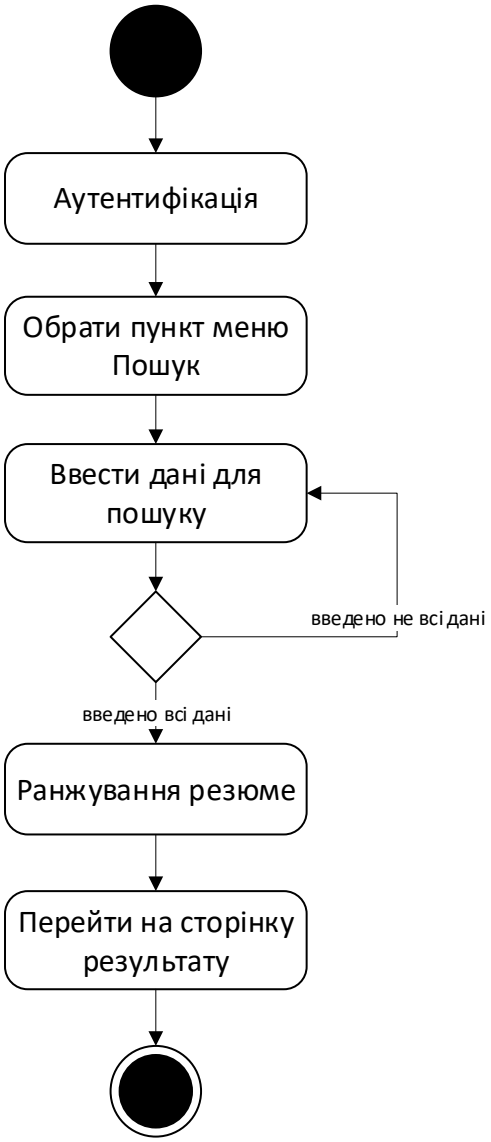

Рисунок 2.4 – Діаграма діяльності варіанту використання «Пошук релевантних резюме» веб-додатку для ранжування резюме

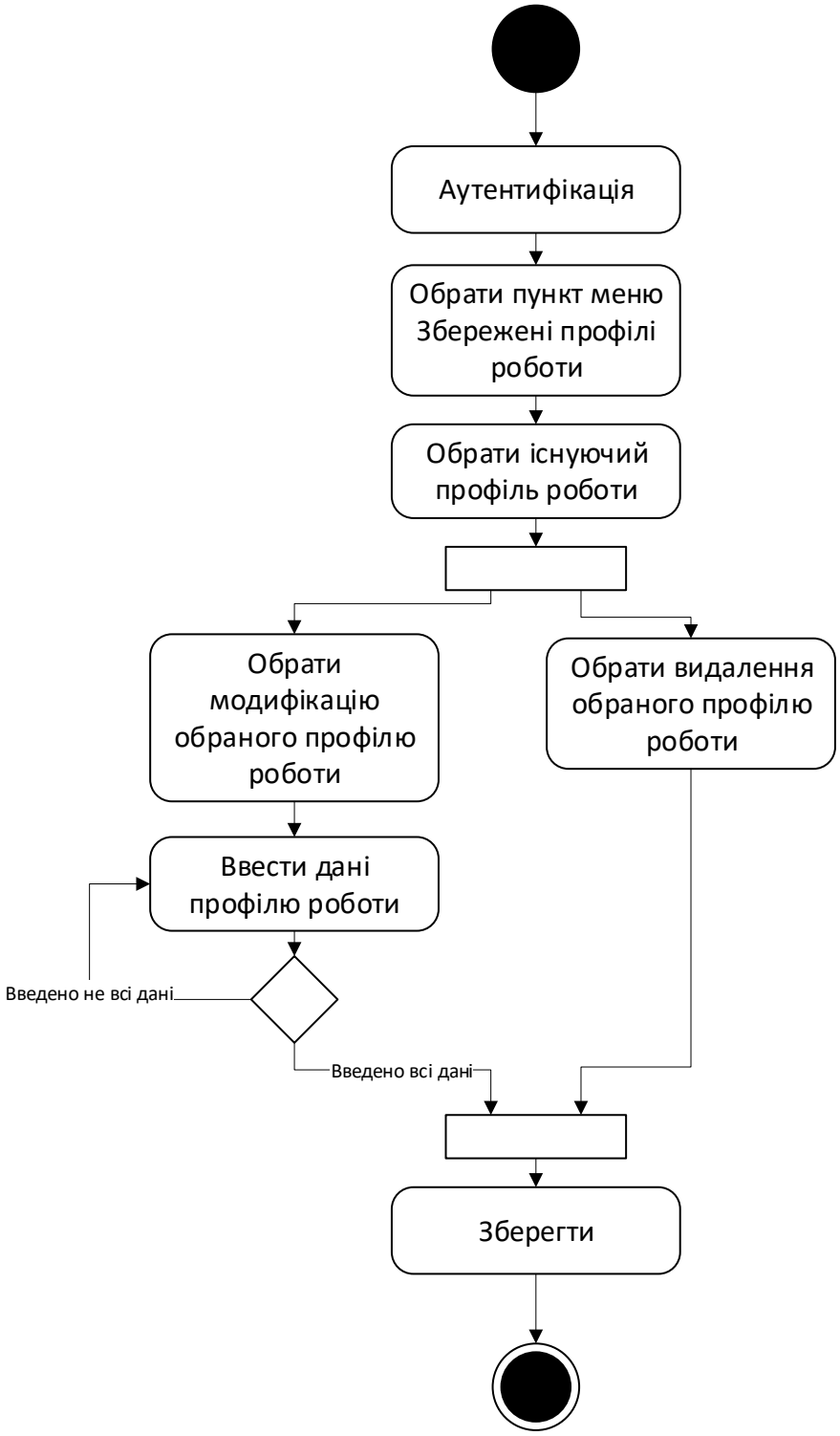

Рисунок 2.5 – Діаграма діяльності варіанту використання «Додавання / модифікація / видалення профілей роботи для пошуку релевантного співробітника» веб-додатку для ранжування резюме

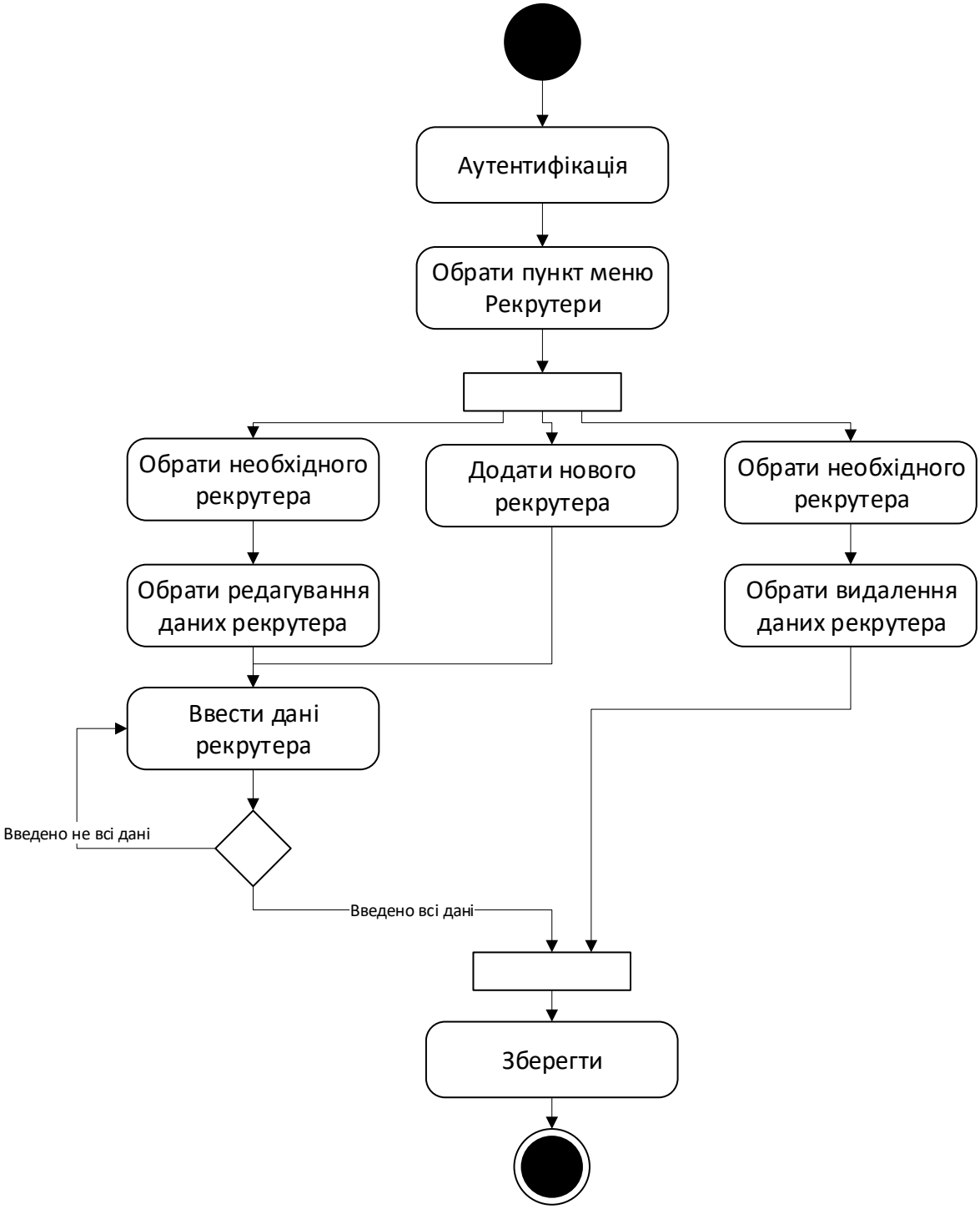

Рисунок 2.6 – Діаграма діяльності варіанту використання «Управління рекрутерами» веб-додатку для ранжування резюме

2.1.4 Проект користувацького інтерфейсу

Інтерфейс користувача це засіб зручної взаємодії користувача з інформаційною системою. Сукупність засобів для обробки та відбиття інформації, якнайбільше пристосованих для зручності користувача; у графічних системах інтерфейс користувача, втілюється багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їхнім розташуванням, сортуванням елементів вікон, гнучкими налаштуваннями як самих вікон, так і окремих їх елементів (файли, теки, ярлики, шрифти тощо), доступністю багатокористувацьких налаштувань.

В цьому підрозділі представлені схеми екранних форм, систем меню, діалогів і т.д. для проектування зовнішнього інтерфейсу. Інтерфейс користувача - це система правил і засобів, що забезпечує взаємодію програми з користувачем.

Розробимо проект користувацького інтерфейсу веб-додатку для ранжування резюме в рекрутингу.

При завантаженні браузера та переходу на сайт, користувач бачить сторінку аутентифікації (рисунок 2.7), яка призупиняє роботу програми до тих пір, поки користувач не буде аутентифікований в системі.

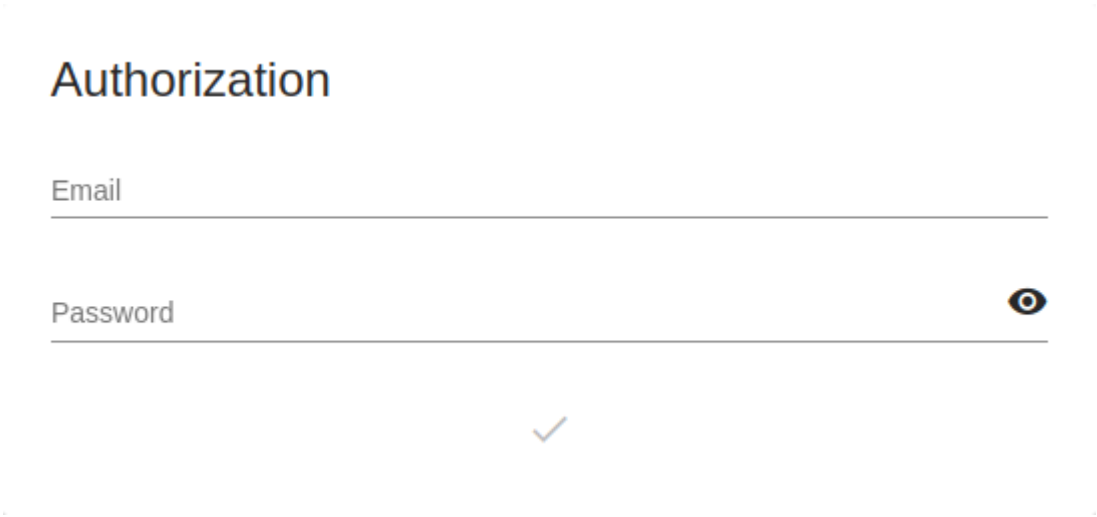

Рисунок 2.7 - Екран при переході на сайт – аутентифікація

Після успішної аутентифікації користувача буде відображена головна сторінка веб-додатку, яка містить головне меню зображене на рисунку 2.8.

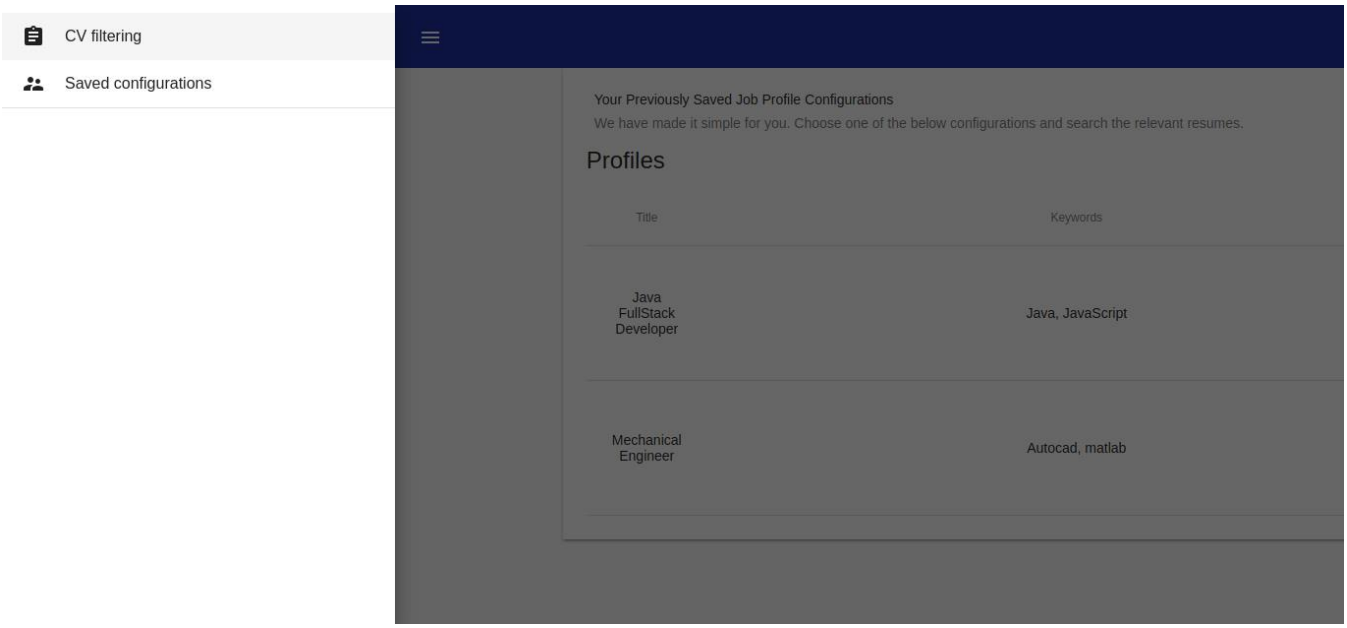

Рисунок 2.8 – Екран головної сторінки з головним меню

Завантаження нових файлів резюме буде здійснюватись за допомогою вбудованого файлового менеджера та його програмного інтерфейсу, а кнопка завантаження виглядатиме у відповідності до рисунку 2.9.

Рисунок 2.9 – Кнопка завантаження нового резюме в систему

Сторінка з можливістю перегляду та ранжування резюме (рисунок 2.10), яка дозволить користувачу виконати пошук найбільш релевантного резюме, зберегти новий профіль роботи для майбутнього пошуку, а також переглянути результат пошуку та завантажити необхідний файл резюме.

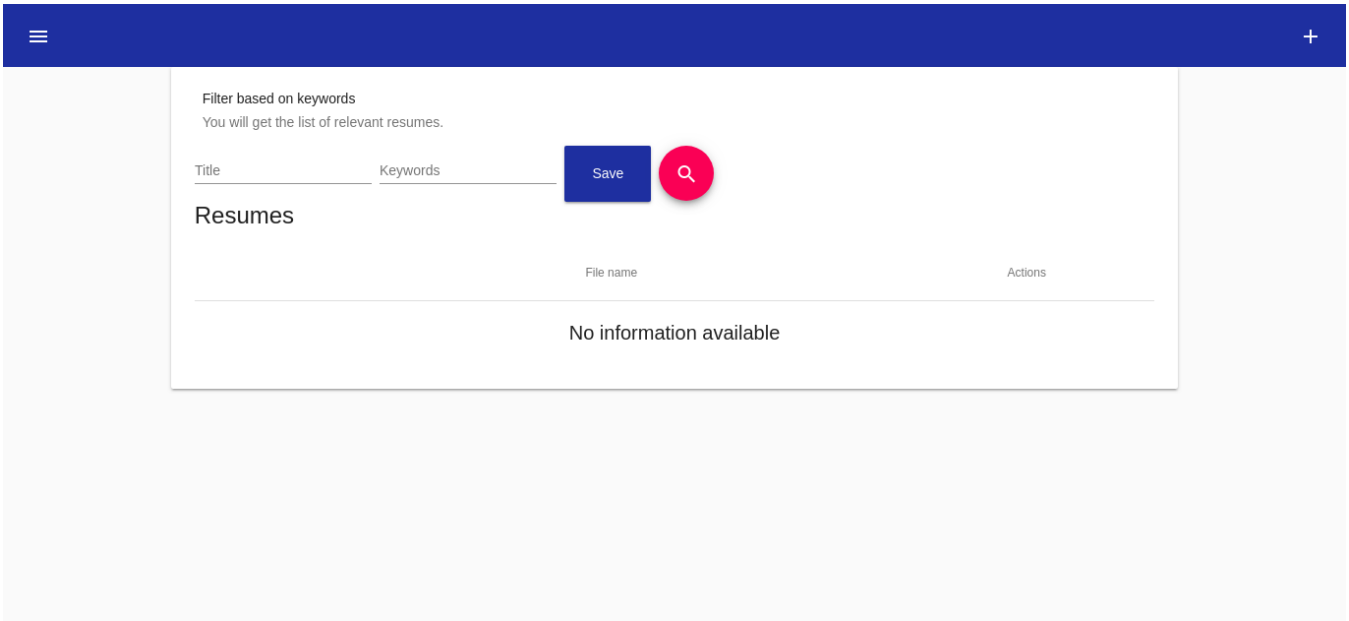

Рисунок 2.10 – Екран з можливістю ранжування резюме, збереження профілю роботи та завантаження файлів резюме із списку результатів

Сторінка перегляду збережених профілей роботи (рисунок 2.11) дозволить виконати пошук у відповідності до обраного профілю, а також видалити та редагувати обраний профіль.

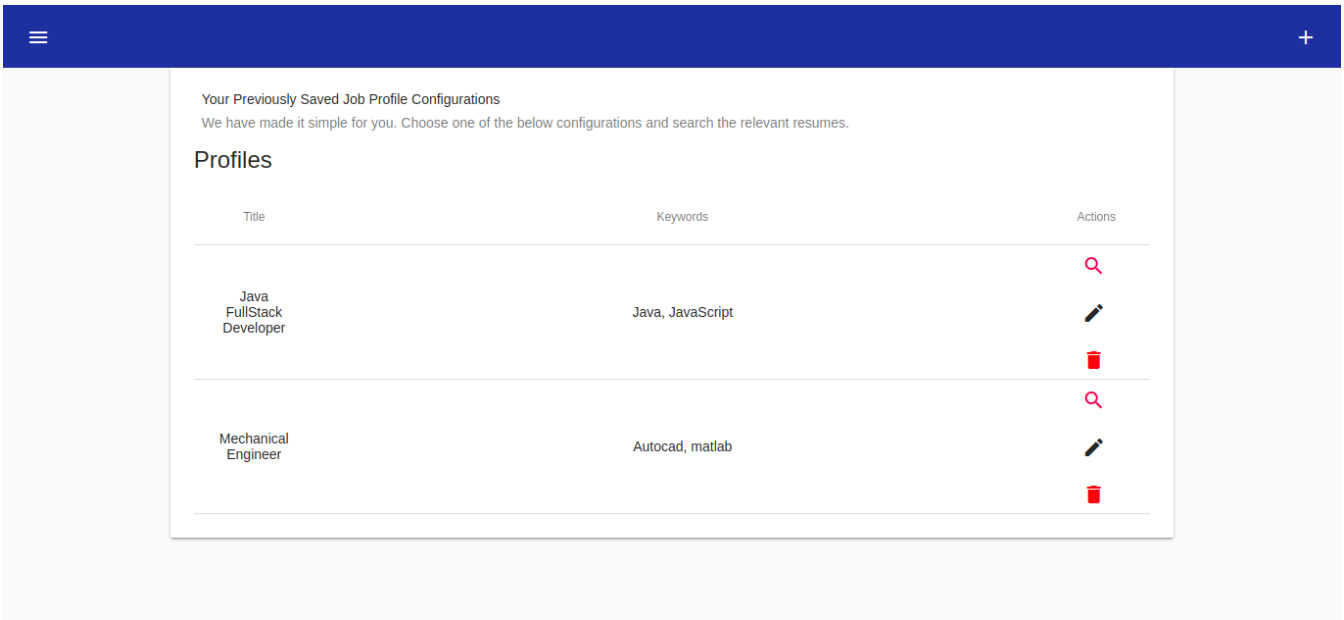

Рисунок 2.11 – Екран з можливістю перегляду, редагуваня збережених профілей роботи та пошуку у відповідності до обраного профілю

2.2 Технічний проект

Проаналізувавши ескізний проект було розроблено технічний проект програмного забезпечення веб-додатку для ранжування резюме в рекрутингу шляхом побудови статичних і динамічних моделей. Статична модель представлена діаграмами класів, а динамічна - діаграмами взаємодії.

2.2.1 Розробка концептуальної моделі програмного забезпечення

Концептуальна модель – модель предметної області, що складається з переліку взаємопов'язаних понять, що використовуються для опису цієї області, разом з властивостями й характеристиками, класифікацією цих понять, за типами, ситуацій, ознаками в даній області і законів протікання процесів в ній.

Розробимо концептуальну модель програмного забезпечення відповідно нотації Мартіна (Crow’s Foot). Діаграма концептуальної моделі програмного забезпечення представлена на рисунку 2.12.

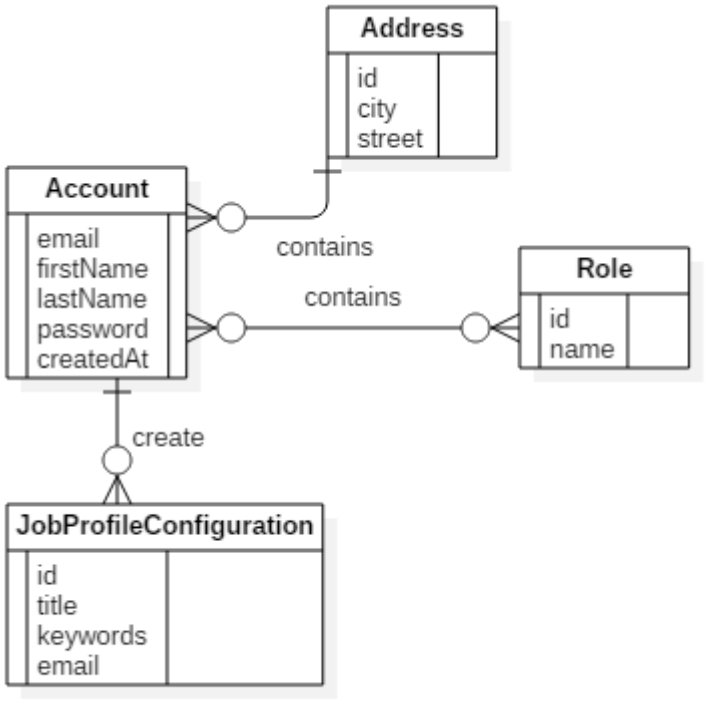

Рисунок 2.12 – Концептуальна модель програмного забезпечення веб-додатку для ранжування резюме в рекрутингу

2.2.2 Статична модель програмного забезпечення

Статичні діаграми представляють сутності і зв'язки між ними, або сумарну інформацію про сутності і зв'язках, або сутності та зв'язки, що існують в якійсь визначений момент часу. Вони не показують способів поведінки цих сутностей.

До статичних діаграм відносять діаграму класів і об'єктів. Провівши аналіз предметної області були виявлені об'єкти і класи програмного забезпечення веб-додатку для ранжування резюме в рекрутингу. Для того, щоб показати структуру програмного забезпечення було прийнято рішення про об'єднання діаграми об'єктів і класів в одну.

На основі моделі варіантів використання, концептуальної моделі і моделі програмного інтерфейсу була розроблена статична модель, представлена діаграмою класів, яка зображена на рисунку 2.13.

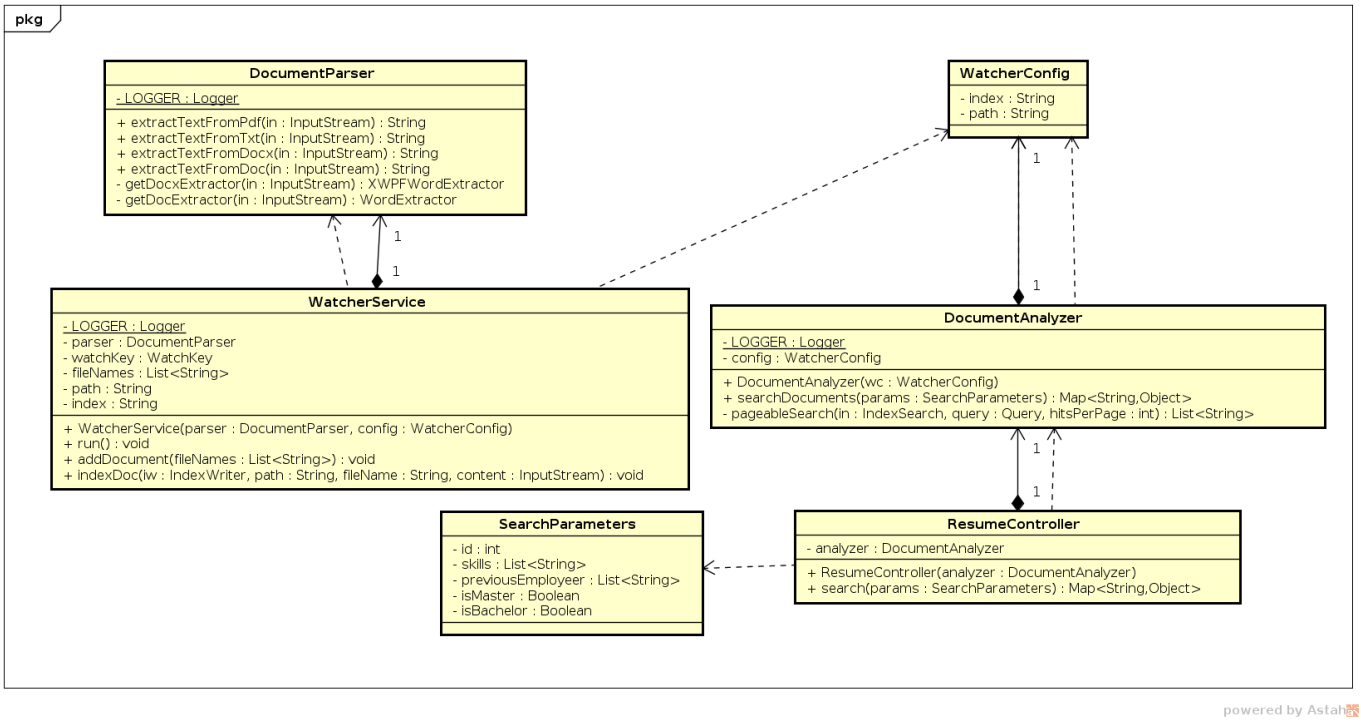

Рисунок 2.13 – Діаграма класів програмного забезпечення веб-додатку для ранжуваня резюме в рекрутингу

Спираючись на діаграму класів, було розроблено специфікацію класів програмного забезпечення онлайн-платформи проведення опитувань:

- DocumentParser – клас, що виконує зчитування даних з завантажених файлів;
- WatcherService – клас, що виконує нагляд над директорією сервера, де зберігаються завантажені файли;
- WatcherConfig – клас, що опсиує налаштування: шлях до збереження файлів та їх індексів;
- DocumentAnalyser – клас, що виконує ранжування файлів за допомогою TF-IDF;
- SearchParameters – клас, що описує критерії пошуку;
- ResumeController – клас, що реалізує спілкування REST доступ до серверу.

2.2.3 Динамічна модель програмного забезпечення

Динамічна модель системи це сукупність співвідношень, що визначають вихід системи в залежності від входу та стану системи. Динамічна модель відтворює зміни об'єкта, які відбуваються з плином часу, або особливості функціонування об'єкта. Динамічні моделі називають також функціональними. Динамічне моделювання використовується для опису поведінки об'єкта в будь-який довільний змінний момент часу.

Для того, щоб описати динамічну модель програмного забезпечення онлайн-платформи проведення опитувань, було розроблено діаграми послідовності. Діаграма послідовності – різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень. На діаграмі послідовностей показано у вигляді вертикальних ліній різні процеси або об'єкти, що існують водночас. Надіслані повідомлення зображуються у вигляді горизонтальних ліній, в порядку відправлення.

На основі моделі сценаріїв варіантів використання і статичної моделі програми була створена динамічна модель програмного забезпечення веб-додатку для

ранжування резюме в рекрутингу. На рисунках 2.14-2.15 наведені діаграми послідовності деяких з основних варіантів використання, які реалізують цю модель.

Рисунок 2.14 – Діаграма послідовності варіанту використання «Ранжування завантажених резюме» веб-додатку для ранжування резюме

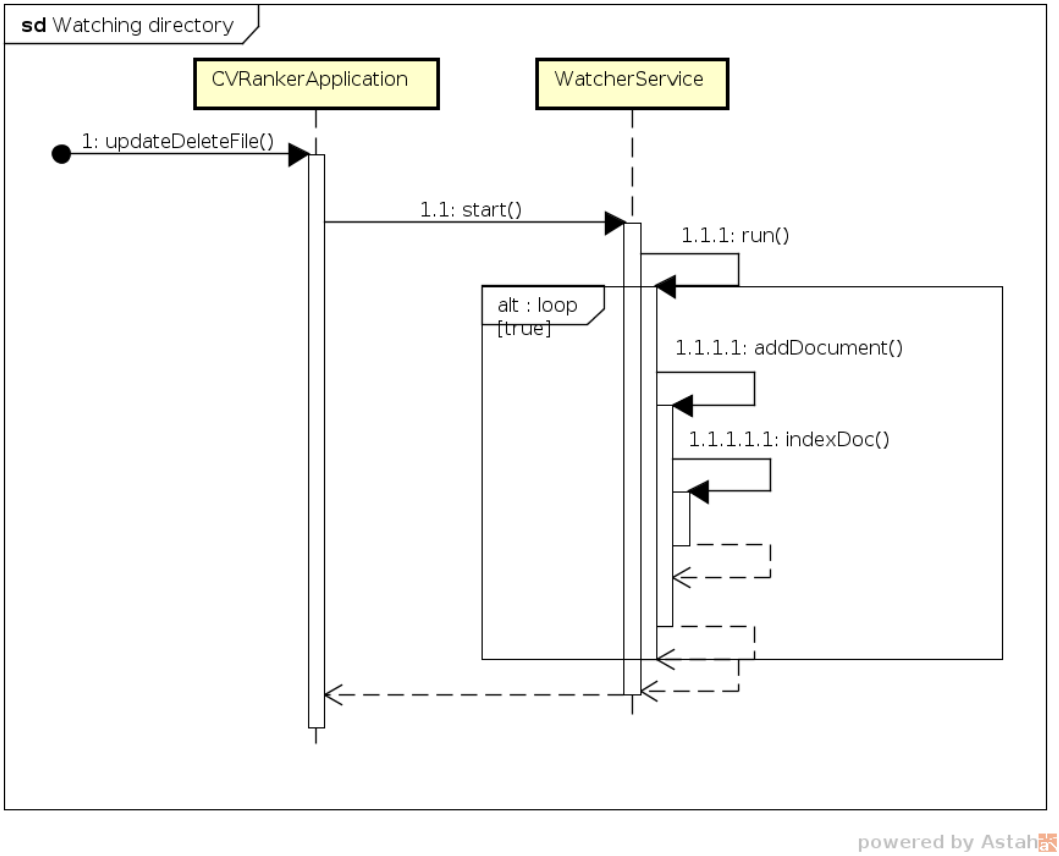

Рисунок 2.15 – Діаграма послідовності варіанту використання «Завантаження, видалення резюме» веб-додатку для ранжування резюме

2.2.4 Розробка логічної моделі структур даних програмного забезпечення

Логічна модель даних є початковим прототипом майбутньої бази даних. Вона будується в термінах інформаційних одиниць, але без прив'язки до конкретної СУБД.

Нам необхідно вводити, зберігати та використовувати дані, отже засобом накопичення та використання інформації нам служить база даних.

Логічна модель є основою для програмної реалізації бази даних, яка відображає взаємозв'язки між таблицями.

Розроблена логічна модель структури даних програмного забезпечення веб-додатку для ранжування резюме в рекрутингу зображена на рисунку 2.16 у вигляді діаграми класів.

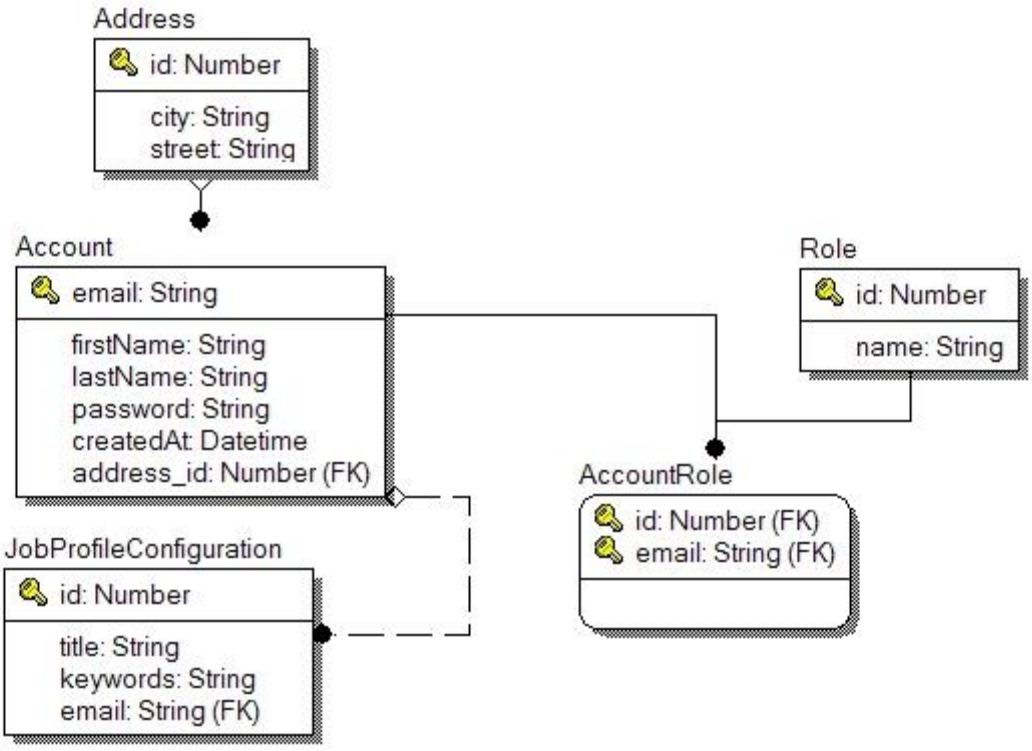

Рисунок 2.16 – Логічна модель даних веб-додатку для ранжування резюме в рекрутингу

Таблиці логічної моделі наведенні в таблицях 2.1-2.5.

Таблиця 2.1 - Таблиця «Адреса»

Поле	Тип поля	Ключ
Ідентифікатор	Число	PK
Місто	Строка	
Вулиця	Строка	

Таблиця 2.2 - Таблиця «Акаунт»

Поле	Тип поля	Клю

		ч
Електронна пошта	Строка	PK
Ім’я	Строка	
Прізвище	Строка	
Пароль	Строка	
Дата створення	Дата	
Ідентифікатор адреси	Число	FK

Таблиця 2.3 - Таблиця «Роль»

Поле	Тип поля	Ключ
Ідентифікатор	Число	PK
Найменування	Строка	

Таблиця 2.4 - Таблиця «АкаунтРоль»

Поле	Тип поля	Ключ
Ідентифікатор акаунта	Число	FK, PK
Ідентифікатор ролі	Число	FK, PK

Таблиця 2.5 - Таблиця «Профіль роботи»

Поле	Тип поля	Клю

		ч
Ідентифікатор	Число	PK
Найменування	Строка	
Ключові слова	Строка	
Електронна пошта	Строка	FK

2.3 Робочий проект

Виходячи з аналізу предметної галузі процесів підбору кадрів та сучасних програмних продуктів, що використовуються в рекрутингу, сформулюємо наступну постановку задачі: розробити програмне забезпечення веб-додатку для ранжування резюме кандидатів в рекрутингу, що дозволить підвищити ефективності роботи рекрутерів компаній шляхом зменшення ймовірності виникнення помилок в роботі через вплив людського фактору та скорочення часу на обробку даних о кандидатах на вакансії.

2.3.1 Обґрунтування вибору засобів розробки

У мови Java є багато переваг перед іншими мовами програмування, що дозволяє вирішувати з його допомогою практично будь-які завдання.

Нижче перераховані основні переваги Java:

- Мова Java проста для вивчення.При розробці Java було приділено велику увагу простоті мови, тому програми на Java, в порівнянні з програмами на інших мовах, простіше писати, компілювати, налагоджувати і вивчати.
- Java - це об'єктно-орієнтована мова. Це дозволяє створювати модульні програми, вихідний код яких може використовуватися багаторазово.
- Мова Java не залежить від платформи. Одним з основних переваг мови Java є можливість перенесення програм з однієї системи в іншу. Оскільки про-грами на Java не залежать від платформи як на рівні вихідного коду, так і на довічним рівні,

їх можна запускати в різних системах, що особливо важливо для програм, призначених для World Wide Web.

– Широкі можливості Java, простота застосування, незалежність від платформи і вбудовані функції захисту роблять цю мову программірова-ня одним з кращих для створення додатків для Internet.

В якості мови програмування буде використовуватися Java і Spring Framework, які показують кращу надійність і швидкість реалізації в розробці комерційного ПЗ. Крім того, Spring Framework дозволяє реалізувати найрізноманітніші типи веб-додатків, а також реалізовує та-кі ключові речі нового покоління, як IoC (Inversional of control) і DI (Dependency injection).

В якості СУБД передбачається використання PostgreSQL (реляційна система управління базами даних). Дана СУБД характеризується клієнт-серверною архітектурою, яка відмінно підходить для реалізації веб-додатків, оскільки дозволять утримувати базу даних програми на окремому сервері. Для виконання адміністрування бази даних можливо використання PGAdmin.

Як сервер для розгортання передбачається використання Tomcat7, як найбільш поширеного.

У якості середовища розробки передбачається використання Intellij Idea, як найбільш багатофункціональне середовище розробки.

Також для розробки системи потрібно використати мову розмітки гіпертекстових документів HTML5 та каскадну таблицю стилів CSS3. Це найкращі та найзручніші інструменти з подібним призначенням, вони і дозволяють будувати web-сторінки із найрізноманітнішими розмірами та формами. Для інтерактивної та активної взаємодії користувача з web-додатком використовується мова програмування javascript. JavaScript — це динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду виконуватися на стороні клієнта, таким чином знімаючи навантаження з основного сервера на якому і виконується основний функціонал системи. Сьогодні існує велика кількість javascript-фреймворків, що призначені для полегшення процесу розробки клієнтських скриптів. Дуже швидко набули популярності так звані реактивні фреймворки (reactive frameworks). Вони засновані на парадигмі реактивного програмування, побудованої на потоках даних і розповсюдженні змін. Це означає, що у програмі має бути можливість легко виразити

статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних.

Найпопулярнішими реактивними фреймворками є Vue.JS, React та Angular.

Angular – написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular — це AngularJS, який переосмислили та який був повністю переписаний тією ж командою розробників. Найбільшу популярність використання AngularJS має в Україні. Angular спроектований з переконанням, що декларативне програмування найкраще пасує для побудови інтерфейсів користувача та опису програмних компонентів, в той час як імперативне програмування пасує для опису бізнес-логіки. Фреймворк адаптує та розширює традиційний HTML, щоб забезпечити двосторонню прив'язку даних для динамічного контенту, що дозволяє автоматично синхронізувати модель та вид. У результаті Angular зменшує роль DOM-маніпуляцій з метою підвищення продуктивності і спрощення тестування.

Проведемо порівняльний аналіз наведених фреймворків. Результати аналізу наведено в таблиці 2.6.

Таблиця 2.6 – Порівняльний аналіз реактивних javascript-фреймворків

Фреймворк / Характеристика	Angular	React	Vue.JS
Стабільність	+	+	+
Підтримка розробки	Google	Facebook	Laravel, Alibaba
Документація	+	+	+
Простота вивчення	-	+ / -	+
Інтеграція з іншими популярними фреймворками	+	+	+
Розмір	566K	139K	58K
Повторне використання коду	+	-	+
Швидкість розробки	Повіл	Норма	Швидк

	ьно	льно	о
Реактивність	+ / -	+	+
Компоненто-орієнтованість	+	+	+

Враховуючи виявлені переваги та недоліки проаналізованих засобів для створення веб-застосунків, було вибрано такі технології

- Spring Framework;
- СУБД PostgreSQL;
- фреймворк Angular;
- мови розмітки сторінок та стилей HTML5 та CSS3.

2.3.2 Розробка фізичної моделі бази даних

Фізична модель даних описує дані засобами конкретної СУБД, в даному випадку PostgreSQL, а також визначає розміри атрибутів та обов'язковість їх заповнення. У ході виконання роботи було побудовано фізичну модель бази даних веб-додатку для ранжування резюме в рекрутингу, яка представлена на рисунку 2.17.

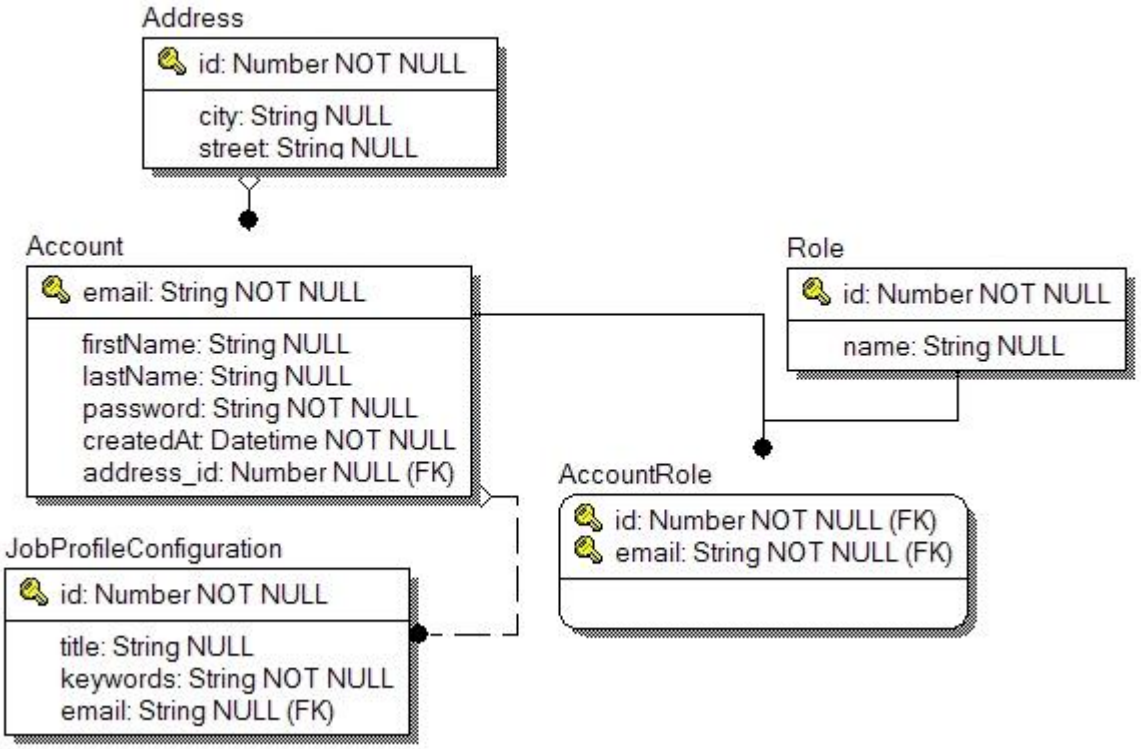

Рисунок 2.17 – Фізична модель бази даних

Таблиці фізичної моделі наведенні в таблицях 2.7-2.11.

Таблиця 2.7 - Таблиця «Адреса»

Поле	Тип поля	Клю	Розмі	Обов.

		ч	р поля	заповнення
Ідентифікатор	Число	PK	Лічильник	Not null
Місто	Строка		Коротка строка	null
Вулиця	Строка		256	null

Таблиця 2.8 - Таблиця «Роль»

Поле	Тип поля	Ключ	Розмір поля	Обов. заповнення
Ідентифікатор	Число	PK	Лічильник	Not null
Найменування	Строка		Коротка строка	null

Таблиця 2.9 - Таблиця «Акаунт»

Поле	Тип поля	Ключ	Розмір поля	Обов. заповнення
Електронна пошта	Строка	PK	Коротка строка	Not null
Ім’я	Строка		30	Null
Прізвище	Строка		30	Null
Пароль	Строка		30	Not null
Дата створення	Дата		Дата	Not null
Ідентифікатор адреси	Число	FK	Довге ціле	null

Таблиця 2.10 - Таблиця «АкаунтРоль»

Поле	Тип поля	Ключ	Розмір поля	Обов. заповнення
Ідентифікатор акаунта	Число	FK, PK	Довге ціле	Not null
Ідентифікатор ролі	Число	FK, PK	Довге ціле	Not null

Таблиця 2.11 - Таблиця «Профіль роботи»

Поле	Тип поля	Ключ	Розмір поля	Обов. заповнення
Ідентифікатор	Число	PK	Лічильник	Not null
Найменування	Строка		Коротка строка	null

Ключові слова	Строка		Довга строка	Not null
Електронна пошта	Строка	FK	Коротка строка	Not null

2.3.3 Діаграма компонентів

Для розробки було обрано архітектурний стиль MVC. Модель-вигляд-контролер - архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

Діаграма компонент — в UML, діаграма, на якій відображаються компоненти, залежності та зв'язки між ними. Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути представлено як компоненту. Деякі компоненти існують під час компіляції, деякі — під час компонування, а деякі під час роботи програми. Компоненти об'єднуються, разом використовуючи структурні зв'язки, щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер». Структурна взаємодія — «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту». При використанні діаграми компонент, щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Надамо діаграму компонентів, представлену на рисунку 2.18, та специфікацію компонентів ПЗ.

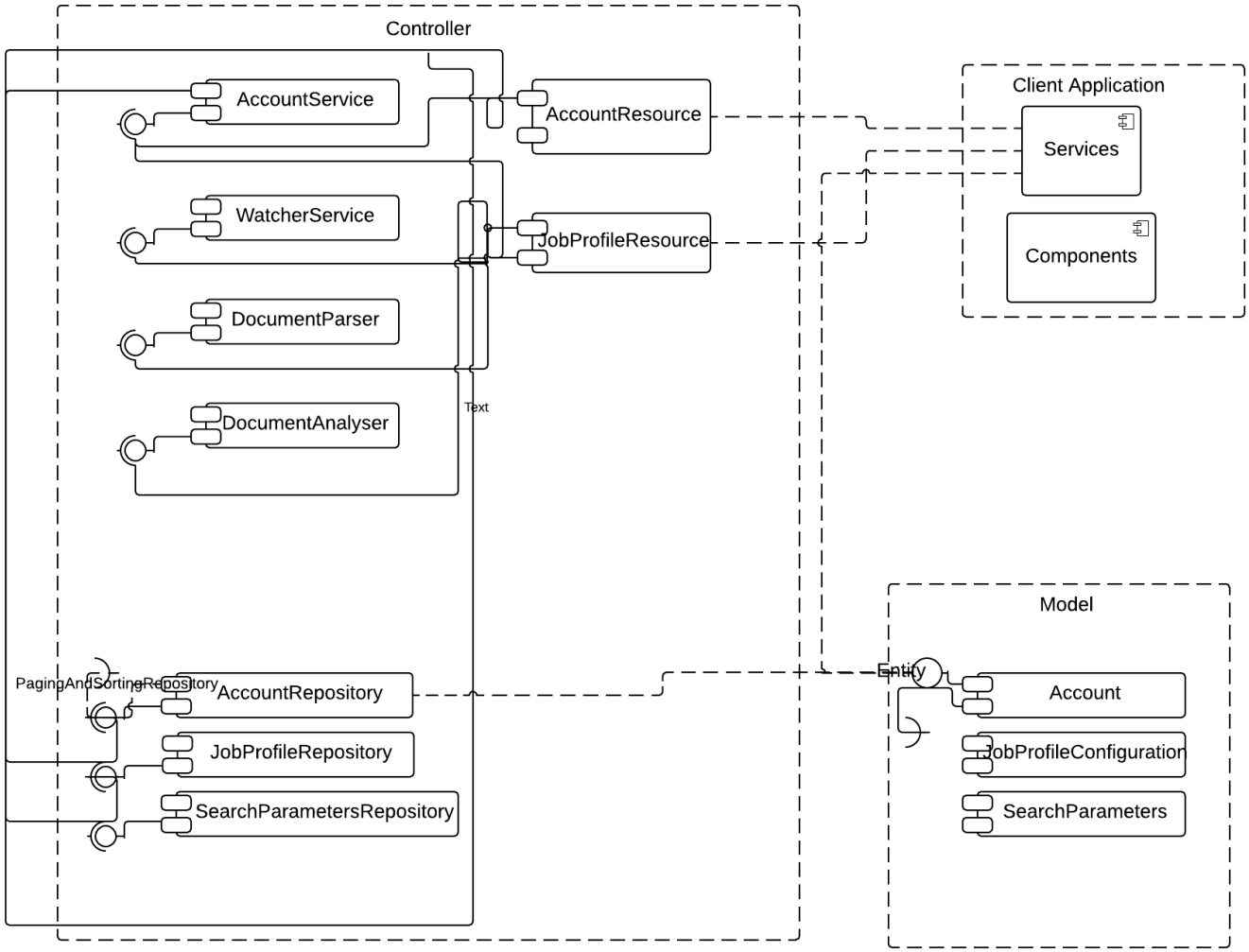

Рисунок 2.18 – Діаграма компонентів веб-додатку для ранжування резюме в рекрутингу

2.3.4 Діаграма розгортання

Модель реалізації визначає розміщення даних, методи доступу і техніку індексування. Модель реалізації задається діаграмами реалізації. Діаграми реалізації призначені для відображення складу компільованих і виконуваних модулів системи, а так само зв'язків між ними. Діаграми реалізацій поділяються на два конкретні види: діаграми компонентів і діаграми розгортання.

Діаграма розгортання показує робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент. Компоненти, що не мають представлення під час роботи програми на діаграмі розгортання не представлені, натомість, вони відображені на діаграмі компонентів.

Діаграма розгортання веб-додатку для ранжування резюме в рекрутингу зображена на рисунку 2.19.

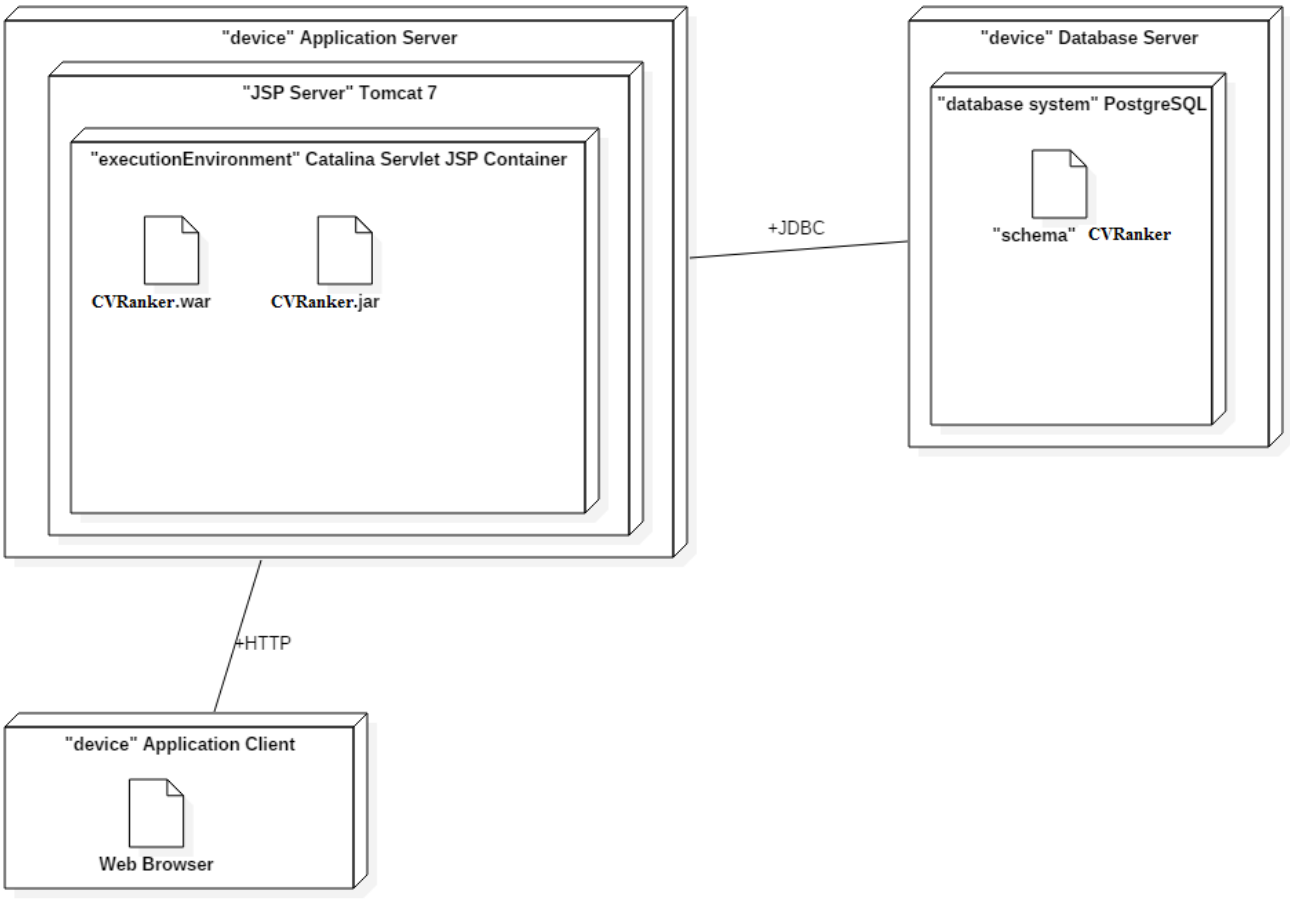

Рисунок 2.19 - Діаграма розгортання веб-додатку для ранжування резюме в рекрутингу

2.3.5 Реалізація програмного забезпечення

Програмне забезпечення було реалізовано за допомогою мови програмуваня Java, використовуючи IDE Intellij Idea та локальний сервер Tomcat. Результати розробки показані на прикладі основних класів сервера. Код основних класів представлений у лістингах 1 та 2. Лістинг програми надано в додатку Б «Тексти програмних модулів».

Лістинг 1 – DocumentParser.java

```
package io.github.tkaczenko.cvranker.service;

import org.apache.poi.hwpf.extractor.WordExtractor;
import org.apache.poi.openxml4j.exceptions.OpenXML4JException;
import org.apache.poi.openxml4j.opc.OPCPackage;
import org.apache.poi.poifs.filesystem.POIFSFileSystem;
import org.apache.poi.xwpf.extractor.XWPFWordExtractor;
import org.apache.tika.exception.TikaException;
import org.apache.tika.metadata.Metadata;
import org.apache.tika.parser.ParseContext;
```

```
import org.apache.tika.parser.pdf.PDFParser;
import org.apache.tika.parser.txt.TXTParser;
import org.apache.tika.sax.BodyContentHandler;
import org.apache.xmlbeans.XmlException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import org.xml.sax.SAXException;

import java.io.IOException;
import java.io.InputStream;

@Component
public class DocumentParser {
private static final Logger LOGGER =
LoggerFactory.getLogger(DocumentParser.class);

public String extractTextFromPdf(InputStream documentInputStream) throws
IOException, TikaException, SAXException {
            BodyContentHandler handler = new BodyContentHandler();
            Metadata metadata = new Metadata();
            ParseContext pcontext = new ParseContext();

            PDFParser pdfparser = new PDFParser();
            pdfparser.parse(documentInputStream, handler, metadata, pcontext);
            return handler.toString();
        }

public String extractTextFromTxt(InputStream documentInputStream) throws
IOException,TikaException, SAXException{
            BodyContentHandler handler = new BodyContentHandler();
            Metadata metadata = new Metadata();
            ParseContext pcontext=new ParseContext();

            TXTParser TexTParser = new TXTParser();
            TexTParser.parse(documentInputStream, handler, metadata,pcontext);
            return handler.toString();
        }

        public String extractTextFromDocx(InputStream documentInputStream) {
            if (documentInputStream == null) {
                return "";
            }
            String extractedText;
            XWPFWordExtractor wordExtractor = getDocxExtractor(documentInputStream);
            extractedText = (wordExtractor == null ? "" : wordExtractor.getText());
            try {
```

```
                documentInputStream.close();
            } catch (IOException e) {
                LOGGER.warn("Couldn't close InputStream", e);
            }
            return extractedText;
        }

        public String extractTextFromDoc(InputStream documentInputStream) {
            if (documentInputStream == null) {
                return "";
            }
            WordExtractor wordExtractor = getDocExtractor(documentInputStream);
            String extractedText = (wordExtractor == null ? "" :
wordExtractor.getText());
            try {
                documentInputStream.close();
            } catch (IOException e) {
                LOGGER.warn("Couldn't close InputStream", e);
            }
            return extractedText;
        }

    private XWPFWordExtractor getDocxExtractor(InputStream documentInputStream) {
            try {
                OPCPackage opcPackage = OPCPackage.open(documentInputStream);
                return new XWPFWordExtractor(opcPackage);
            } catch (IOException | OpenXML4JException | XmlException e) {
                LOGGER.warn("Cannot create extractor", e);
            }
            LOGGER.warn("XWPFWordExtractor not created");
            return null;
        }

        private WordExtractor getDocExtractor(InputStream documentInputStream) {
            try {
                POIFSFileSystem fileSystem = new
POIFSFileSystem(documentInputStream);
                return new WordExtractor(fileSystem);
            } catch (IOException e) {
                LOGGER.warn("Cannot create extractor", e);
            }
            LOGGER.warn("WordExtractor not created");
            return null;
        }
    }
```

Лістинг 2 – WatcherService.java

```
package io.github.tkaczenko.cvranker.service;

import io.github.tkaczenko.cvranker.WatcherConfig;
import org.apache.commons.io.FilenameUtils;
import org.apache.commons.io.IOUtils;
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.document.StringField;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.index.Term;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.tika.exception.TikaException;
import org.jsoup.helper.Validate;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.xml.sax.SAXException;
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.ArrayList;
import java.util.List;

public class WatcherService implements Runnable {
private static final Logger LOGGER =
LoggerFactory.getLogger(WatcherService.class);

    private final DocumentParser parser;
    private final WatchKey watchKey;
    private final List<String> fileNames = new ArrayList<>();

    private final String path;
    private final String index;

public WatcherService(DocumentParser parser, WatcherConfig config) throws
IOException, InterruptedException {
        this.parser = parser;
        this.path = config.getPath();
        this.index = config.getIndex();
        Path dir = Paths.get(path);
        WatchService watchService = dir.getFileSystem().newWatchService();
```

```
            dir.register(watchService, StandardWatchEventKinds.ENTRY_CREATE,
StandardWatchEventKinds.ENTRY_DELETE);
            watchKey = watchService.take();
        }

        @Override
        public void run() {
            while (true) {
                List<WatchEvent<?>> watchEvents = watchKey.pollEvents();
                watchEvents.forEach(watchEvent -> {
                    if
(StandardWatchEventKinds.ENTRY_CREATE.equals(watchEvent.kind())) {
                        LOGGER.info("Created " + watchEvent.context().toString());
                        fileNames.add(watchEvent.context().toString());
                    } else if
(StandardWatchEventKinds.ENTRY_DELETE.equals(watchEvent.kind())) {
                        LOGGER.info("Deleted " + watchEvent.context().toString());
                    }
                });
                if (fileNames.size() > 0) {
                    try {
                        addDocument(fileNames);
                    } catch (IOException e) {
                        LOGGER.warn("Cannot open index directory", e);
                    }
                    fileNames.clear();
                }
            }
        }

        private void addDocument(List<String> fileNames) throws IOException {
            LOGGER.info("Indexing to directory '{}'", new Object[]{index});
            Directory dir = FSDirectory.open(Paths.get(index));
            Analyzer analyzer = new StandardAnalyzer();
            IndexWriterConfig iwc = new IndexWriterConfig(analyzer);

            iwc.setOpenMode(IndexWriterConfig.OpenMode.CREATE_OR_APPEND);

            IndexWriter writer = new IndexWriter(dir, iwc);
            fileNames.forEach(fileName -> {
                String ext = FilenameUtils.getExtension(fileName);
                String filePath = path + fileName;
                String content = null;
                try {
                    switch (ext) {
                        case "pdf":
```

```
                                content = parser.extractTextFromPdf(new
FileInputStream(filePath));
                                break;
                            case "docx":
                                content = parser.extractTextFromDocx(new
FileInputStream(filePath));
                                break;
                            case "doc":
                                content = parser.extractTextFromDoc(new
FileInputStream(filePath));
                                break;
                            case "txt":
                                content = parser.extractTextFromTxt(new
FileInputStream(filePath));
                        }
                    } catch (IOException | TikaException | SAXException e) {
                        LOGGER.warn("Cannot extract file", e);
                    }
                    Validate.notNull(content, "Cannot extract content of file " +
fileName);
          InputStream inputStream = IOUtils.toInputStream(content,
StandardCharsets.UTF_8);
                    try {
                        indexDoc(writer, filePath, fileName, inputStream);
                    } catch (IOException e) {
                        LOGGER.warn("Cannot indexing file", e);
                    }
                });
                writer.close();
            }

          private void indexDoc(IndexWriter writer, String path, String fileName,
InputStream content) throws IOException {
                // make a new, empty document
                Document doc = new Document();

                // Add the path of the file as a field named "path".  Use a
                // field that is indexed (i.e. searchable), but don't tokenize
                // the field into separate words and don't index term frequency
                // or positional information:
                Field pathField = new StringField("path", path, Field.Store.YES);
                doc.add(pathField);

                Field nameField = new StringField("name", fileName, Field.Store.YES);
                doc.add(nameField);

                // Add the last modified date of the file a field named "modified".
```

```
                // Use a LongField that is indexed (i.e. efficiently filterable with
                // NumericRangeFilter).  This indexes to milli-second resolution, which
                // is often too fine.  You could instead create a number based on
                // year/month/day/hour/minutes/seconds, down the resolution you require.
                // For example the long value 2011021714 would mean
                // February 17, 2011, 2-3 PM.
                // doc.add(new LongField("modified", lastModified, Field.Store.NO));

      // Add the contents of the file to a field named "contents".  Specify a Reader,
      // so that the text of the file is tokenized and indexed, but not stored.
                // Note that FileReader expects the file to be in UTF-8 encoding.
                // If that's not the case searching for special characters will fail.
                doc.add(new TextField("content", new BufferedReader(new
InputStreamReader(content, StandardCharsets.UTF_8))));

      if (writer.getConfig().getOpenMode() ==
IndexWriterConfig.OpenMode.CREATE_OR_APPEND) {
      // New index, so we just add the document (no old document can be there):
      LOGGER.info("Adding new index for document {}", new Object[]{path + fileName});
                    writer.addDocument(doc);
                } else {
                    // Existing index (an old copy of this document may have been
indexed) so
      // we use updateDocument instead to replace the old one matching the exact
                    // path, if present:
                    writer.updateDocument(new Term("path", path), doc);
                }

            }
      }
```

2.3.6 Тестування програмного забезпечення

Для тестування програмного забезпечення веб-додатку для ранжування резюме мінімальним об’єктом для тестування є клас, адже програмне забезпечення є об’єктно-орієнтованим. В якості тестового середовища буде використано фреймворк JUnit для створення тест-класів. Класи тестуються окремо з драйверами. Головний клас тестується разом з вже відтестованими модулями.

Тестування програмного забезпечення для редагування речення проведемо за допомогою прийому тестування методом «чорного ящика» - класів еквівалентності.

Відповідно до даної методики необхідно розбити множину значень вхідних даних на кінцеве число підмножин (які називаються класами еквівалентності), щоб кожний тест, що є представником певного класу, був еквівалентним будь-якому

іншому тесту цього класу. Два тести є еквівалентними, якщо вони виявляють ті самі помилки.

Проектування тестів за методом класів еквівалентності проводиться у два етапи:

- виділення за специфікацією класів еквівалентності;
- побудова множини тестів.

На першому етапі відбувається вибір зі специфікації кожної вхідної умови та розбиття її на дві або більше групи, що відповідають так званим – правильним класам еквівалентності та – неправильним класам еквівалентності, тобто множинам допустимих для програми й недопустимих значень вхідних даних. Цей процес залежить від вигляду вхідної умови.

Для тестування варіанта використання «Аутентифікація» був використаний метод еквівалентної розбивки. Виділені класи еквівалентності, представлені в таблиці 2.12. Результати тестувань представлені в таблицях 2.13 та 2.14.

Таблиця 2.12 – Класи еквівалентності

	Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
	Логін	1. Введено вірний логін	2. Введено неіснуючий логін 3. Логін не було введено
	Пароль	4. Введено вірний пароль	5. Введено неіснуючий пароль 6. Пароль не було введено

Таблиця 2.13 – Тести із правильних класів еквівалентності

№ тесту	№ класу еквівалентності	Вхідні умови	Вихідні данні	Результат
1	1, 4	Логін: admin Пароль:admin47	Перехід до панелі адміністратора	пройдено

Таблиця 2.14 – Тести із неправильних класів еквівалентності

тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
	2,5	Логін: natali Пароль: natali47	Повідомлення про помилку введення даних	Тест пройдено
	3,6	Логін: - Пароль: -	Повідомлення про помилку	Тест пройдено

			введення даних	

Для тестування варіанта використання «Ранжування завантажених резюме» був використаний метод еквівалентної розбивки. Виділені класи еквівалентності, представлені в таблиці 2.15. Результати тестувань представлені в таблицях 2.16 та 2.17.

Таблиця 2.15 – Класи еквівалентності

	Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
	Найменування	1. Введено вірне найменуваня	Відсутні
	Ключові слова	2. Введено ключові слова 3. Введено одне ключове слово	4. Ключові слова не було введено

Таблиця 2.16 – Тести із правильних класів еквівалентності

№ тесту	№ класу еквівалентності	Вхідні умови	Вихідні данні	Результат
1	1, 3	Найменування: новий пошук Ключові слова: java	Відображено список релевантних резюме	пройдено
2	1, 2	Найменування: новий пошук Ключові слова: java, javascript	Відображено список релевантних резюме	пройдено

Таблиця 2.17 – Тести із неправильних класів еквівалентності

тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
	4	Найменування: новий пошук Ключові слова:	Повідомлення про помилку введення даних	Тест пройдено

Для тестування варіанта використання «Додавання / модифікація / видалення профілей роботи для пошуку релевантного співробітника» був використаний метод еквівалентної розбивки. Виділені класи еквівалентності, представлені в таблиці 2.18. Результати тестувань представлені в таблицях 2.19 та 2.20.

Таблиця 2.18 – Класи еквівалентності

	Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
	Найменування	1. Введено вірне найменуваня	Відсутні
	Ключові слова	2. Введено ключові слова 3. Введено одне ключове слово	4. Ключові слова не було введено

Таблиця 2.19 – Тести із правильних класів еквівалентності

№ тесту	№ класу еквівалентності	Вхідні умови	Вихідні данні	Результат
1	1, 3	Найменування: новий пошук Ключові	Збережено новий / змінений профіль для	пройдено

		слова: java	майбутнього пошуку релевантного резюме	

Продовження Таблиці 2.19 – Тести із правильних класів еквівалентності

2	1, 2	Найменування: новий пошук Ключові слова: java, javascript	Збережено новий / змінений профіль для майбутнього пошуку релевантного резюме	пройдено

Таблиця 2.20 – Тести із неправильних класів еквівалентності

тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
	4	Найменування: новий пошук Ключові слова:	Повідомлення про помилку введення даних	Тест пройдено

Для тестування варіанта використання «Завантаження, видалення резюме» був використаний метод еквівалентної розбивки. Виділені класи еквівалентності, представлені в таблиці 2.21. Результати тестувань представлені в таблиці 2.22.

Таблиця 2.21 – Класи еквівалентності

	Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
	Найменування файлу	1. Обрано найменування файлу	Відсутні

Таблиця 2.22 – Тести із правильних класів еквівалентності

№ тесту	№ класу еквівалентності	Вхідні умови	Вихідні данні	Результат
1	1	Найменування: sample1.pdf	Завантажено обраний файл резюме	пройдено
2	1	Найменування: sample1.pdf	Видалено обраний файл резюме	пройдено

РОЗДІЛ 3. РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБ-ДОДАТКУ ДЛЯ РАНЖУВАННЯ РЕЗЮМЕ В РЕКРУТИНГУ

В процесі виконання дипломної роботи було розроблено програмне забезпечення веб-додатку для ранжування резюме в рекрутингу. Дане програмне забезпечення призначене для використання рекрутерами.

Програмне забезпечення було розроблене мовою програмування Java, яка відповідає всім вимогам до написання веб-додатків. Серед переваг даної мови програмування над іншими хочеться відмітити такі мова Java не залежить від платформи. А також те, що Java - це об'єктно-орієнтована мова.

Під час написання програми були використані такі допоміжні програмні засоби:

– середовище розробки Intellij Idea;
– контейнер сервлетів с відкритим віхідним кодом Tomcat;
– база для збереження даних PostgreSQL;
– технології: HTML5, CSS3, JavaScript;
– фреймворки: Spring, Angular.

До числа основних факторів, що характеризують ефективність даної розробки, відносяться:

– підвищення ефективності роботи рекрутера;
– полегшення доступу до даних;
– простий і інтуїтивно зрозумілий користувальницький інтерфейс.

У процесі виконання дипломної роботи було створене технічне завдання (додаток А), а також такі моделі як:

– модель варіантів використання;
– інформаційна модель;
– статична модель;
– динамічна модель;
– модель реалізації.

Тестування програмного продукту веб-додатку для ранжування резюме в рекрутингу показало, що програмне забезпечення успішно справляється з усіма поставленими перед ним задачами.

Текст програмних модулів наведено у додатку В.

Робота із програмою здійснюється з використанням зручного користувальницького інтерфейсу й розрахована на роботу в режимі діалогу з користувачем, що не є професійним програмістом.

Однією з переваги розробки такого програмного продукту є те, що він представлятиме собою веб додаток. Веб додатки є невеликі за обсягом, швидко завантажуються і швидко відповідають на дії користувачів. Навіть найскладніший додаток завантажується всього за кілька секунд. Користувачеві непотрібно завантажувати на свій комп'ютер увесь продукт цілком, щоб почати з ним працювати. Досить завантажити тільки ту його частину, яка потрібна для виконання конкретної поточної задачі.

Програмний продукт не пред'являє ніяких вимог до апаратної платформи, тому функціонує під управлінням будь-якої операційної системи. Для роботи з продуктом необхідно використовувати сучасний браузер (Mozzilla FireFox, Opera, Google Chrome, Internet Explorer) та мати можливість виходу в мережу Інтернет.

Як результат розробки, на рисунку 3.1, 3.2 зображено головні вікна програмного забезпечення веб-додатку для ранжування резюме.

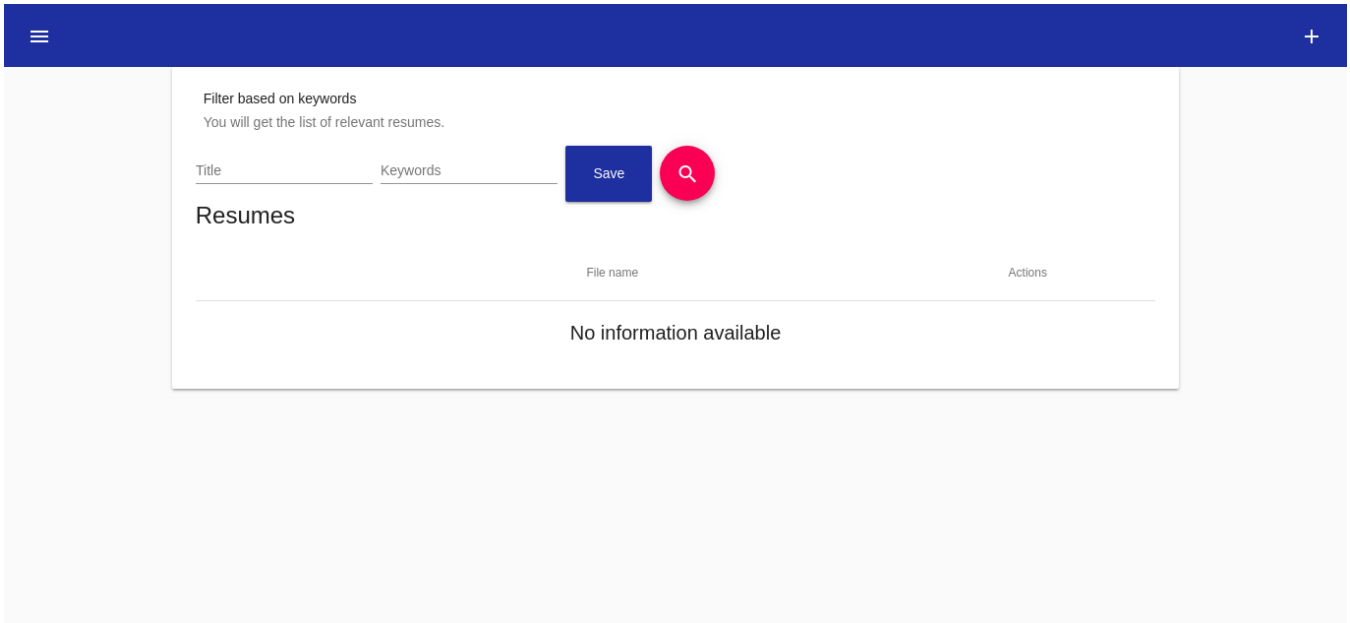

Рисунок 3.1 – Головне вікно з можливістю ранжування резюме, збереження профілю роботи та завантаження файлів резюме із списку результатів

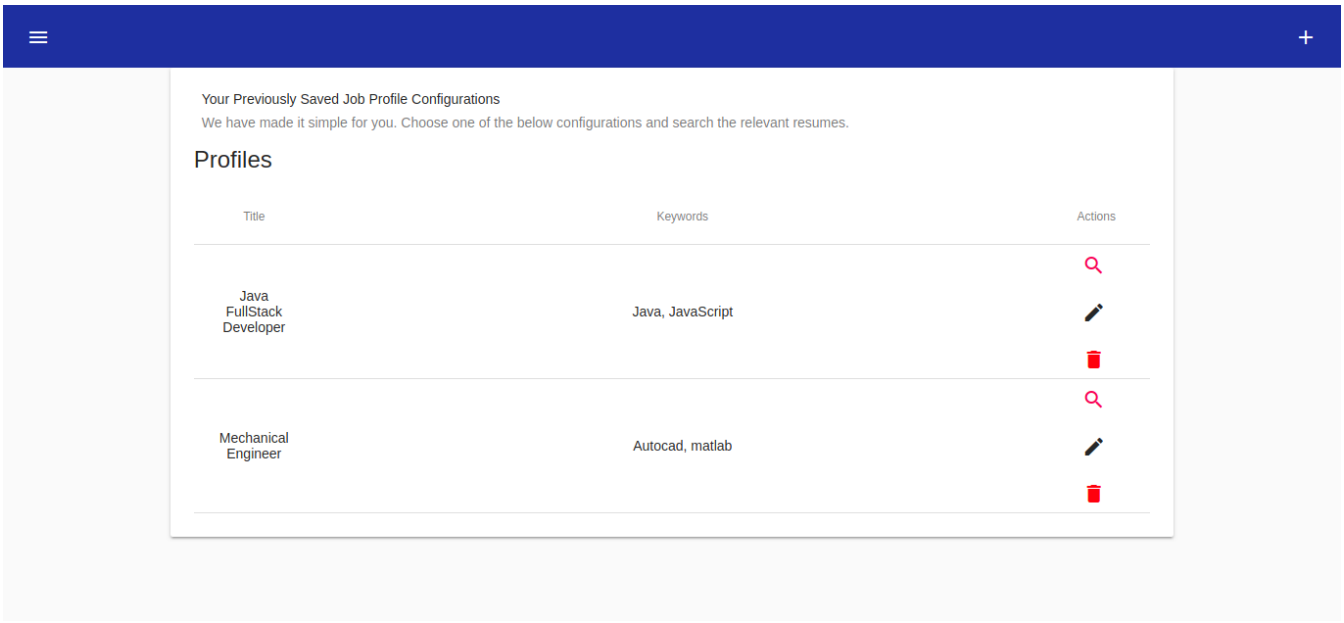

Рисунок 3.2 – Головне вікно з можливістю перегляду, редагуваня збережених профілей роботи та пошуку у відповідності до обраного профілю

4ОХОРОНА ПРАЦІ

Вступ

Охорона праці – система законодавчих актів, соціально-економічних, організаційних, технічних, гігієнічних і лікувально-профілактичних заходів та засобів, що забезпечують безпеку, збереження здоров'я і працездатності людини в процесі праці. Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової праці. Їх праця стала більш інтенсивним, напруженим, які вимагають значних витрат розумової, емоційної і фізичної енергії. Це зажадало комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку. Охорона здоров'я працюючих, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму складає одну з головних турбот людського суспільства.

Звертається увага на необхідність широкого застосування прогресивних форм наукової організації праці, зведення до мінімуму ручної, малокваліфікованої праці, створення обстановки, що виключає професійні захворювання і виробничий травматизм.

Даний розділ дипломного проекту присвячений розгляду наступних питань: організація робочого місця програміста; визначення оптимальних умов праці програміста. Опис робочого місця програміста Робоче місце - це частина простору, в якому інженер здійснює трудову діяльність, і проводить велику частину робочого часу. Робоче місце, добре пристосоване до трудової діяльності інженера, правильно і доцільно організоване, щодо простору, форми, розміру забезпечує йому зручне положення при роботі і високу продуктивність праці при найменшому фізичному і психічному напруженні.

При правильній організації робочого місця продуктивність праці зростає з 8 до 20 відсотків. Відповідно до ГОСТ 12.2.032-78 конструкція робочого місця і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам [8].

Велике значення має також характер роботи. Зокрема, при організації робочого місця програміста повинні бути дотримані наступні основні умови: оптимальне розміщення устаткування, що до складу робочого місця; достатній робочий простір,

що дозволяє здійснювати всі необхідні рухи і переміщення; необхідно природне і штучне освітлення для виконання поставлених завдань.

Організація роботи полягає у виборі та формуванні такої структури управління охороною праці на підприємстві, яка найповніше відповідала б меті створення безпечних і сприятливих умов праці.

Об’єктом управління охороною праці є діяльність функціональних служб та структурних підрозділів підприємства щодо забезпечення безпечних і здорових умов праці на робочих місцях, виробничих ділянках, у цехах та підприємстві в цілому.

Організація й координація робіт у галузі охорони праці повинні передбачати формування органів управління охороною праці, встановлення обов’язків і порядку взаємодії осіб, які беруть участь в управлінні, а також у прийнятті та реалізації управлінських рішень. Управління охороною праці на підприємстві здійснює власник підприємства (роботодавець), а у структурних підрозділах — відповідні керівники підрозділів.

Зобов’язання, права та відповідальність посадових осіб за виконання покладених на них функцій щодо питань охорони праці розглядаються в посадових інструкціях, форма яких розроблена Держнагляд охорони праці. Згідно зі ст. 13 Закону України «Про охорону праці» роботодавець зобов’язаний створити в кожному структурному підрозділі та на робочому місці умови праці відповідно до вимог нормативних актів, а також забезпечити додержання прав працівників, гарантованих законодавством про охорону праці. З цією метою він забезпечує функціонування системи управління охороною праці, для чого:

- створює відповідні служби і призначає посадових осіб, які забезпечують функціонування системи охорони праці;
- розробляє за участю сторін колективного договору й реалізує комплексні заходи для досягнення встановлених нормативів з охорони праці, впроваджує прогресивні досягнення з охорони праці;
- забезпечує усунення причин, що призводять до нещасних випадків, професійних захворювань, і виконання профілактичних заходів, визначених комісіями за підсумками розслідування цих причин;
- розробляє і затверджує положення, інструкції, інші нормативні акти про охорону праці, що діють у межах підприємства;

- здійснює постійний контроль за додержанням працівниками технологічних процесів, правил та вимог щодо охорони праці;
- організовує пропаганду безпечних методів праці й співробітництво з працівниками у галузі охорони праці.

У випадку відсутності в нормативних актах про охорону праці вимог, які необхідно виконати для забезпечення безпечних і нешкідливих умов праці на певних роботах, власник зобов'язаний вжити заходів, що забезпечать безпеку працівників.

2.4 Аналіз шкідливих та небезпечних факторів

При організації умов праці необхідно враховувати вплив на працюючих небезпечних і шкідливих виробничих факторів, які можуть привести до травми або іншого раптового різкого погіршення здоров'я та захворювання або зниження працездатності.

Небезпечні і шкідливі виробничі фактори підрозділяються по природі дії на чотири групи: фізичні, хімічні, біологічні і психофізіологічні [9].

До небезпечних фізичних факторів відносяться: машини і механізми, що рухаються; різні підйомно-транспортні пристрої і переміщувані вантажі; незахищені рухливі елементи виробничого устаткування (приводні і передавальні механізми, різальні інструменти, пристосування, що обертаються і переміщуються й ін.); відлітаючи частки оброблюваного матеріалу та інструменту, електричний струм, підвищена температура поверхонь устаткування й оброблюваних матеріалів та ін.

Шкідливими для здоров'я фізичними факторами є: підвищена чи знижена температура повітря робочої зони; висока вологість і швидкість руху повітря; підвищені рівні шуму, вібрації, ультразвуку і різних випромінювань – теплових, іонізуючих, електромагнітних, інфрачервоних і ін. До шкідливих фізичних факторів відносяться також запиленість і загазованість повітря робочої зони; недостатня освітленість робочих місць, проходів і проїздів; підвищена яскравість світла і пульсація світлового потоку.

Хімічні небезпечні і шкідливі виробничі фактори за характером дії на організм людини підрозділяються на наступні підгрупи: загальнотоксичні, подразнюючі, сенсибілізуючі (ті, що викликають алергійні захворювання), канцерогенні (ті, що викликають розвиток пухлин), мутагені (ті, що діють на статеві клітини організму). У цю групу входять численні пари і гази: пари бензолу і толуолу, окис вуглецю,

сірчистий ангідрид, окисли азоту, аерозолі свинцю та ін., токсичні пили, що утворюються, наприклад, при обробці різанням берилію, свинцюватих бронз і латуней і деяких пластмас зі шкідливими наповнювачами. До цієї групи відносяться агресивні рідини (кислоти, луги), що можуть заподіяти хімічні опіки шкіряного покриву при зіткненні з ними.

До біологічних небезпечних і шкідливих виробничих факторів відносяться мікроорганізми (бактерії, віруси й ін.) та макроорганізми (рослини і тварини), вплив яких на працюючих викликає травми або захворювання.

До психофізіологічних небезпечних і шкідливих виробничих факторів відносяться фізичні перевантаження (статичні та динамічні) і нервово-психічні перевантаження (розумові перенапруги, перенапруга аналізаторів слуху, зору та ін.).

Між шкідливими і небезпечними виробничими факторами спостерігається визначений взаємозв'язок. У багатьох випадках наявність шкідливих факторів сприяє прояву травмонебезбечних факторів. Наприклад, надмірна вологість у виробничому приміщенні і наявність струмопровідного пилу (шкідливі фактори) підвищують небезпеку ураження людини електричним струмом (небезпечний фактор).

Рівні впливу на працюючих шкідливих виробничих факторів нормовані гранично-допустимими рівнями, значення яких зазначені у відповідних стандартах системи стандартів безпеки праці і санітарно-гігієнічних правил.

Гранично припустиме значення шкідливого виробничого фактора (за ДСТУ 12.0.002-80) – це граничне значення величини шкідливого виробничого фактора, вплив якого при щоденній регламентованій тривалості протягом усього виробничого стажу не приводить до зниження працездатності і захворювання як у період трудової діяльності, так і до захворювання в наступний період життя, а також несприятливо не впливає на здоров'я потомства.

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності працівників розумової праці. Їхня праця стала більш інтенсивною, напруженою, потребуючою значних витрат розумової, емоційної і фізичної енергії. Це зажадало комплексного рішення проблем ергономіки, гігієни й організації праці, регламентації режимів праці і відпочинку.

На даний час комп'ютерна техніка широко застосовується у всіх галузях діяльності людини. При роботі з комп'ютером людина піддається впливу низки небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон

радіочастот: ВЧ, УВЧ і СВЧ), інфрачервоного та іонізуючого випромінювань, шуму і вібрації, статичної електрики та ін. [9, 10].

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і досить великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ. Велике значення має раціональна конструкція і розташування елементів робочого місця, що важливо для підтримки оптимальної робочої пози людини-оператора.

У процесі роботи з комп'ютером необхідно дотримуватися правильного режиму праці і відпочинку. В противному випадку в персоналу відзначається значна напруга зорового апарату з появою скарг на незадоволеність роботою, головні болі, дратівливість, порушення сну, утому і хворобливі відчуття в очах, у попереку, в області шиї і руках.

Розглянемо шкідливі виробничі фактори, які спостерігаються в приміщеннях, де здійснюється робота з електронно-обчислювальною технікою.

Правильно спроектоване і виконане виробниче освітлення поліпшує умови зорової роботи, знижує стомлюваність, сприяє підвищенню продуктивності праці, благотворно впливає на виробниче середовище, створюючи позитивний психологічний вплив на працюючого, підвищує безпеку праці і знижує травматизм.

Недостатність освітлення приводить до напруги зору, послабляє увагу, приводить до передчасної стомленості. Надмірно яскраве освітлення викликає осліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Усі ці причини можуть привести до нещасливого випадку або профзахворювання, тому настільки важливий правильний розрахунок освітленості.

Існує три види освітлення – природне, штучне і сполучене (природне і штучне разом) [9, 10].

Природне освітлення – освітлення приміщень денним світлом, яке проникає через світлові прорізи в зовнішніх конструкціях приміщень. Природне освітлення характеризується тим, що міняється в широких межах в залежності від часу дня, пори року, характеру області та низки інших факторів.

Штучне освітлення застосовується при роботі в темний час доби і вдень, коли не вдається забезпечити нормовані значення коефіцієнта природного освітлення (похмура погода, короткий світловий день). Освітлення, при якому недостатнє по

нормах природне освітлення доповнюється штучним, називається сполученим освітленням.

Штучне освітлення підрозділяється на робоче, аварійне, евакуаційне, охоронне. Робоче освітлення, в свою чергу, може бути загальним чи комбінованим. Загальне – освітлення, при якому світильники розміщуються у верхній зоні приміщення чи рівномірно стосовно до розташування устаткування. Комбіноване – освітлення, при якому до загального додається місцеве освітлення.

Згідно до Сніп II-4-79 у приміщеннях обчислювальних центрів необхідно застосувати систему комбінованого освітлення.

При виконанні робіт категорії високої зорової точності (найменший розмір об'єкта розрізнення 0,3...0,5 мм) величина коефіцієнта природного освітлення (КЕО) повинна бути не нижче 1,5%, а при зоровій роботі середньої точності (найменший розмір об'єкта розрізнення 0,5...1,0 мм) КЕО повинний бути не нижче 1,0%. В якості джерела штучного освітлення зазвичай використовуються люмінесцентні лампи типу ЛБ або ДРЛ, які попарно поєднуються у світильники, та повинні розташовуватися над робочими поверхнями рівномірно [9].

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери, наступні: при виконанні зорових робіт високої точності загальна освітленість повинна складати 300 лк, а комбінована – 750 лк; аналогічні вимоги при виконанні робіт середньої точності – 200 і 300 лк відповідно.

Крім того усе поле зору повинно бути освітлене досить рівномірно – це основна гігієнічна вимога. Іншими словами, ступінь освітлення приміщення і яскравість екрана комп'ютера повинні бути приблизно однаковими, тому що яскраве світло в районі периферійного зору значно збільшує напруженість очей і, як наслідок, приводить до їх швидкої стомлюваності.

Параметри мікроклімату можуть мінятися в широких межах, в той час як необхідною умовою життєдіяльності людини є підтримка сталості температури тіла завдяки терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище. Принцип нормування мікроклімату – створення оптимальних умов для теплообміну тіла людини з навколишнім середовищем.

Обчислювальна техніка є джерелом істотного тепловиділення, яке може привести до підвищення температури і зниження відносної вологості в приміщенні.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися визначені параметри мікроклімату. У санітарних нормах СН-245-71 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (таблиця 3.1).

Таблиця 3.1 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	22…24°С 40…60% до 0,1м/с
Теплий	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	23…25°С 40…60% 0,1…0,2м/с

Шумом називають усякий несприятливо діючий на людину звук. З фізичної точки зору звук являє собою механічні коливання пружного середовища [10].

Слуховий орган людини сприймає у вигляді чутного звуку коливання пружного середовища, які мають частоту приблизно від 20 до 20000 Гц, але найбільш важливий для слухового сприйняття інтервал від 45 до 10000 Гц.

Сприйняття людиною звуку залежить не тільки від його частоти, але і від інтенсивності і звукового тиску.

Несприятлива дія шуму на людину залежить не тільки від рівня звукового тиску, але і від частотного діапазону шуму, а також від рівномірності впливу протягом робочого часу.

У результаті несприятливого впливу шуму на працюючу людину відбувається зниження продуктивності праці, збільшується брак у роботі, створюються передумови до виникнення нещасливих випадків. Усе це обумовлює велике оздоровче й економічне значення заходів щодо боротьби із шумом.

У таблиці 3.2 зазначені граничні рівні звуку в залежності від категорії ваги і напруженості праці, які є безпечними у відношенні збереження здоров'я і працездатності.

Таблиця 3.2 – Граничні рівні звуку, дБ, на робочих місцях.

Категорія напруженості праці	Категорія важкості праці			
	I Легка	II Середня	III Важка	IV Дуже важка
I Мало напружена	80	80	75	75
II Помірковано напружена	70	70	65	65
III Напружена	60	60	–	–
IV Дуже напружена	50	50	–	–

Електромагнітне й іонізуюче випромінювання.

Більшість учених вважає, що як короткочасний, так і тривалий вплив усіх видів випромінювання від екрана монітора не небезпечний для здоров'я персоналу, що обслуговує комп'ютери. Однак вичерпних даних щодо небезпеки впливу випромінювання від моніторів на працюючих з комп'ютерами не існує і дослідження в цьому напрямку продовжуються [8, 9].

Припустимі значення параметрів неіонізуючих електромагнітних випромінювань від монітора комп'ютера представлені в таблиці 3.3.

Максимальний рівень рентгенівського випромінювання на робочому місці оператора комп'ютера зазвичай не перевищує 10мкбер/год, а інтенсивність ультрафіолетового й інфрачервоного випромінювань від екрана монітора знаходиться в межах 10…100мВт/м^2.

Таблиця 3.3 – Припустимі значення параметрів неіонізуючих електромагнітних випромінювань (відповідно до СанПіН 2.2.2.542-96)

Найменування параметра	Припустимі значення
Напруженість електричної складової електромагнітного поля або на відстані 50см від поверхні	10В/м
Напруженість магнітної складової електромагнітного поля або на відстані 50 см від поверхні відеомонітора	0,3А/м
Напруженість електростатичного поля не повинна перевищувати: для дорослих користувачів для дітей дошкільних установ і учнів середніх спеціальних і вищих навчальних закладів	 20 кв/м 15 кв/м

2.5 Розробка заходів щодо забезпечення сприятливих умов праці при роботі з персональним комп'ютером

Сьогодні фахівці в області ергономіки вже зрозуміли, що не можна знайти ідеальне положення, у якому можна перебувати і працювати протягом усього робочого дня. Для більшості людей комфортабельним робочим місцем повинне бути таке, яке можна пристосувати не менш чим для двох позицій, при цьому положення крісла, монітора повинні щораз відповідати виконуваній роботі і звичкам. Багато хто вважають, що для роботи на комп'ютері більше підходять вертикальне і трохи похиле положення. Можливо, щоб крісло було злегка нахилене вперед.

Дисплей. Положення тіла звичайно відповідає напрямку погляду; дисплеї, розташовані занадто низько чи під неправильним кутом, є основними причинами сутулості. Відстань від дисплея до очей може варіюватися в залежності від характеру виконуваної роботи - 40-70 см. При роботі з текстом відстань від екрана до очей повинна лише небагато перевищувати відстань між книгою й очима і складати 40- 45 см. Необхідно давати відпочинок очам і час від часу їх просто закривати. Не можна допускати, щоб очі увесь час були сфокусовані на одній відстані, працюючи з текстом, рекомендується установлювати великий шрифт. Дуже важливо, щоб екран монітора не мерехтів. Частота регенерації зображення повинна бути не менш 72 Гц,

тому що при більш низькій частоті помітне мерехтіння, хоча люди з особливо чуттєвим зором можуть помітити його і при більш високій частоті. Їм доведеться підібрати графічну плату і монітор з частотою регенерації 85 Гц і вище.

Рекомендується встановлювати на екран монітора спеціальні скляні поляризаційні фільтри з заземленням. Необхідно уникати того, щоб термінал був звернений екраном убік вікна, оскільки інтенсивна освітленість області зору може затопити потоками світла очі і розмити зображення на сітківці. Якщо приходиться сидіти поруч з вікном, то треба розташуватися під прямим кутом до нього, причому екран дисплея повинний бути перпендикулярний шибці – цим виключаються відблиски на екрані.

Для зниження впливу низькочастотного електромагнітного випромінювання монітора на ЭЛТ рекомендується вибирати монітор, що задовольняє стандарту MPR II, чи TCO 99. У противному випадку необхідно установити на екран захисний фільтр, найбільш сучасні моделі, затримують до 99 % електромагнітного випромінювання.

Форма спинки крісла повинна повторювати форму спини працюючого. Крісло необхідно установити на такій висоті, щоб не почувався тиск на куприк (крісло розташоване занадто низько) на стегна (крісло розташоване занадто високо). Фахівці з ергономіки, вважали, що кут між стегнами і хребтом повинний складати 90°, однак недавно проведені дослідження показали, що більшість людей переважно сидять трохи відкинувшись.

Клавіатуру необхідно установити так, щоб не треба було до неї тягтися і нахилятися вперед. При зміні положення тіла (наприклад, з вертикального на похиле) обов'язково треба перемінити положення клавіатури і дисплея. Може виявитися корисною регульована підставка клавіатури, завдяки якій можна без усякої напруги працювати з маніпулятором "миша". Можна поставити клавіатуру і на коліна. Руки при роботі повинні бути прямими в зап'ястях і зігнуті в ліктях приблизно під прямим кутом. Пальці також повинні бути злегка зігнуті. Удари по клавішах не повинні бути занадто сильними.

Зручна висота столу особливо важлива в тому випадку, коли на ньому розташовується клавіатура. Якщо в клавіатури немає підставки, а висоту столу не можна змінити (і він занадто високий), то треба вище підняти сидіння крісла, а під

ноги підставити лавочку: чи що-небудь інше. Якщо стіл занадто низький, необхідно підкласти що-небудь під його ніжки.

Якщо при роботі часто приходиться дивитися на документи, треба установити підставку з оригіналом документа в одній площині і на одній, висоті з екраном. Якщо треба частіше дивитися на оригінал, чим на екран, необхідно повернути крісло й екран таким чином, щоб оригінал розміщався прямо перед очима, а комп'ютер ледве збоку.

Щогодини необхідно робити перерву в роботі. У цей час рекомендується робити вправи для зап’ясть. Нижче описаний можливий комплекс вправ.

Покласти руку на край столу долонею вниз. Взявшись, за пальці, іншою рукою відвести кисть назад і утримувати протягом 5 секунд. Повторити для іншої руки.

Злегка упертися рукою в стіл і на 5 секунд напружити пальці в зап'ястя. Повторити іншою рукою.

Сильно стиснути пальці, а потім розпрямити їх. Виконання приведених рекомендацій дозволяє скоротити вплив шкідливих, виробничих: факторів на організм людини.

Відповідно до «Закону про охорону праці» роботодавець зобов'язаний забезпечити:

безпеку працівників при експлуатації устаткування;

застосовування засобів індивідуального захисту працівників;

відповідні вимоги охорони праці, умови праці на кожному робочому місці; дотримання режиму праці і відпочинку працівників;

навчання безпечним методам і прийомам виконання робіт;

інструктаж з охорони праці;

організацію контролю за станом умов праці на робочих місцях;

проведення атестації робочих місць за умовами праці;

інформування працівників про умови й охорону праці на робочих місцях, про існуючий ризик ушкодження здоров'я і компенсаціях при ушкоджені та засобах індивідуального захисту.

Вимоги до освітлення. У приміщеннях, де експлуатуються комп'ютери, штучне освітлення повинне бути загальним і рівномірним. Однак якщо співробітники переважно працюють з документами, то допускається застосувати комбіноване освітлення: крім загального установлюються світильники місцевого

освітлення, які не повинні створювати відблисків на поверхні екрана і збільшувати його освітленість більш 300 лк. Освітленість поверхні столу в зоні розміщення робочого документа повинна становити 300-500 лк.

Джерела освітлення варто встановлювати таким чином, щоб вони не осліплювали, при цьому яскравість світних поверхонь (вікна, світильники та ін.), які розташовуються в полі зору, повинна бути не більш 200 кд/м2.

В якості джерела світла при штучному освітленні повинні застосовуватися переважно люмінесцентні лампи типу ЛБ. При пристрої відбитого освітлення допускається застосування металогалогених ламп потужністю до 250 Вт, а у світильниках місцевого освітлення – ламп накалювання.

Для забезпечення нормованих значень освітленості в приміщеннях необхідно не рідше двох разів на рік чистити скло, віконні рами і світильники та вчасно заміняти перегорілі лампи.

Робочі місця повинні розташовуватися таким чином, щоб природне світло падало збоку, переважно ліворуч.

Для внутрішньої обробки приміщень повинні використовуватися дифузно-відбиваючі матеріали, з коефіцієнтом відбиття від стелі – 0,7-0,8; для стін – 0,5-0,6; для підлоги – 0,3-0,5. Полімерні матеріали для внутрішньої обробки повинні бути дозволені для застосування органами й установами Держсанепіднагляду.

Поверхня підлоги в приміщеннях повинна бути рівною, без вибоїн, неслизькою, зручною для очистки і вологого прибирання, мати антистатичні властивості.

Вимоги до кондиціонування. У виробничих приміщеннях, у яких установлені комп'ютери, мікроклімат повинен відповідати наступним санітарним нормам:

температура повітря в теплий період року – не більш 230-250 С, у холодний– 220-240 С;

відносна вологість повітря – 40-60%;

швидкість руху повітря – 0,1м/с.

Для підвищення вологості повітря в приміщеннях варто застосовувати зволожувачі повітря, щодня заправляти їх дистильованою або кип'яченою водою.

Вимоги до розміщення, оснащення й організації робочих місць. Відстань між

робочими столами з моніторами (у напрямку тилу поверхні одного монітора та екрана іншого) повинне бути не менш 2 м, а між бічними поверхнями моніторів – не менш 1,2 м.

Віконні отвори повинні бути обладнані регульованими жалюзями, завісами, зовнішніми навісами та ін.

Бажано, щоб висоту робочої поверхні столу можна було регулювати в межах 680-800 мм, а при відсутності такої можливості вона повинна дорівнювати 725 мм. Модульними розмірами робочої поверхні комп'ютерного столу, на базі яких розраховують конструктивні розміри, варто вважати: ширину 800, 1000, 1200 і 1400 мм; глибину 800 і 1000 мм.

Екран монітора повинен знаходитись від очей користувача на оптимальній відстані 600-700 мм, але не ближче 500 мм з урахуванням розмірів алфавітно-цифрових знаків і символів.

Вимоги до безпеки під час експлуатації й обслуговування ЕОМ. У приміщенні з комп'ютерами повинно проводитися щоденне вологе прибирання. Приміщення повинні оснащатися аптечкою першої допомоги та вуглекислотними вогнегасниками.

Режими праці і відпочинку. Режими праці і відпочинку при роботі на комп'ютерах залежать від виду і категорії трудової діяльності.

При восьмигодинній робочій зміні і роботі на комп'ютері регламентовані перерви варто встановлювати:

для I категорії робіт – через 2 години від початку робочої зміни і через 2 години після обідньої перерви тривалістю по 15 хв;

для II категорії робіт – через 2 години від початку робочої зміни і через 1,5 - 2 години після обідньої перерви тривалістю по 15 хв або через кожну годину роботи тривалістю по 10 хв.

Під час регламентованих перерв із метою зниження нервово-емоційної напруги, зменшення стомлення очей, усунення гіподинаміки і гіпокінезії доцільно виконувати комплекси вправ, викладених у санітарних нормах.

Після проведення дослідження, щодо відповідності всіх норм та правил, можемо зробити висновок, що робота в приміщеннях станції швидкої медичної допомоги проходить в умовах, які відповідають усім санітарним нормам та техніці безпеки.

Особливе значення для запобігання професійним захворюванням при роботі з обчислювальної техніки має режим праці і відпочинку, який повинен запобігати тривалому впливу шкідливих виробничих чинників на людину.

Недостатня освітленість в централізованих нормах правил роботи з обчислювальною технікою заставляє трудові колективи, працівників і наймачів йти на зустріч один з одним у вирішенні питань охорони праці і закріплювати такі правила в нормах локального характеру, дозволяючи працівникові використовувати гнучкий графік роботи.

Висновки

Питання охорони праці є одним з важливих на сучасному етапі життя нашого суспільства, в період коли працедавці ставлять для себе основним завданням щонайшвидше і з мінімальним вкладенням засобів витягувати найбільшу кількість прибутку, і користуючись таким, що виник останнім часом у нас в країні дефіцитом робочих місць, коли наші громадяни готові за мізерну оплату виконувати найбруднішу роботу мало уваги приділяють, а деколи і взагалі ігнорують вимоги охорони праці.

Збільшення кількості професійних захворювань, нещасних випадків на робочих місцях, що призводять до травм а інколи і до загибелі людей, все це заставляє задуматися про досконалість нашого законодавства в області охорони праці, і думається, що наші законодавчі, старанні і судові органи державної влади чекає ще багато роботи в цьому напрямі.

Одним з напрямів діяльності держави на поліпшення ситуації в області охорони праці є розширення використання норм локального характеру, що дозволяє особливості охорони праці конкретного підприємства відобразити в колективних і трудових договорах

ВИСНОВКИ

Метою даної дипломної роботи була розробка програмного забезпечення веб-додатку для ранжування резюме в рекрутингу, що забезпечить рекрутерам можливість ранжування резюме, та буде зручною та зрозумілою у використанні.

Під час виконання дипломної роботи було:

- на підставі аналізу сучасних програмних продуктів для автоматизації рекрутингової діяльності та особливостей процесу підбору персоналу виконати постановку завдання на розробку відповідного програмного забезпечення;
- проведено аналіз процесу підбору кадрів;
- проведено аналіз вже існуючих програмних рішень;
- поставлена задача на розробку програмного забезпечення веб-додатку для ранжування резюме в рекрутингу;
- розроблене технічне завдання;
- створено проект програмного забезпечення, що містить ескізний, технічний та робочий проекти;
- проведено кодування класів програми та усунення помилок в програмному коді;
- розроблено спеціальний розділ з охорони праці.
- розроблена документація.

В процесі випробувань, за методикою представленою в додатку Д, не було виявлено ніяких помилок та аварійних ситуацій, що свідчить про те, що розроблений програмний продукт працює коректно. Випробування підтверджують відповідність вимогам, викладеним в документі «Технічне завдання» (додаток А).

Отже, в ході даної дипломної роботи було досягнуто мету роботи, а саме створено програмне забезпечення веб-додатку для ранжування резюме в рекрутингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Особливості розвитку ринку рекрутингових послуг в Україні [Електронний ресурс] – Режим доступу: http://eir.pstu.edu/bitstream/handle/123456789/12451/c.119-124.pdf?sequence=1

2 Оптимизация практики рекрутинга [Електронний ресурс] – Режим доступу: https://www.work.ua/articles/career/388/

3 Методика подбора персонала [Електронний ресурс] – Режим доступу: http://fond-samara.com/upload/metodika-podbora-personala.pdf

4 Техніка безпеки при роботі за комп’ютером [Електронний ресурс] – Режим доступу: http://tex-bezbeka.in.ua/vpobut/N_nev.php?nev=11

5 Эскизный проект [Електронний ресурс]. – Режим доступа: http://studbooks.net/1169456/informatika/eskiznyy_proekt

6 Spring Guides [Електронний ресурс]. – Режим доступа: https://spring.io/guides

7 Java Tutorials [Електронний ресурс]. – Режим доступа: https://docs.oracle.com/javase/tutorial/

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Повне найменування теми розробки: розробка програмного забезпечення веб-додатку для ранжування резюме кандидатів.

Коротке найменування програми: ПЗ для ранжування резюме.

Коротка характеристика галузі застосування: програмне забезпечення застосовується для ранжування резюме в рекрутингу, що дозволить підвищити ефективність, спростить процесу підбору кадрів рекрутером та знизить ймовірність виникнення помилок в роботі через вплив людського фактору.

2. Підстави для розробки

Розробка виконується на підставі завдання на переддипломну практику видане кафедрою ПЗАС.

3. Призначення розробки

3.1 Функціональне призначення програми

Функціональним призначенням розроблюваного програмного забезпечення є ранжування резюме кандидатів для спрощення роботи рекрутера.

3.2 Експлуатаційне призначення програми

Експлутаційне призначенням є спрощення роботи кінцевих користувачів, тобто рекрутерів компаній.

Програма призначена для комерційного та некомерційного використання.

4. Вимоги до програми або програмного продукту

4.1. Вимоги до функціональних характеристик

4.1.1. Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

1 перегляд доступних резюме в системі: резюме зберігають по визначеному шляху, з якого користувач має можливість завантажувати нові резюме;

2 ранжування резюме кандидатів на основі наданого профілю роботи;

3 пошук найбільш релевантних резюме: система має виконати ранжування існуючих резюме на основі наданого профілю роботи і відобразити список відповідних;

4 додавання / модифікація / видалення профілей роботи для пошуку релевантного співробітника: користувач може додавати / модифікавати / видаляти профілі роботи для пошуку найбільш релевантних резюме;

5 додавання / видалення резюме в систему: резюме зберігають по визначеному шляху, до якого користувач має можливість завантажувати або видаляти резюме;

6 аутентифікація: можливість використання програмного забезпечення зареєстрованими користувачами;

7 управління рекрутерами: додавання / модифікація / видалення рекрутерів в системі.

4.1.2. Вимоги до організації вхідних даних

Вхідні дані повинні надходити коректні дані з веб-форм від клієнтів.

Резюме кандидатів можуть бути завантажені в систему, як файл, одного з перерахованих форматів: doc, docx, pdf або дані з веб-форм від клієнтів.

До даних користувача відносять: електронну пошту, пароль.

4.1.3. Вимоги до організації вихідних даних

Розроблюване ПЗ повинно генерувати веб-сторінку і відправляти її в якості відповіді на запит. Веб-сторінка

До даних профілю роботи слід віднести: ідентифікатор, найменування, ключові слова, електронна пошта.

До даних користувача слід віднести: електронну пошту, пароль.

4.1.4. Вимоги до часових характеристик

Вимоги до часових характеристик не надаються.

4.2. Вимоги до надійності

4.2.1. Вимоги до забезпечення надійного (сталого) функціонування програми

Особливі вимоги до надійності функціонування не надаються.

4.2.2. Контроль вхідних даних

При введенні даних у форми від клієнта повинна здійснюватися їх валідація.

4.2.3. Час відновлення після відмови

Час відновлення після відмови, викликаного нефатальним збоєм, не повинно перевищувати часу запуску програми на конкретному веб-клієнті.

Час відновлення після відмови, викликаного несправністю технічних засобів, не повинно перевищувати часу на усунення несправностей технічних засобів.

4.3. Умови експлуатації

4.3.1. Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

4.3.2. Вимоги до видів обслуговування

Програма не вимагає проведення будь-яких видів обслуговування.

4.3.3. Вимоги до чисельності та кваліфікації персоналу

Для роботи з програмним продуктом необхідна 1 людина - кінцевий користувач (рекрутер).

4.4. Вимоги до складу і параметрів технічних засобів

До складу технічних засобів висуваються вимоги не нижче наступних:

- центральний процесор - Intel або AMD від 2.0 GHz і вище;
- жорсткий диск – об’єм не менший 40 Гб;
- оперативна пам’ять – об’єм не менший 2 Гб;
- мережевий адаптер або інші засоби доступу до мережі Internet.

Також необхідний доступ до мережі Internet.

4.5. Вимоги до інформаційної та програмної сумісності

Вихідні тексти програми повинні бути реалізовані на мові Java з використанням фреймворку Spring та на мові TypeScript з використанням фреймворку Angular. В

якості інтегрованої середовища розробки програми повинна бути використане середовище IntelliJ IDEA 2018.

Для роботи програми необхідні:

Веб-клієнт: комп’ютер або програма-клієнт в мережах з клієнт-серверною або термінальною архітектурою, браузером.

Вимоги до захисту інформації не пред'являються.

4.6. Вимоги до маркування та пакування

Вимог до упаковки не висуваються.

4.7. Вимоги до транспортування і зберігання

Вимог до транспортування і зберігання не пред'являється.

5. Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

1 керівництво користувача;

2 тексти програмних модулів;

3 опис програми.

4 програма та методика випробувань.

6. Техніко-економічні показники

Техніко-економічні показники не розраховуються.

7. Стадії і етапи розробки

Стадії та етапи розробки ПЗ наведені у таблиці Д1.

Таблиця Д1 – Стадії і етапи розробки ПЗ

Стадії розробки	Етапи робіт	Термін виконання робіт	
		початок етапу	кінець етапу
1 Технічне завдання	1.1 Обгрунтування необхідності розробки програми	05.02.18	15.02.18
	1.2 Розробка технічного завдання	15.02.18	09.03.18
	1.3 Затвердження технічного завдання	09.03.18	20.03.18
2 Ескізний проект	2.1 Розробка ескізного проекту	20.03.18	24.04.18
	2.2 Затвердження ескізного проекту	24.04.18	26.04.18
3 Технічний проект	3.1 Розробка технічного проекту	26.04.18	08.05.18
	3.2 Затвердження технічного проекту	08.05.18	10.05.18
4 Робочий проект	4.1 Розробка програми	10.05.18	05.06.18
	4.2 Розробка програмної документації	05.06.18	12.06.18
	4.3 Випробування програми	12.06.18	20.06.18

8. Порядок контролю і приймання

Приймання та контроль програми повинні проводитися відповідно до узгоджених заздалегідь із замовником програми і методики випробувань.

Кожна стадія розробки повинна бути представлена в зазначені терміни і узгоджена з замовником.

Хід проведення приймально-здавальних випробувань проводиться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань.

На підставі протоколу проведення випробувань виконавець спільно з замовником підписують акт приймання-здачі програмного забезпечення в експлуатацію.

У разі знаходження помилок під час прийому програмного виробу складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджується керівниками організації – замовника та організації – розробника. Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки, і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б – ТЕКСТИ ПРОГРАМНИХ МОДУЛІВ

Лістинг 1 – DocumentParser.java

```
package io.github.tkaczenko.cvranker.service;

import org.apache.poi.hwpf.extractor.WordExtractor;
import org.apache.poi.openxml4j.exceptions.OpenXML4JException;
import org.apache.poi.openxml4j.opc.OPCPackage;
import org.apache.poi.poifs.filesystem.POIFSFileSystem;
import org.apache.poi.xwpf.extractor.XWPFWordExtractor;
import org.apache.tika.exception.TikaException;
import org.apache.tika.metadata.Metadata;
import org.apache.tika.parser.ParseContext;
import org.apache.tika.parser.pdf.PDFParser;
import org.apache.tika.parser.txt.TXTParser;
import org.apache.tika.sax.BodyContentHandler;
import org.apache.xmlbeans.XmlException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import org.xml.sax.SAXException;

import java.io.IOException;
import java.io.InputStream;

@Component
public class DocumentParser {
private static final Logger LOGGER =
LoggerFactory.getLogger(DocumentParser.class);

public String extractTextFromPdf(InputStream documentInputStream) throws
IOException, TikaException, SAXException {
            BodyContentHandler handler = new BodyContentHandler();
            Metadata metadata = new Metadata();
            ParseContext pcontext = new ParseContext();

            PDFParser pdfparser = new PDFParser();
            pdfparser.parse(documentInputStream, handler, metadata, pcontext);
            return handler.toString();
        }

public String extractTextFromTxt(InputStream documentInputStream) throws
IOException,TikaException, SAXException{
            BodyContentHandler handler = new BodyContentHandler();
            Metadata metadata = new Metadata();
            ParseContext pcontext=new ParseContext();
```

```
            TXTParser TexTParser = new TXTParser();
            TexTParser.parse(documentInputStream, handler, metadata,pcontext);
            return handler.toString();
        }

        public String extractTextFromDocx(InputStream documentInputStream) {
            if (documentInputStream == null) {
                return "";
            }
            String extractedText;
            XWPFWordExtractor wordExtractor = getDocxExtractor(documentInputStream);
            extractedText = (wordExtractor == null ? "" : wordExtractor.getText());
            try {
                documentInputStream.close();
            } catch (IOException e) {
                LOGGER.warn("Couldn't close InputStream", e);
            }
            return extractedText;
        }

        public String extractTextFromDoc(InputStream documentInputStream) {
            if (documentInputStream == null) {
                return "";
            }
            WordExtractor wordExtractor = getDocExtractor(documentInputStream);
            String extractedText = (wordExtractor == null ? "" :
wordExtractor.getText());
            try {
                documentInputStream.close();
            } catch (IOException e) {
                LOGGER.warn("Couldn't close InputStream", e);
            }
            return extractedText;
        }

    private XWPFWordExtractor getDocxExtractor(InputStream documentInputStream) {
            try {
                OPCPackage opcPackage = OPCPackage.open(documentInputStream);
                return new XWPFWordExtractor(opcPackage);
            } catch (IOException | OpenXML4JException | XmlException e) {
                LOGGER.warn("Cannot create extractor", e);
            }
            LOGGER.warn("XWPFWordExtractor not created");
            return null;
        }

        private WordExtractor getDocExtractor(InputStream documentInputStream) {
```

```
            try {
                POIFSFileSystem fileSystem = new
POIFSFileSystem(documentInputStream);
                return new WordExtractor(fileSystem);
            } catch (IOException e) {
                LOGGER.warn("Cannot create extractor", e);
            }
            LOGGER.warn("WordExtractor not created");
            return null;
        }
    }
```

Лістинг 2 – WatcherService.java

```
    package io.github.tkaczenko.cvranker.service;

    import io.github.tkaczenko.cvranker.WatcherConfig;
    import org.apache.commons.io.FilenameUtils;
    import org.apache.commons.io.IOUtils;
    import org.apache.lucene.analysis.Analyzer;
    import org.apache.lucene.analysis.standard.StandardAnalyzer;
    import org.apache.lucene.document.Document;
    import org.apache.lucene.document.Field;
    import org.apache.lucene.document.StringField;
    import org.apache.lucene.document.TextField;
    import org.apache.lucene.index.IndexWriter;
    import org.apache.lucene.index.IndexWriterConfig;
    import org.apache.lucene.index.Term;
    import org.apache.lucene.store.Directory;
    import org.apache.lucene.store.FSDirectory;
    import org.apache.tika.exception.TikaException;
    import org.jsoup.helper.Validate;
    import org.slf4j.Logger;
    import org.slf4j.LoggerFactory;
    import org.xml.sax.SAXException;
    import java.io.*;
    import java.nio.charset.StandardCharsets;
    import java.nio.file.*;
    import java.util.ArrayList;
    import java.util.List;

    public class WatcherService implements Runnable {
    private static final Logger LOGGER =
LoggerFactory.getLogger(WatcherService.class);

        private final DocumentParser parser;
        private final WatchKey watchKey;
```

```
        private final List<String> fileNames = new ArrayList<>();

        private final String path;
        private final String index;

    public WatcherService(DocumentParser parser, WatcherConfig config) throws
IOException, InterruptedException {
            this.parser = parser;
            this.path = config.getPath();
            this.index = config.getIndex();
            Path dir = Paths.get(path);
            WatchService watchService = dir.getFileSystem().newWatchService();
            dir.register(watchService, StandardWatchEventKinds.ENTRY_CREATE,
StandardWatchEventKinds.ENTRY_DELETE);
            watchKey = watchService.take();
        }

        @Override
        public void run() {
            while (true) {
                List<WatchEvent<?>> watchEvents = watchKey.pollEvents();
                watchEvents.forEach(watchEvent -> {
                    if
(StandardWatchEventKinds.ENTRY_CREATE.equals(watchEvent.kind())) {
                        LOGGER.info("Created " + watchEvent.context().toString());
                        fileNames.add(watchEvent.context().toString());
                    } else if
(StandardWatchEventKinds.ENTRY_DELETE.equals(watchEvent.kind())) {
                        LOGGER.info("Deleted " + watchEvent.context().toString());
                    }
                });
                if (fileNames.size() > 0) {
                    try {
                        addDocument(fileNames);
                    } catch (IOException e) {
                        LOGGER.warn("Cannot open index directory", e);
                    }
                    fileNames.clear();
                }
            }
        }

        private void addDocument(List<String> fileNames) throws IOException {
            LOGGER.info("Indexing to directory '{}'", new Object[]{index});
            Directory dir = FSDirectory.open(Paths.get(index));
            Analyzer analyzer = new StandardAnalyzer();
            IndexWriterConfig iwc = new IndexWriterConfig(analyzer);
```

```
            iwc.setOpenMode(IndexWriterConfig.OpenMode.CREATE_OR_APPEND);

            IndexWriter writer = new IndexWriter(dir, iwc);
            fileNames.forEach(fileName -> {
                String ext = FilenameUtils.getExtension(fileName);
                String filePath = path + fileName;
                String content = null;
                try {
                    switch (ext) {
                        case "pdf":
                            content = parser.extractTextFromPdf(new
FileInputStream(filePath));
                            break;
                        case "docx":
                            content = parser.extractTextFromDocx(new
FileInputStream(filePath));
                            break;
                        case "doc":
                            content = parser.extractTextFromDoc(new
FileInputStream(filePath));
                            break;
                        case "txt":
                            content = parser.extractTextFromTxt(new
FileInputStream(filePath));
                    }
                } catch (IOException | TikaException | SAXException e) {
                    LOGGER.warn("Cannot extract file", e);
                }
                Validate.notNull(content, "Cannot extract content of file " +
fileName);
      InputStream inputStream = IOUtils.toInputStream(content,
StandardCharsets.UTF_8);
                try {
                    indexDoc(writer, filePath, fileName, inputStream);
                } catch (IOException e) {
                    LOGGER.warn("Cannot indexing file", e);
                }
            });
            writer.close();
        }

      private void indexDoc(IndexWriter writer, String path, String fileName,
InputStream content) throws IOException {
            // make a new, empty document
            Document doc = new Document();
```

```
            // Add the path of the file as a field named "path".  Use a
            // field that is indexed (i.e. searchable), but don't tokenize
            // the field into separate words and don't index term frequency
            // or positional information:
            Field pathField = new StringField("path", path, Field.Store.YES);
            doc.add(pathField);

            Field nameField = new StringField("name", fileName, Field.Store.YES);
            doc.add(nameField);

            // Add the last modified date of the file a field named "modified".
            // Use a LongField that is indexed (i.e. efficiently filterable with
            // NumericRangeFilter).  This indexes to milli-second resolution, which
            // is often too fine.  You could instead create a number based on
            // year/month/day/hour/minutes/seconds, down the resolution you require.
            // For example the long value 2011021714 would mean
            // February 17, 2011, 2-3 PM.
            // doc.add(new LongField("modified", lastModified, Field.Store.NO));

      // Add the contents of the file to a field named "contents".  Specify a Reader,
      // so that the text of the file is tokenized and indexed, but not stored.
            // Note that FileReader expects the file to be in UTF-8 encoding.
            // If that's not the case searching for special characters will fail.
            doc.add(new TextField("content", new BufferedReader(new
InputStreamReader(content, StandardCharsets.UTF_8))));

      if (writer.getConfig().getOpenMode() ==
IndexWriterConfig.OpenMode.CREATE_OR_APPEND) {
      // New index, so we just add the document (no old document can be there):
      LOGGER.info("Adding new index for document {}", new Object[]{path + fileName});
                writer.addDocument(doc);
            } else {
                // Existing index (an old copy of this document may have been
indexed) so
      // we use updateDocument instead to replace the old one matching the exact
                // path, if present:
                writer.updateDocument(new Term("path", path), doc);
            }

        }
      }
```

Лістинг 3 – DocumentAnalyzer.java

```
      package io.github.tkaczenko.cvranker.service;

      import io.github.tkaczenko.cvranker.WatcherConfig;
```

```
import io.github.tkaczenko.cvranker.domain.SearchParameters;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.Term;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.search.*;
import org.apache.lucene.store.FSDirectory;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.io.IOException;
import java.nio.file.Paths;
import java.util.*;

@Component
public class DocumentAnalyzer {
private static final Logger LOGGER =
LoggerFactory.getLogger(DocumentAnalyzer.class);
    private final WatcherConfig config;

    @Autowired
    public DocumentAnalyzer(WatcherConfig config) {
        this.config = config;
    }

public Map<String, Object> searchDocuments(SearchParameters params) throws
IOException {
        String field = "content";
        String queries = null;
        int hitsPerPage = 10;

DirectoryReader reader =
DirectoryReader.open(FSDirectory.open(Paths.get(config.getIndex())));
        IndexSearcher searcher = new IndexSearcher(reader);
        StandardAnalyzer analyzer = new StandardAnalyzer();

        QueryParser parser = new QueryParser(field, analyzer);
        Query query = null;
//        String strQuery = buildQuery(params); // todo
        Term[] terms = {
                new Term(field, "work"), new Term(field, "job"),
                new Term(field, "employment"), new Term(field, "professional"),
                new Term(field, "software"), new Term(field, "mechanical"),
                new Term(field, "engineering"), new Term(field, "full stack")
```

```
            };
            Term[] terms1 = {
                    new Term(field, "experience"), new Term(field, "projects"),
                    new Term(field, "history"), new Term(field, "intern"),
                    new Term(field, "engineer"), new Term(field, "developer")
            };
            Set<String> fileNames = new HashSet<>();
            MultiPhraseQuery multiPhraseQuery = new MultiPhraseQuery.Builder()
                    .add(terms)
                    .add(terms1)
                    .build();
            fileNames.addAll(pageableSearch(searcher, multiPhraseQuery,
hitsPerPage));

            //todo add custom query to search
            return null;
        }

    private List<String> pageableSearch(IndexSearcher searcher, Query query, int
hitsPerPage) throws IOException {
            List<String> fileNames = new ArrayList<>();
            TopFieldDocs fieldDocs = searcher.search(query, 5 * hitsPerPage,
Sort.RELEVANCE);
            for (ScoreDoc doc : fieldDocs.scoreDocs) {
                Document document = searcher.doc(doc.doc);
                String path = document.get("path");
                String name = document.get("name");
                fileNames.add(name);
                LOGGER.info("Found {}", new Object[]{path});
            }
            return fileNames;
        }
    }
```

Лістинг 4 – CVRankerApplication.java

```
package io.github.tkaczenko.cvranker;

import io.github.tkaczenko.cvranker.service.DocumentParser;
import io.github.tkaczenko.cvranker.service.WatcherService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CVRankerApplication implements CommandLineRunner {
```

```
    private final DocumentParser parser;
    private final WatcherConfig config;

    @Autowired
    public CVRankerApplication(DocumentParser parser, WatcherConfig config) {
        this.parser = parser;
        this.config = config;
    }

    public static void main(String[] args) {
        SpringApplication.run(CVRankerApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        new Thread(new WatcherService(parser, config), "watcher-
service").start();
    }
}
```

Лістинг 5 – RepositoryRestConfig.java

```
package io.github.tkaczenko.cvranker;

import org.springframework.beans.factory.config.BeanDefinition;
import
org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.type.filter.RegexPatternTypeFilter;
import org.springframework.data.rest.core.config.RepositoryRestConfiguration;
import
org.springframework.data.rest.webmvc.config.RepositoryRestConfigurerAdapter;

import java.util.Set;
import java.util.regex.Pattern;

@Configuration
public class RepositoryRestConfig extends RepositoryRestConfigurerAdapter {
    @Override
    public void configureRepositoryRestConfiguration(RepositoryRestConfiguration
config) {
ClassPathScanningCandidateComponentProvider provider = new
ClassPathScanningCandidateComponentProvider(false);
provider.addIncludeFilter(new RegexPatternTypeFilter(Pattern.compile(".*")));

Set<BeanDefinition> beans =
provider.findCandidateComponents("io.github.tkaczenko.cvranker.domain");
```

```
            for (BeanDefinition bean : beans) {
                Class<?> idExposedClasses;
                try {
                    idExposedClasses = Class.forName(bean.getBeanClassName());
                    config.exposeIdsFor(Class.forName(idExposedClasses.getName()));
                } catch (ClassNotFoundException e) {
                    // Can't throw ClassNotFoundException due to the method
signature. Need to cast it
      throw new RuntimeException("Failed to expose `id` field due to", e);
                }
            }
        }
      }
```

Лістинг 6 – WatcherConfig.java

```
package io.github.tkaczenko.cvranker;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

@Component
@ConfigurationProperties("watcher")
public class WatcherConfig {
    private String path;

    private String index;

    public String getPath() {
        return path;
    }

    public void setPath(String path) {
        this.path = path;
    }

    public String getIndex() {
        return index;
    }

    public void setIndex(String index) {
        this.index = index;
    }
}
```

Лістинг 7 – WebSecurityConfig.java

```
package io.github.tkaczenko.cvranker;
```

```
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
Adapter;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable();
        http.authorizeRequests()
                .antMatchers("/api/**").permitAll();
    }
}
```

Лістинг 8 – Account.kt

```
package io.github.tkaczenko.cvranker.domain

import java.util.*
import javax.persistence.*
import javax.validation.constraints.NotBlank

@Entity
data class Account(
        @Id
        val email: String,
        val firstName: String,
        val lastName: String,
        @NotBlank
        @Column(nullable = false)
        val password: String,
        @NotBlank
        @Column(nullable = false, updatable = false)
        val createdAt: Date,
        @OneToOne(fetch = FetchType.LAZY)
        val address: Address
)
```

Лістинг 9 – Address.kt

```
package io.github.tkaczenko.cvranker.domain
```

```
import javax.persistence.GeneratedValue
import javax.persistence.Id

data class Address(
        @Id @GeneratedValue
        val id: Int,
        val city: String,
        val street: String
)
```

Лістинг 10 – Role.kt

```
package io.github.tkaczenko.cvranker.domain

import lombok.Builder
import javax.persistence.Column
import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.Id
import javax.validation.constraints.NotBlank

@Entity
@Builder
data class Role(
        @Id @GeneratedValue
        val id: Long,
        @NotBlank
        @Column(nullable = false)
        val name: String
)
```

Лістинг 11 – AccountRole.kt

```
package io.github.tkaczenko.cvranker.domain

import lombok.Builder
import javax.persistence.*

@Entity
@Builder
data class AccountRole(
        @Id @GeneratedValue
        val id: Int,
        @ManyToOne
        @JoinColumn(foreignKey = ForeignKey(name = "account_id_fkey"))
        val account: Account,
        @ManyToOne
        @JoinColumn(foreignKey = ForeignKey(name = "role_id_fkey"))
```

```
        val role: Role
)
```

Лістинг 12 – JobProfileConfiguration.kt

```
package io.github.tkaczenko.cvranker.domain

import lombok.Builder
import javax.persistence.Column
import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.Id
import javax.validation.constraints.NotBlank

@Entity
@Builder
data class JobProfileConfiguration(
        @Id @GeneratedValue
        val jobId: Long,
        @NotBlank
        @Column(nullable = false)
        val jobTitle: String,
        @NotBlank
        @Column(nullable = false)
        val keywords: String,
        @NotBlank
        @Column(nullable = false)
        val email: String
)
```

Лістинг 13 – JobProfileConfiguration.kt

```
package io.github.tkaczenko.cvranker.domain

data class SearchParameters(
        val id: Int,
        val skills: Array<String>,
        val previosEmployeer: Array<String>,
        val isMaster: Boolean,
        val isBachelor: Boolean
)
```

Лістинг 14 – JobProfileConfigurationRepository.kt

```
package io.github.tkaczenko.cvranker.repository

import io.github.tkaczenko.cvranker.domain.JobProfileConfiguration
import org.springframework.data.repository.PagingAndSortingRepository
import org.springframework.data.rest.core.annotation.RepositoryRestResource
```

```
import org.springframework.web.bind.annotation.CrossOrigin

@RepositoryRestResource
@CrossOrigin(origins = ["http://localhost:4200"])
interface JobProfileConfigurationRepository :
PagingAndSortingRepository<JobProfileConfiguration, Long>
```

Лістинг 15 – AccountRepository.kt

```
package io.github.tkaczenko.cvranker.repository

import io.github.tkaczenko.cvranker.domain.Account
import org.springframework.data.repository.CrudRepository
import org.springframework.data.rest.core.annotation.RepositoryRestResource
import org.springframework.web.bind.annotation.CrossOrigin

@RepositoryRestResource
@CrossOrigin(origins = ["http://localhost:4200"])
interface AccountRepository : CrudRepository<Account, Long>
```

Лістинг 16 – JobProfileConfigurationResourse.kt

```
package io.github.tkaczenko.cvranker.web.rest

import io.github.tkaczenko.cvranker.repository.JobProfileConfigurationRepository
import org.springframework.data.rest.webmvc.RepositoryRestController
import org.springframework.web.bind.annotation.CrossOrigin

@RepositoryRestController
@CrossOrigin(origins = ["http://localhost:4200"])
class JobProfileConfigurationResourse(private val repository:
JobProfileConfigurationRepository)
```

Лістинг 17 – ResumeController.kt

```
package io.github.tkaczenko.cvranker.web.rest

import io.github.tkaczenko.cvranker.domain.SearchParameters
import io.github.tkaczenko.cvranker.service.DocumentAnalyzer
import org.springframework.web.bind.annotation.*
import javax.validation.Valid

@RestController
@RequestMapping("/resume")
@CrossOrigin(origins = ["http://localhost:4200"])
class ResumeController(private val analyzer: DocumentAnalyzer) {

    @PostMapping
```

```
        fun search(@Valid @RequestBody searchParameters: SearchParameters):
Map<String, Any> {
            return analyzer.searchDocuments(searchParameters)
        }
    }
```

ДОДАТОК В – ОПИС ПРОГРАМИ

1. Загальні відомості

Розроблено програмне забезпечення веб-додатку для ранжування резюме включає. Розроблений веб-клієнт для ранжування резюме є простим, зрозумілим та функціональним інструментом для ранжування резюме.

Програмне забезпечення було розроблено використовуючи Intellij IDEA в якості засобу розробки та Tomcat в якості локального веб-серверу. Програмне забезпечення розроблене із використанням наступних технологій:

- HTML 5, CSS 3 – розмітка та стилізація веб-сторінок;
- JavaScript (AJAX, Angular) – клієнтські скрипти, асинхронне завантаження даних, реактивність користувацького інтерфейсу;
- PostgreSQL – обробка та збереження даних у БД;
- Java Spring Framework – backend-розробка.

2. Функціональне призначення

Веб-сайт забезпечує виконання наступних функцій респондентом:

- реєстрація;
- проходження опитування.

3. Опис логічної структури

Програмне забезпечення розроблене із використанням шалону проектування MVC (Model-View-Controller). У рамках цього шаблону проектування програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами – модель, представлення та контроллер.

4. Технічні та програмні засоби

До складу технічних засобів висуваються вимоги не нижче наступних:

- центральний процесор - Intel або AMD від 2.0 GHz і вище;
- жорсткий диск – об’єм не менший 40 Гб;
- оперативна пам’ять – об’єм не менший 2 Гб;
- мережевий адаптер або інші засоби доступу до мережі Internet.

Також необхідний доступ до мережі Internet.

5. Виклик та навантаження

Програмне забезпечення проходить підготовку до публікації за допомоги середовища розробки Intellij Idea. Підготовлені вихідні файли завантажуються на віддалений сервер. Після завантаження програмного забезпечення серверу, веб-сайт стають доступні за певними посиланнями у браузері.

6. Вхідні та вихідні дані

Вхідні дані організовані у вигляді спеціальних форм та полів, що відповідають певному шаблону. Данні, що вводяться вручну, перевіряються на коректність у програмному інтерфейсі та на клієнтському програмному забезпеченні. Файли зберігаються на сервері.

Вихідні дані представлені у вигляди списків та таблиць. Також користувач отримує інформацію про результат виконаної ним операції.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА

Програмне забезпечення розроблено, як веб-додаток доступний за визначеним посиланням. При зверненні до посилання буде відкрито головну сторінку програми, що зображена на рисунку Г.1.

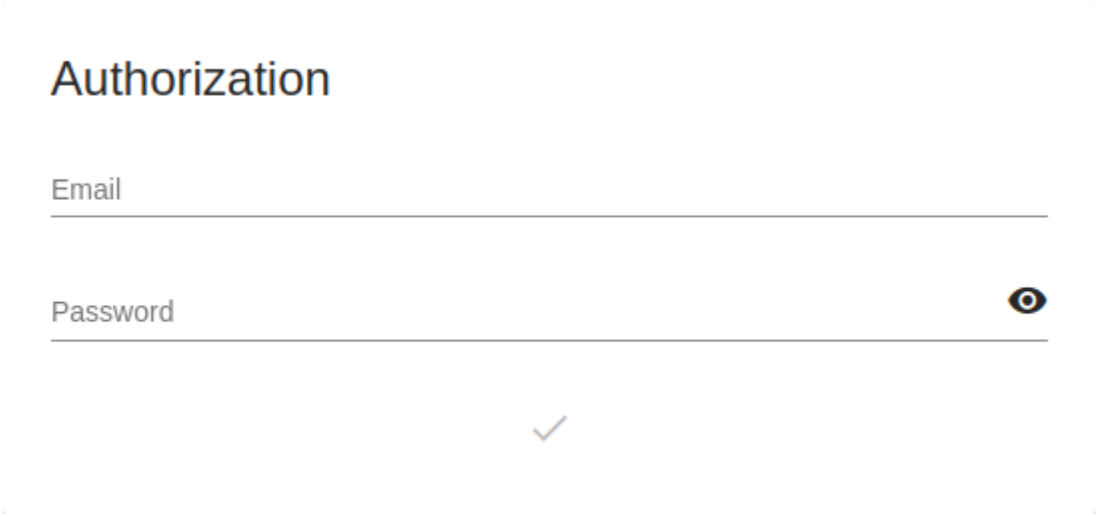

Рисунок Г.1 – Вікно аутентифікації

Після успішної аутентифікації користувача буде відображена головна сторінка веб-додатку, яка містить головне меню зображене на рисунку Г.2.

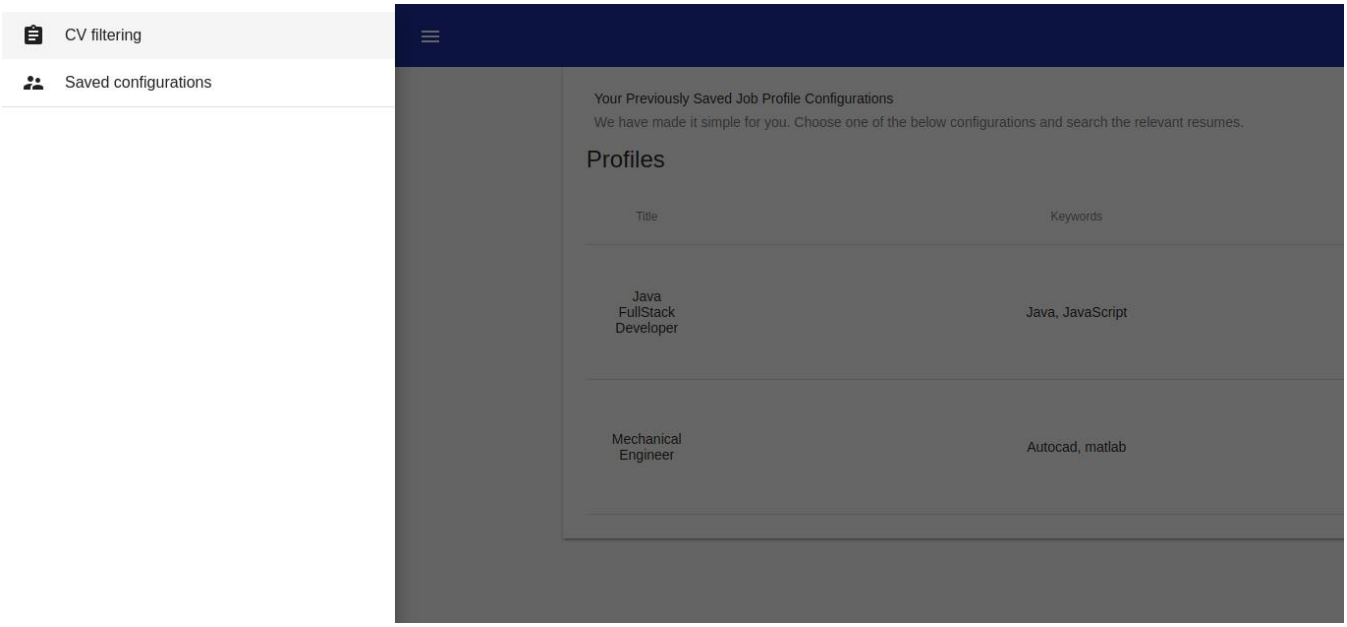

Рисунок Г.2 – Головна сторінка з головним меню

Сторінка з можливістю перегляду та ранжування резюме (рисунок Г.3), яка дозволяє користувачу виконати пошук найбільш релевантного резюме, зберегти новий профіль роботи для майбутнього пошуку, а також переглянути результат

пошуку та завантажити необхідний файл резюме. Для виконаня вище перерахованих дій використовуються кнопки з інтуєтивно зрозумілими іконками.

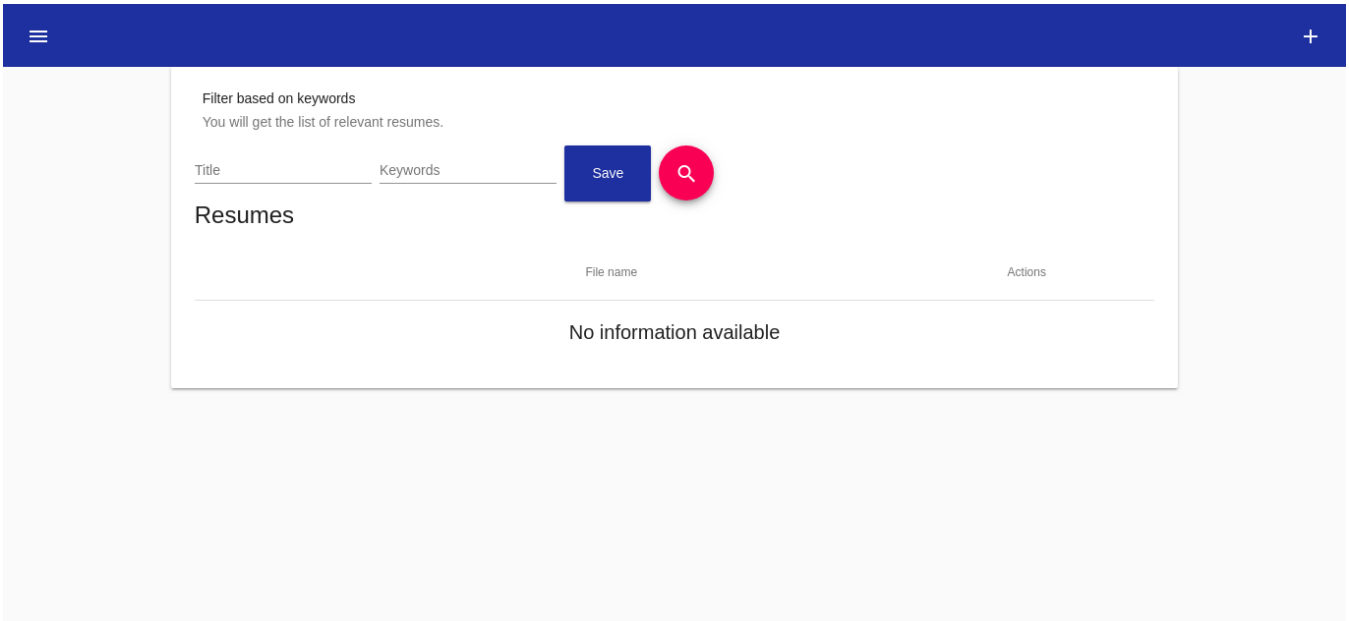

Рисунок Г.3 – Сторінка з можливістю ранжування резюме, збереження профілю роботи та завантаження файлів резюме із списку результатів

Сторінка перегляду збережених профілей роботи (рисунок Г.4) дозволить виконати пошук у відповідності до обраного профілю, а також видалити та редагувати обраний профіль за допомогою інтуєтивно зрозумілих кнопок.

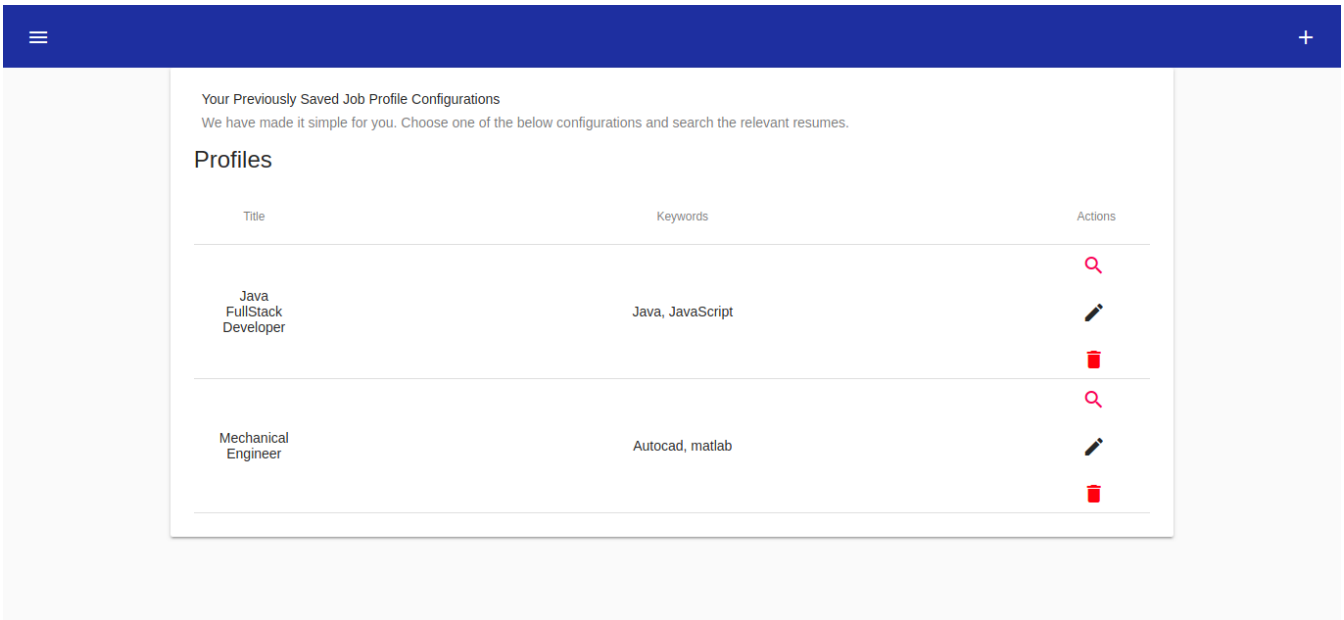

Рисунок Г.4 – Головне вікно з можливістю перегляду, редагуваня збережених профілей роботи та пошуку у відповідності до обраного профілю

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об'єкт випробування

Об’єктом випробування є програмне забезпечення веб-додатку для ранжування резюме в рекрутингу, розроблене в рамках даної дипломної роботи.

2 Мета випробування

Метою проведення випробування є перевірка працездатності даного програмного продукту, перевірка відповідності характеристик та вимог розробленої програми, викладених у технічному завдані, приведення прикладу роботи та отримання відповідних навичок.

3 Вимоги до додатка

При проведенні випробувань функціональні характеристики (можливості) програми підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

4 Вимоги до програмної документації

4.1 Склад програмної документації, пропонованої на випробування

Програмна документація повинна включати в себе наступні документи:

1 Технічне завдання;

2 Текст програми;

3 Опис програми;

4 Програма та методика випробувань.

4.2 Спеціальні вимоги

Спеціальні вимоги до програмної документації не висуваються.

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувалися під час випробувань:

- Процесор Intel Core i7 2ГГц;
- Пам'ять 8Гб;
- Жорсткий диск 70Гб RAID 1;
- Здатність екрану – 1360 х 768 пікселів;
- Клавіатура 101/102-х клавішна рус / лат;
- Маніпулятор «миша».

5.2 Програмні засоби, що використовуються під час випробувань

Програмне забезпечення веб-додатку для ранжування резюе в рекрутингу не пред'являє ніяких вимог до апаратної платформи.

Склад програмних засобів, що використовувалися під час випробувань:

Операційні системи:

- Windows 10;
- Ubuntu 16.04.

Браузери:

- Mozzilla FireFox 60.0;
- Microsoft Internet Explorer 11.0;
- Google Chrome 69.0.

5.3 Порядок проведення випробувань

Порядок проведення випробувань складається з перевірки вимог до функціональних характеристик програмного продукту та перевірки вимог до програмної документації.

Методики проведення перевірок, що входять в перелік другого етапу випробувань, викладені в даному програмному документі в розділі «Методи випробувань».

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Перевірка дотримання вимог програмної документації на програмний продукт виконується візуально представником служби, відповідальної за експлуатацію. У ході перевірки зіставляється склад і комплектність програмної документації,

представленої розробником, з переліком програмної документації, наведеним у пункті «Склад програмної документації, пропонованої на випробування» цього документа.

Перевірка вважається завершеною у випадку відповідності складу та комплектності програмної документації, представленої розробником, переліком програмної документації, наведеному у зазначеному вище пункті.

За результатами проведення перевірки представник служби, відповідальної за експлуатацію, вносить запис до протоколу випробувань – “Комплектність технічних і програмних засобів відповідає (не відповідає) вимогам пунктів «Технічні засоби, що використовується під час випробувань» та «Програмні засоби, що використовуються під час випробувань» даного документа”.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності програми виконується згідно з пунктом «Вимоги до функціональних характеристик» технічного завдання.

Перевірка вважається завершеною у разі відповідності складу і послідовності дій, при виконанні даної перевірки, вказаною вище пункту технічного завдання.

В процесі тестування була перевірена функціональність програмного забезпечення веб-додатку для ранжування резюме в рекрутингу. У таблиці, що наведено нижче, вказано перелік випробувань основних функціональних можливостей.

За результатами проведення перевірки представник служби, відповідальної за експлуатацію, вносить запис до протоколу випробувань - ”Функціональні вимоги відповідають (не відповідають) вимогам пункту «Вимоги до програми» цього документа”.

Таблиця Д1 – Методика випробування

	Дія	Очікуваний результат	Результат перевірки	Зауваження
	2	3	4	5
	Завантаження браузера та перехід на сайт	У вікні браузера з’являється форма аутентифікації	Виконано	
	Перевірка правильності переходу до панелі адміністратора	У вікні браузера з’являється панель адміністратора	Виконано	
	Перевірка перегляду завантажених резюме	У вікні браузера з’являться сторінка зі списком завантажених резюме	Виконано	
	Перевірка завантаження резюме	Система виконує збереження обраного файлу резюме	Виконано	
	Перевірка видалення резюме	Система виконує видалення обраного файлу резюме	Виконано	
	Перевірка перегляду збережених профілей роботи	У вікні браузера з’являться сторінка зі списком профілей роботи	Виконано	
	Перевірка додавання профілей роботи	Система виконує збереження обраного профілю роботи	Виконано	

Продовження таблиці Д1 – Методика випробування

	2	3	4	
	Перевірка видалення профілей роботи	Система виконує видалення обраного профілю роботи	Виконано	
	Ранжування резюме	У вікні браузера з’являться сторінка зі списком релевантних резюме	Виконано	