

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПОШУКУ ВАКАНСІЙ ДЛЯ ПРАЦЕВЛАШТУВАННЯ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
SEARCHING JOB VACANCIES

Виконав: студент(ка) 2 курсу, групи КНТ-129м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Гвоздилко М.О.

(прізвище та ініціали)

Керівник Зайко Т.А.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Гвоздилка Микити Олександровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів пошуку вакансій для працевлаштування
Research and Software Implementation of Methods for Searching Job Vacancies

керівник проєкту (роботи) Зайко Тетяна Анатоліївна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) грудень 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми пошуку вакансій для працевлаштування. 2. Матеріали і методи. 3. Опис програми. 4. Розробка програми. 5. Експлуатація та тестування програми. 6. Економіко-організаційна частина, 7. Охорона праці та безпека в надзвичайних ситуаціях.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Зайко Т.А., доцент		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “ 1 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10 тиждень	Розділ 5
6	Розробка економіко-організаційної частини	11 тиждень	Розділ 6
7	Розробка заходів з охорони праці та безпеки в надзвичайних ситуаціях	11 тиждень	Розділ 7
8	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
9	Захист роботи	13 тиждень	

Студент(ка)

(підпис) Гвоздилко М.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Зайко Т.А.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
107 с., 27 рис., 9 табл., 3 дод., 30 джерел.

ПОШУК ВАКАНСІЙ, ПРАЦЕВЛАШТУВАННЯ,
ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, КЛАСИФІКАЦІЯ ДАНИХ,
WORD2VEC.

Об'єкт дослідження – процес працевлаштування.

Предмет дослідження – методи пошуку вакансій для працевлаштування.

Метою роботи є розроблення програмного забезпечення пошуку вакансій для працевлаштування для полегшення рішень, що приймаються в цій галузі на основі підвищення їх обґрунтованості.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Python, вебфреймворк Flask, система управління базою даних MySQL, метод k-найближчих сусідів, word2vec.

Результати. Розроблено програмне забезпечення підтримки приймаємих рішень щодо пошуку вакансій для працевлаштування.

Наукова новизна роботи полягає в розробленому методі пошуку вакансій для працевлаштування, що виконує пошук подібних вакансій на основі їх класифікації за допомогою методу k-найближчих сусідів, використовуючи в якості додаткових ознак закодовані за допомогою моделі word2vec значення, що дозволяє обґрунтовано приймати рішення про вибір вакансій або резюме для детального аналізу.

Висновки. Розроблено вебзастосунок пошуку вакансій для працевлаштування.

Галузь використання – пошук вакансій, керування людськими ресурсами.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 107 pages, 27 figures, 9 tables, 3 appendixes, 30 sources.

JOB SEARCH, EMPLOYMENT, DATA MINING, DATA CLASSIFICATION, WORD2VEC.

The object of research is the employment process.

The subject of research is methods of finding vacancies for employment.

The aim of the work is to develop software for searching job vacancies for employment to facilitate decisions made in this area on the basis of increasing its validity.

Materials, methods and technical means: object-oriented programming, Python programming language, Flask web framework, MySQL database management system, k-nearest neighbor method, word2vec.

Results. The software of support of the accepted decisions concerning search of vacancies for employment is developed.

The scientific novelty of the work lies in the developed method of finding vacancies for employment, which searches for similar vacancies based on their classification using the method of k-nearest neighbors, using text descriptions encoded using the word2vec model, which allows reasonable decision-making on vacancies or resume for detailed analysis.

Conclusions. A web application for job search for employment has been developed.

Field of usage is job search, human resource management.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми пошуку вакансій для працевлаштування.....	10
1.1 Основні принципи пошуку вакансій для працевлаштування.....	10
1.2 Огляд програмного забезпечення пошуку вакансій для працевлаштування	14
1.3 Висновки за розділом 1	17
2 Матеріали і методи.....	18
2.1 Визначення задачі пошуку вакансій для працевлаштування	18
2.2 Модель Word2vec	18
2.3 Метод k-найближчих сусідів	20
2.4 Метод пошуку вакансій для працевлаштування.....	22
2.5 Висновки за розділом 2	24
3 Опис програми.....	25
3.1 Вимоги до програми	25
3.2 Вибір засобів і середовища проєктування і розроблення	27
3.3 Проєктування бази даних	28
3.4 Структура програмного проєкту	33
3.5 Висновки за розділом 3	34
4 Розробка програми	36
4.1 Опис логіки функціонування програми.....	36
4.2 Опис реалізації вебзастосунку	38
4.3 Висновки за розділом 4	40
5 Експлуатація та тестування програми	41
5.1 Призначення програми	41
5.2 Умови виконання програми	41
5.3 Опис роботи користувача з програмою	41
6 Економіко-організаційна частина.....	48

6.1 Планування розробки програмного продукту.....	48
6.2 Планування розробки програмного продукту.....	48
6.3 Розрахунок основної заробітної плати	51
6.4 Розрахунок додаткової заробітної плати	52
6.5 Єдиний соціальний внесок	52
6.6 Визначення витрат на матеріали	52
6.7 Витрати на спеціальне обладнання	53
6.8 Інші прямі витрати	57
6.9 Розрахунок накладних витрат.....	58
6.10 Розрахунок економічної ефективності програмного продукту.....	59
7 Охорона праці та безпека в надзвичайних ситуаціях.....	60
7.1 Аналіз потенційних небезпек.....	60
7.2 Заходи забезпечення безпеки.....	60
7.3 Заходи з забезпечення виробничої санітарії та гігієни праці	61
7.4 Заходи з пожежної безпеки	66
7.5 Заходи забезпечення безпеки у надзвичайних ситуаціях	67
Висновки	69
Перелік джерел посилання	71
Додаток А Технічне завдання	74
Додаток Б Текст програми	79
Додаток В Слайди презентації.....	100

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ЕОМ – електронно-обчислювальна машина;

ПК – персональний комп'ютер.

ВСТУП

Майже будь-яка людина в процесі свого життя стикається з пошуком вакансій для працевлаштування, а часто і неодноразово. Інформаційні технології в даній галузі можуть призвести до позитивного ефекту для кожної людини. Зазвичай на таких сайтах існує багато вакансій, але пошук дещо ускладнений, тому що виходячи лише з назви або окремих фільтрів важко знайти потрібне оголошення, окрім як проаналізувавши всі доступні оголошення. Це вимагає багато часу, концентрації. Створення якісного програмного рішення, що допоможе в цій сфері, є важливим і актуальним практичним завданням.

Багато інформації про вакансію міститься в її тексті, тому якщо його проаналізувати, то можна отримати інструмент для виконання потенційно якісного аналізу даних. Результатом такого аналізу є набір вакансій, які відповідають пошуковому запиту користувача, враховуючи індивідуальні особливості зокрема. Саме пошуку такого засобу і засобів виділення подібних між собою вакансій для організації пошуку вакансій для працевлаштування і присвячено дану роботу.

Метою виконуваної роботи є розроблення програмного забезпечення пошуку вакансій для працевлаштування для полегшення рішень, що приймаються в цій галузі на основі підвищення їх обґрунтованості.

1 АНАЛІЗ ПРОБЛЕМИ ПОШУКУ ВАКАНСІЙ ДЛЯ ПРАЦЕВЛАШТУВАННЯ

1.1 Основні принципи пошуку вакансій для працевлаштування

Пошук вакансій для працевлаштування – це достатньо непростий процес, для якого можна виділити цілий набір шляхів [1]-[3]:

- знайти найкращі списки вакансій: необхідно переглянути найкращі сайти пошукових систем роботи, банки вакансій, вебсайти компаній, мережеві сайти, нішеві сайти та сайти, перелічені за типами роботи, також потрібно подумати про співпрацю з рекрутером, щоб максимізувати свої можливості,

- уважніше слідкувати за пошуком роботи: необхідно використовувати пошукові системи для пошуку роботи, використовуючи ключові слова, які відповідають інтересам та місцезнаходженню користувача, звуження критеріїв пошуку допоможе зосередитись на пошуку роботи та надасть більше релевантних списків вакансій для перегляду та меншу кількість нерелевантних списків вакансій, які потрібно відсіяти, потрібно використовувати розширені параметри пошуку, щоб детально ознайомитися з місцем, де людина хоче працювати, і конкретними посадами, які цікавлять людину;

- створити свій професійний бренд: необхідно створити профілі на LinkedIn та інших мережевих сайтах, сильний особистий бренд, який зображує людину в професійному світлі, забезпечить рекрутерам і роботодавцям сильне позитивне враження про кандидата, який повинен їх зацікавити;

- зв'язатися зі своїми контактами: тепер, коли створені профілі на мережевих сайтах, необхідно почати їх використовувати, зв'яжіться з усіма, кого знаєте, тому що людина ніколи не знає, який контакт може допомогти у пошуку роботи або зв'язати з кимось, хто може допомогти;

– використовувати програми та інструменти пошуку роботи: існує багато застосунків, віджетів, гаджетів та інструментів, які допоможуть пришвидшити пошук роботи та керувати кар'єрою, необхідно використовувати їх, щоб організувати пошук роботи та заощадити цінний час на пошук роботи, можна виконувати багато завдань з пошуку роботи зі свого смартфона або планшета;

– створити список компаній, в є бажання працювати: непогано дослідити інформацію про компанії та створити список компаній, на які потрібно націлити свій пошук роботи, вся необхідна інформація доступна в Інтернеті, і в Інтернеті легко знайти детальну інформацію про потенційних роботодавців, коли у людини є список роботодавців, у яких вона мріє працювати, можна зробити спеціальну роз'яснювальну роботу, щоб помітити заявку, можна навіть зареєструватися, щоб отримувати сповіщення електронною поштою про нові вакансії відразу після їх розміщення;

– не потрібно поспішати націлювати своє резюме та супровідний лист на одне конкретне рішення: звідки роботодавці можуть знати, що у здобувача є навички, які вони шукають, якщо їх не показати, важливо не поспішати з написанням цільових резюме та супровідних листів, які конкретно пов'язують кваліфікацію з критеріями прийняття на роботу, на яку здобувач претендує, менеджер з найму зможе з першого погляду побачити, чому і як здобувач кваліфікований для роботи, буде набагато більше шансів отримати співбесіду, ніж якби просто надіслали загальний лист та резюме;

– підготуватися до інтерв'ю: якщо заздалегідь підготуватися до співбесіди, це допоможе досягти успіху, чим більше людина підготовлена, тим менше це буде стресом, потрібно дослідити компанію перед тим, як піти на співбесіду, одягнутися належним чином, потренуватися відповідати та задавати питання на співбесіді та докладати спільних зусиль, щоб вразити інтерв'юера своїми вміннями, досвідом, впевненістю та досвідом;

– не потрібно забувати про подальші дії: важливо продовжити роботу після співбесіди, подякувавши всім, з ким зустрічалися, також потрібно

повторити про свою зацікавленість у цій посаді та нагадати менеджеру з найму, чому саме ви чудовий кандидат на цю роботу, кожен любить, щоб його оцінювали, і швидкий електронний лист чи примітка, в якій подякують інтерв'юєру за його час, дасть ще одну можливість справити гарне враження;

– потрібно подумати, приймати або відхилити пропозицію про роботу: колю здобувач отримує пропозицію про роботу, важливо не поспішаючи ретельно оцінити її, щоб можна було прийняти якісне рішення, не потрібно приймати роботу лише тому, що вона була запропонована, потрібно ретельно її оцінити, і якщо здобувач відмовляється, то необхідно зробити це ввічливо [1]-[3].

З виконаного аналізу видно, що одним з головних принципів пошуку вакансій для працевлаштування є використання відповідних програмних засобів. Від того, як саме реалізується в них пошук залежить успішність пошуку для конкретної людини.

Безумовний плюс сайтів працевлаштування в тому, що вони прекрасно економлять час, при цьому можна виділити наступне:

–здобувачеві не потрібно годинами обдзвонювати компанії, що цікавлять його, і потім витратити час на дорогу до них для того, щоб почути про незацікавленість, кожен раз розповідати новому зацікавленому роботодавцю про свій досвід і вміння теж не потрібно: на офіційному сайті вакансій можна розмістити резюме з необхідною особистою інформацією: починаючи від віку і освіти і закінчуючи скан-копіями професійних сертифікатів і рекомендаціями з попереднього місця роботи, крім того, на сайтах роботи є фільтри за різними параметрами, які допомагають знайти найбільш підходящу пропозицію, до них відноситься бажана зарплата, сфера діяльності, наявність або відсутність досвіду роботи, територіальне розташування фірми, тип зайнятості і багато іншого, за таким же принципом цей фільтр можуть використовувати і роботодавці, яким сайти працевлаштування теж істотно полегшують життя;

– прямі сайти вакансій бувають як загальні, так і цілеспрямовані: наприклад, існують портали тільки для перукарів-стилістів або сайти, де представлені вакансії в певному місті чи районі, тому критерії вибору сайту працевлаштування залежать від конкретного претендента, однак варто вивчити репутацію порталу, щоб не змарнувати час на малоефективний сервіс і не зіткнутися з неперевіреними роботодавцями, зробити це можна, вивчивши відгуки на сторонніх ресурсах [4].

Загалом існуючі сервіси пошуку вакансій для працевлаштування можна розділити за наступними властивостями:

– деякі вебсайти пропонують лише списки вакансій, а інші також пропонують можливості для створення мереж, дискусійні групи, статті та інформаційні бюлетені з порадами щодо пошуку роботи та навіть списки торгових асоціацій, з якими можна зв'язатися, щоб отримати додаткову інформацію про певні професії;

– більшість найбільших вебсайтів про роботу щодня мають сотні або тисячі нових публікацій, але, немає сенсу реєструватися на сайті, який отримує лише кілька нових списків вакансій на тиждень, якщо на сайті багато списків, яким принаймні 30 днів, або лише декілька з поточного тижня, цей сайт, ймовірно, не вартий витраченого часу;

– деякі веб-сайти з пошуку роботи заробляють гроші на стороні, продаючи інформацію про учасників рекламодавцям та іншим тіньовим компаніям, тому обов'язково потрібно переглянути політику конфіденційності кожного сайту, щоб переконатися, що ваша інформація не буде передана третім сторонам, якщо політика цього не гарантує, є багато інших сайтів, якими можна скористатися;

– залежно від вашої ситуації деякі платні функції можуть коштувати грошей: досить багато сайтів з пошуку роботи є безкоштовними для базового використання, але деякі беруть плату за такі розширені функції, як обмін великими повідомленнями, залежно від ситуації, деякі платні функції можуть

коштувати – але ніколи не потрібно довіряти сайту, який стягує плату або змушує зареєструватися, лише щоб зазирнути [5].

1.2 Огляд програмного забезпечення пошуку вакансій для працевлаштування

Одним з найвідоміших сервісів пошуку вакансій для працевлаштування в Україні є сервіс *rabota.ua* (рис. 1.1).

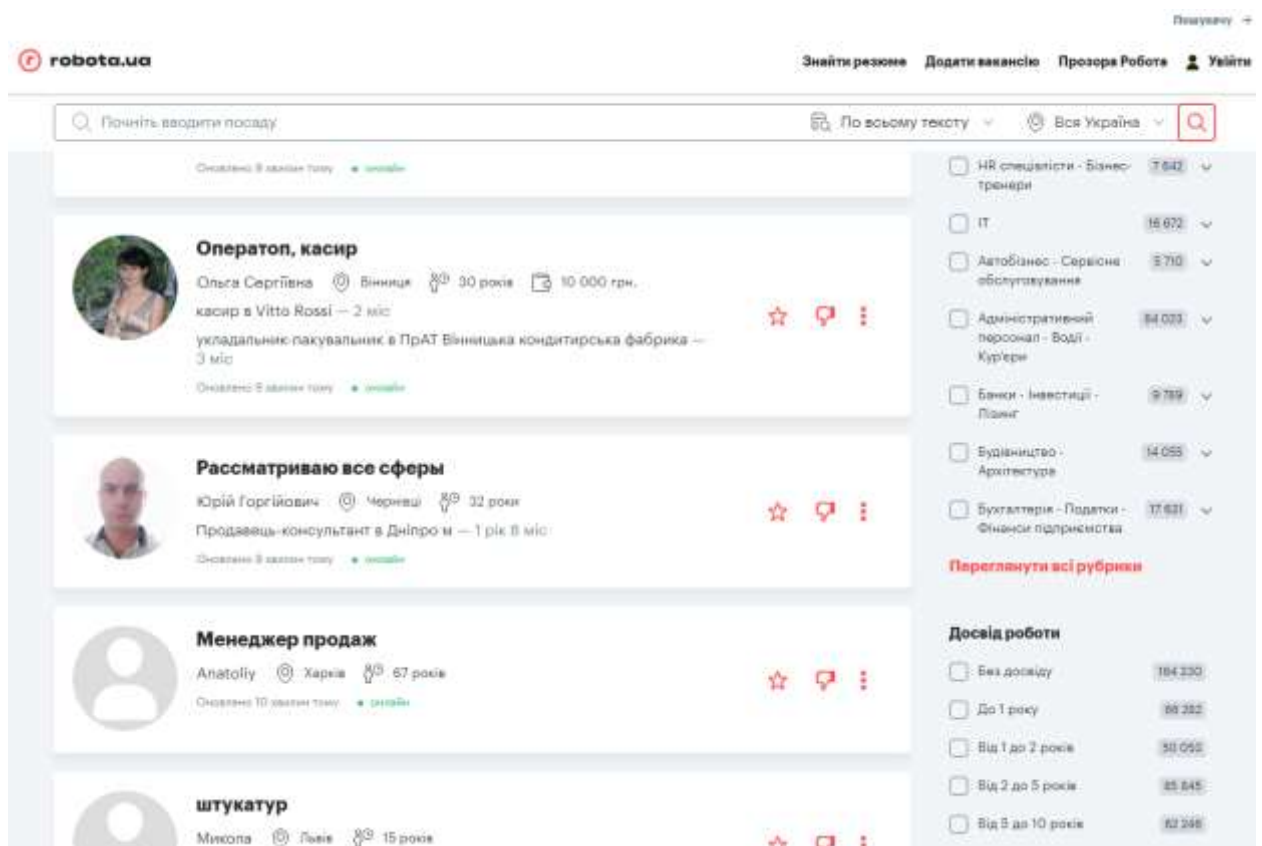


Рисунок 1.1 – Вебінтерфейс *rabota.ua*

Даний сервіс надає наступні функціональні можливості:

– пошук вакансій за назвою і іншими параметрами та набором фільтрів, що включають зокрема вікові фільтри, статеві, рівень очікуваної зарплати, досвід роботи, володіння мовами, тип зайнятості, галузь, рівень освіти, можливість переїзду, місто;

- працедавці можуть додавати власні вакансії, виконувати пошук резюме здобувачів, переглядати їх повні дані;
- здобувачам переглядати перелік популярних професій, міст;
- здобувачам розміщати власне резюме.

Одним з найраших засобів пошуку вакансій для працевлаштування у світі є сервіс Indeed.

Даний сервіс надає можливості здобувачам пошуку вакансій у різних галузях, обмежуючи при цьому місце пошуку, час останнього оновлення, визначаючи порядок сортування, параметри пошуку, а представникам компаній виконувати внесення вакансій, при цьому інтерфейс застосунку максимально простий, в деякій мірі навіть дещо несучасний (рис. 1.2), але у даному випадку важливим є наповнення даними та безкоштовність.

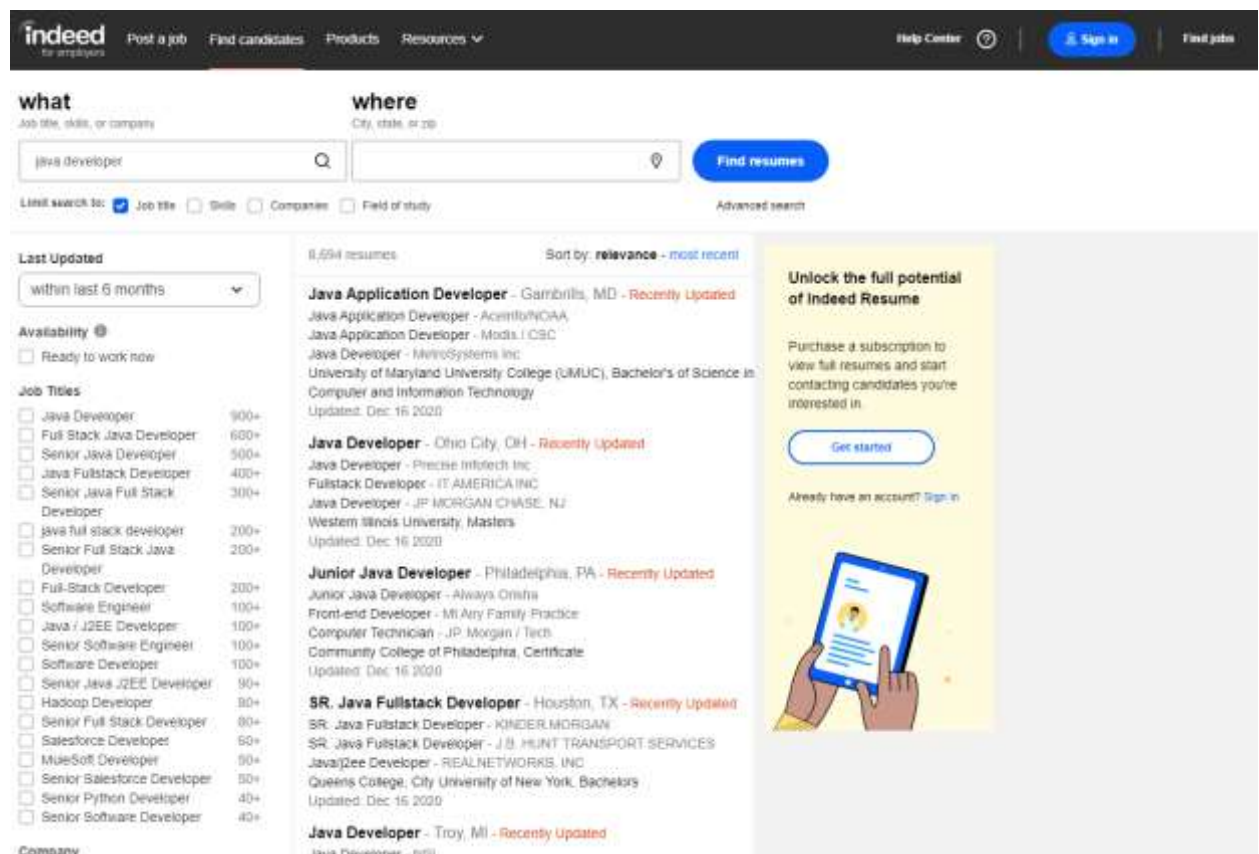


Рисунок 1.2 – Вебінтерфейс Indeed

Сервіс надає списки посад для тих, хто шукає роботу в будь-якій галузі, на кожному рівні, від початківця до керівника, та для кожного режиму роботи (позаштатний, неповний робочий день, стажування, повний робочий день) [6].

Після реєстрації в програмі можна отримувати сповіщення по електронній пошті про розміщення нових робочих місць, завантажувати своє резюме для швидшого заповнення заявок та отримувати повідомлення від рекрутерів та майбутніх роботодавців, також пропонується інструмент порівняння заробітної плати та розділ про огляди компаній, можна прочитати відверті думки щодо потенційних роботодавців перед тим, як подати заявку на участь або прийняти пропозицію [6].

Важливим джерелом привертання уваги до сучасних програмних засобів пошуку вакансій є засоби створення рекомендацій. Будь-який крок до персоналізації призводить до того, що потенційно більша кількість здобувачів буде використовувати цей програмний засіб.

Достатньо особливим сервісом є Snagajob (рис. 1.3), який дозволяє знаходити роботу з погодинною оплатою: надається можливість виконувати погодинні завдання в галузі охорони здоров'я, обслуговування клієнтів, гостинності, роздрібних продажів, безпеки та доставки їжі, шукачі роботи можуть шукати віддалені або локальні роботи, після реєстрації можна увімкнути отримання електронних повідомлень про нові списки вакансій, які відповідають інтересам користувача [6].

Особливістю виконання пошуку в даному сервісі є те, що він повністю ґрунтується на геолокації, іншого базового варіанту застосунок не має. Відповідно для того, щоб виконати пошук, користувачу потрібно задати своє розташування, а пошук виконується в заданому радіусі для цієї позиції. Оскільки пошук відбвається саме погодинної роботи, то використання такого базового фільтру є достатньо зрозумілим засобом.

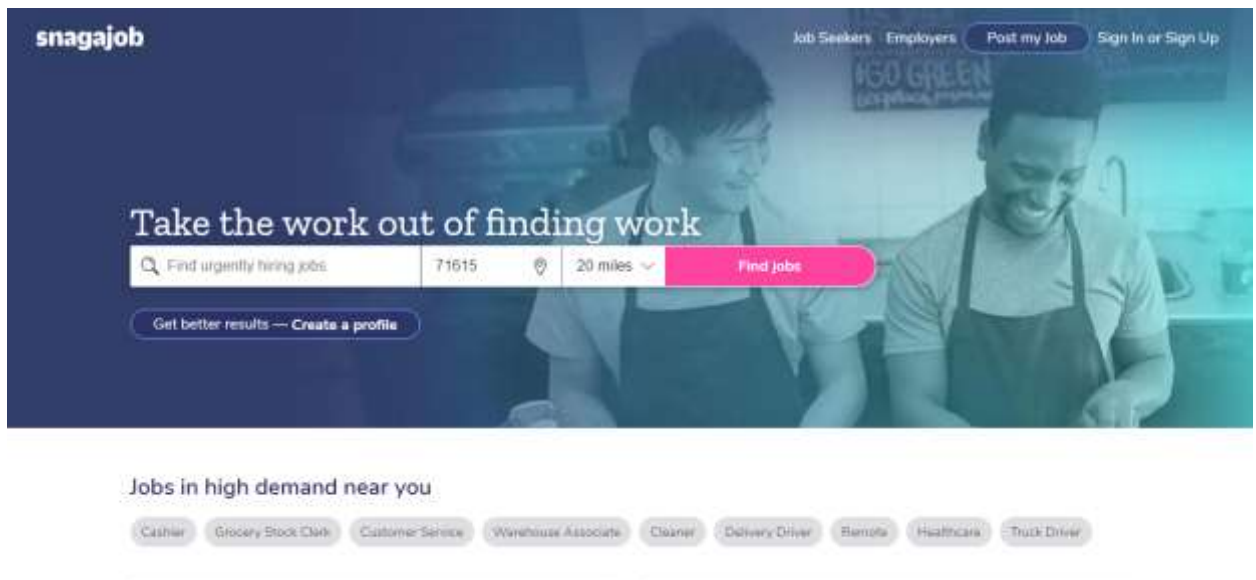


Рисунок 1.3 – Вебінтерфейс Snagajob

1.3 Висновки за розділом 1

Виконано огляд проблеми пошуку вакансій для працевлаштування. Виділено основні шляхи для виконання пошуку вакансій для працевлаштування. Виконано огляд програмного забезпечення пошуку вакансій для працевлаштування, серед яких розглянуто сервіси rabota.ua, Indeed, Snagajob.

До завдань, які мають бути виконані в дипломній кваліфікаційній роботі бакалавра, належать зокрема наступні:

- розробка методу пошуку вакансій для працевлаштування;
- проєктування програми;
- розробка програми;
- розробка організаційно-економічної частини;
- розробка заходів з охорони праці та цивільної безпеки.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Визначення задачі пошуку вакансій для працевлаштування

Задача пошуку вакансій для працевлаштування полягає в знаходженні вакансій, які є подібними на ту, яку передивляється користувач.

Для цього потрібно визначити набір ознак, якими характеризується кожна вакансія. Тоді задача пошуку вакансій для працевлаштування полягає у визначенні множини вакансій V , що складається з вакансій v_i , що за власними значеннями ознак є найбільш подібними до вхідної вакансії v_r , що характеризується набором значень, які задають значення ознак.

Тоді для розв'язання цієї задачі потрібно визначити наступне:

- метод або модель, які будуть визначати функцію, що встановлює описану залежність;
- набір ознак, якими характеризуються вакансії.

2.2 Модель Word2vec

Вбудовування слів – це тип подання слів, що дозволяє зрозуміти слова з подібним значенням за допомогою алгоритмів машинного навчання, це відображення слів у вектори дійсних чисел за допомогою нейронної мережі, імовірнісної моделі або зменшення розмірності на матриці спільного виникнення слів, це моделювання мови та техніка навчання, це також спосіб виконання зіставлення за допомогою нейронної мережі [7].

Вбудовування слів допомагає у створенні функцій, кластеризації документів, класифікації тексту та завданнях з обробки природної мови, що загалом охоплює наступні завдання:

- знайти подібні слова: вбудовування слів використовується, щоб запропонувати слова, подібні до слова, що подано в модель прогнозування;

- створити групу споріднених слів: використовується для семантичного групування, яке об'єднує подібні характеристики разом, а несхожі розташовує віддалено;

- характеристика для класифікації тексту: текст відображається в масиви векторів, які подаються до моделі для навчання, а також прогнозування;

- кластеризація документів;

- обробка природної мови: є багато застосувань, де вбудовування слів є корисним і перемагає над засобами витягання особливостей, такими як теги частин мови, сентиментальний аналіз та синтаксичний аналіз [7]-[8].

Модель word2vec ґрунтується на наступних принципах:

- word2vec схожа на автокодер, кодуючи кожне слово у векторі, але замість того, щоб навчатися за входами шляхом реконструкції, як це робить обмежена машина Больцмана, word2vec вивчає слова проти інших слів, які є сусідами їх у корпусі введення, робить це одним із двох способів (або використовуючи контекст для прогнозування цільового слова (метод, відомий як безперервна сумка слів), або використовуючи слово для прогнозування цільового контексту, яке називається skip-gram), на великих наборах даних використовується останній метод, оскільки він дає точніші результати;

- вхідне або контекстне слово – це один one-hot вектор розміром V , прихований шар містить N нейронів, а вихід – це знову ж таки вектор довжиною V , елементи якого є значеннями softmax;

- нейрони прихованого шару просто копіюють зважену суму входів на наступний шар, немає такої функції активації, як сигмоїдна, гіпорболічна функція або ReLU, єдина нелінійність – це розрахунки softmax на вихідному шарі;

- у word2vec використовується розподілене представлення слова: якщо взяти вектор із кількома сотнями вимірів, кожне слово представлено розподілом ваги між цими елементами, отже, замість однозначного

відображення між елементом у векторі та словом, подання слова розподілено по всіх елементах у векторі, і кожен елемент у векторі сприяє визначенню багатьох слів, такий вектор виконує прохід, щоб якимось абстрактним чином відобразити значення слова, при цьому якщо просто дослідити великий корпус, можна дізнатися вектори слів, які здатні фіксувати зв'язки між словами виразно [9]-[10].

2.3 Метод k-найближчих сусідів

Метод k-найближчих сусідів – це один з найпростіших алгоритмів машинного навчання, заснований на техніці контрольованого навчання, він відзначається наступним:

- метод k-найближчих сусідів передбачає схожість між новим випадком або даними і наявними випадками та відносить новий випадок до категорії, яка найбільш схожа на наявні категорії;

- метод k-найближчих сусідів зберігає всі наявні дані та класифікує нову точку даних на основі подібності – це означає, що коли з'являються нові дані, їх можна легко класифікувати, відносячи до однієї з категорій за допомогою методу k-найближчих сусідів;

- метод k-найближчих сусідів може бути використаний як для регресії, так і для класифікації, але здебільшого він використовується для задач класифікації;

- метод k-найближчих сусідів – це непараметричний алгоритм, що означає, що він не робить жодних припущень щодо базових даних;

- метод називають алгоритмом ледачого учня, оскільки він не вчиться з навчального набору відразу, натомість зберігає набір даних і під час класифікації виконує дію над набором даних;

- метод k-найближчих сусідів на етапі навчання просто зберігає набір даних, і коли він отримує нові дані, то класифікує ці дані за категорією, багато в чому подібною до нових даних [11].

По суті метод k -найближчих сусідів складається з наступних кроків:

- обрати число K сусідів;
- обчислити євклідову відстань для K сусідів;
- взяти K найближчих сусідів відповідно до розрахованої євклідової відстані;
- серед цих K сусідів підрахувати кількість точок даних у кожній категорії;
- призначити нові точки даних тій категорії, для якої кількість сусідів максимальна;
- в результаті модель готова [11].

Незважаючи на свою простоту, даний метод дуже активно використовується на практиці.

Метод k -найближчих сусідів часто використовується в простих системах рекомендацій, технології розпізнавання зображень та моделях прийняття рішень, саме такі алгоритми використовують компанії, такі як Netflix або Amazon, щоб рекомендувати різні фільми для перегляду або книги для покупки, в якості даних використовується набір даних, зібраних про фільми, які дивились користувачі, або книги, які були придбані на вебсайті, потім ці компанії вводять доступні дані про клієнтів і порівнюють їх з іншими клієнтами, які переглядали подібні фільми або купували подібні книги, потім ця точка даних буде класифікована як певний профіль на основі минулого за допомогою методу k -найближчих сусідів, потім рекомендовані фільми та книги залежатимуть від того, як алгоритм класифікує цю точку даних [12].

На результати роботи методу значно впливають наступні фактори:

- функція відстані або метрика відстані, що використовується для визначення найближчих сусідів;
- правило прийняття рішення, яке використовувалося для отримання класифікації від найближчих сусідів;

– кількість сусідів, що використовувались для класифікації нового приклад [13].

Загалом метод знаходить застосування в наступних областях:

- фінанси – під час прогнозування кредитного рейтингу клієнтів;
- охорона здоров'я;
- політологія – під час реалізації визначення, до якого класу буде належати конкретний виборець: до тих, що буду голосувати, чи ні;
- визначення рукописного тексту;
- розпізнавання зображень;
- розпізнавання відео;
- розпізнавання образів [13].

2.4 Метод пошуку вакансій для працевлаштування

Розроблений метод пошуку вакансій для працевлаштування ґрунтується на знаходженні подібних вакансій, щоби здобувач в залежності від того, що його цікавить, міг отримати подібні в просторі ознак вакансії.

Даний метод є інструментом підтримки прийняття рішень в тому сенсі, що він надає варіант для прийняття рішень, в такому випадку кандидат на працевлаштування може не переглядати всі наявні вакансії, а обирати для перегляду ті, які було отримано в результаті роботи методу, вивчати їх докладно та обирати потрібну. В результаті це звужує простір пошуку.

Простір ознак формується з двох складових. Першими складовими є стандартні виділені з вакансії характеристики, які зокрема включають:

- вид зайнятості;
- рівень посади;
- вік;
- стать;
- розмір очікуваної зарплати;
- досвід роботи;

- володіння мовами;
- галузь знань;
- рівень освіти;
- місто.

Всі ці ознаки є базовими для визначення вакансії. Але окрім того кожна вакансія містить текстові описи, які також мають потрібну інформацію всередині себе, але ніяк не визначаються в даному наборі або визначаються тільки частково. Тому для роботи методу запропоновано додатково використовувати ознаку, що представляє закодоване представлення текстового опису вакансії.

На першому етапі роботи методу реалізується виконання кодування текстових даних для кожної вакансії за допомогою моделі word2vec.

На другому етапі формується набір значень, кожне з яких характеризує відповідну вакансію і характеризується зі сформованого набору описаних вище ознак і кодового представлення тексту вакансії після виконання нормалізації.

На наступному етапі формується вибірка даних, що представляє вакансії, які зацікавили користувачів. Кожен екземпляр – це набір вакансій, які зацікавили користувача з номером, що відповідає номеру екземпляра.

Під зацікавленістю розуміється те, що кандидат подав запит на працевлаштування.

Отримані дані використовуються для попереднього навчання моделі.

При цьому робота відбувається окремо для кожної категорії.

На наступному етапі подається нова вакансія, для якої визначається шляхом застосування методу k-найближчих сусідів вакансії, які можуть зацікавити здобувача, який подав запит на вхідну вакансію.

Найцікавішою ідеєю методу k-найближчих сусідів для пошуку вакансій на працевлаштування є обчислення схожості, що визначається в даному методі.

Обчислення міри схожості запропоновано виконувати на основі косинусної міри. Косинусна міра розраховується на основі формули (2.1):

$$s_{i,j}^g = \cos(m_i^g m_j^g), \quad (2.1)$$

де $s_{i,j}^g$ – подібність за матрицею, що представляє набір значень ознак, які були описані вище;

m_i^g – набір значень ознак для i -ої вакансії;

m_j^g – набір значень ознак для j -ої вакансії.

За розрахованими значеннями визначаються K найближчих сусідів, на основі приналежності яких до виділеного класу і визначається, до якого класу належить дана вакансія. Відповідно в результаті найближчі вакансії з того ж самого класу і видаються як результат роботи методу.

2.5 Висновки за розділом 2

Розглянуто задачу пошуку вакансій для працевлаштування. Проаналізовано метод k -найближчих сусідів, принципи його застосування. Проаналізовано принципи роботи моделі word2vec.

Визначено власний метод пошуку вакансій для працевлаштування. В якості міри схожості використано косинусну подібність.

Наукова новизна роботи полягає в розробленому методі пошуку вакансій для працевлаштування, що виконує пошук подібних вакансій на основі їх класифікації за допомогою методу k -найближчих сусідів, використовуючи в якості додаткових ознак закодовані за допомогою моделі word2vec значення, що дозволяє обґрунтовано приймати рішення про вибір вакансій або резюме для детального аналізу.

3 ОПИС ПРОГРАМИ

3.1 Вимоги до програми

Основними вимогами до програми є функціональні вимоги, адже від розуміння виконуваних функцій програмою залежить те, які саме сутності мають бути реалізовані, а відповідно це база даних, розроблені класи, а також обрані засоби розробки.

На рис. 3.1 представлено варіанти використання програми менеджером.

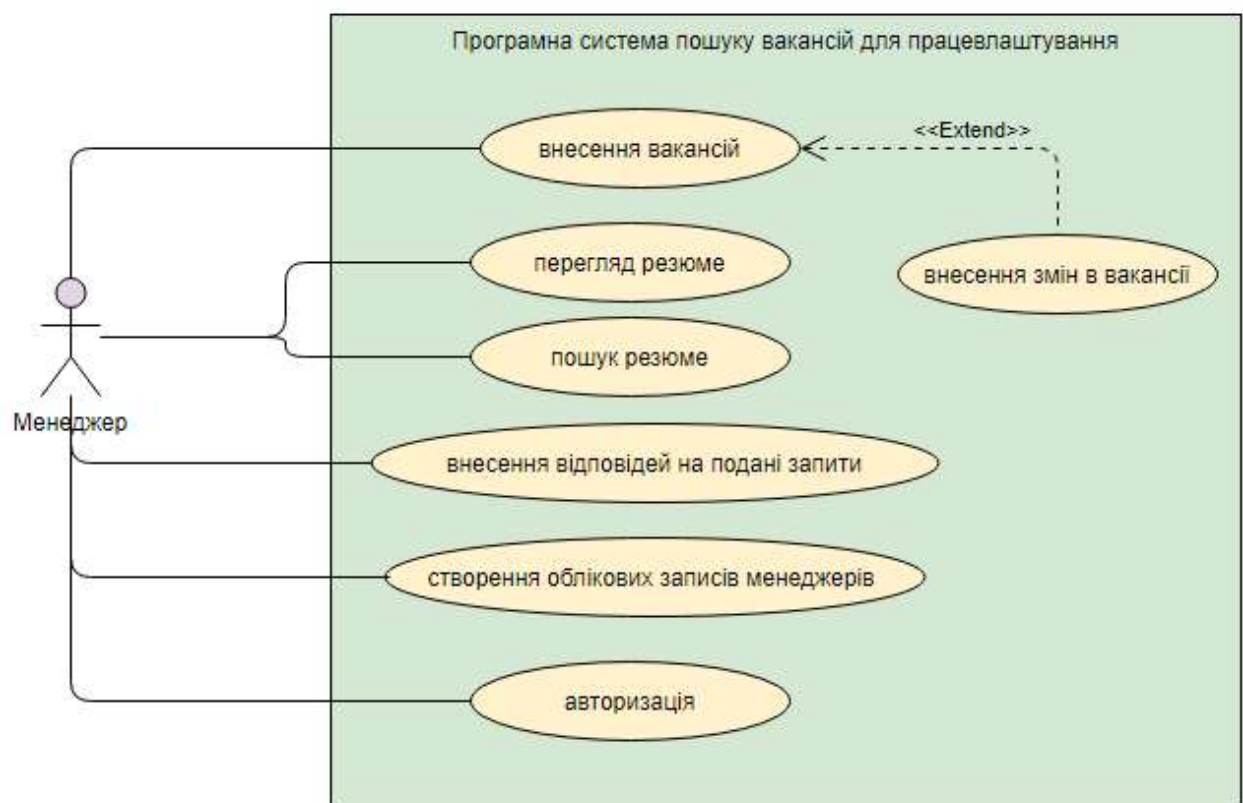


Рисунок 3.1 – Діаграма варіантів використання програми менеджером

Програма повинна реалізовувати наступні функціональні можливості:

- вносити дані наявних вакансій на працевлаштування;
- створювати резюме кандидатів;
- переглядати вакансії на працевлаштування;
- виконувати пошук вакансій;

- підбирати вакансії;
- вносити зміни в вакансії;
- вносити запити на працевлаштування;
- вносити відповіді на подані запити;
- реєструвати облікові записи кандидатів;
- створювати облікові записи менеджерів компанії;
- авторизувати облікові записи;
- вносити дані про попередню роботу та освіту кандидата;
- переглядати найбільш популярні за запитами на працевлаштування компанії.

На рис. 3.2 представлено варіанти використання програми кандидатом.

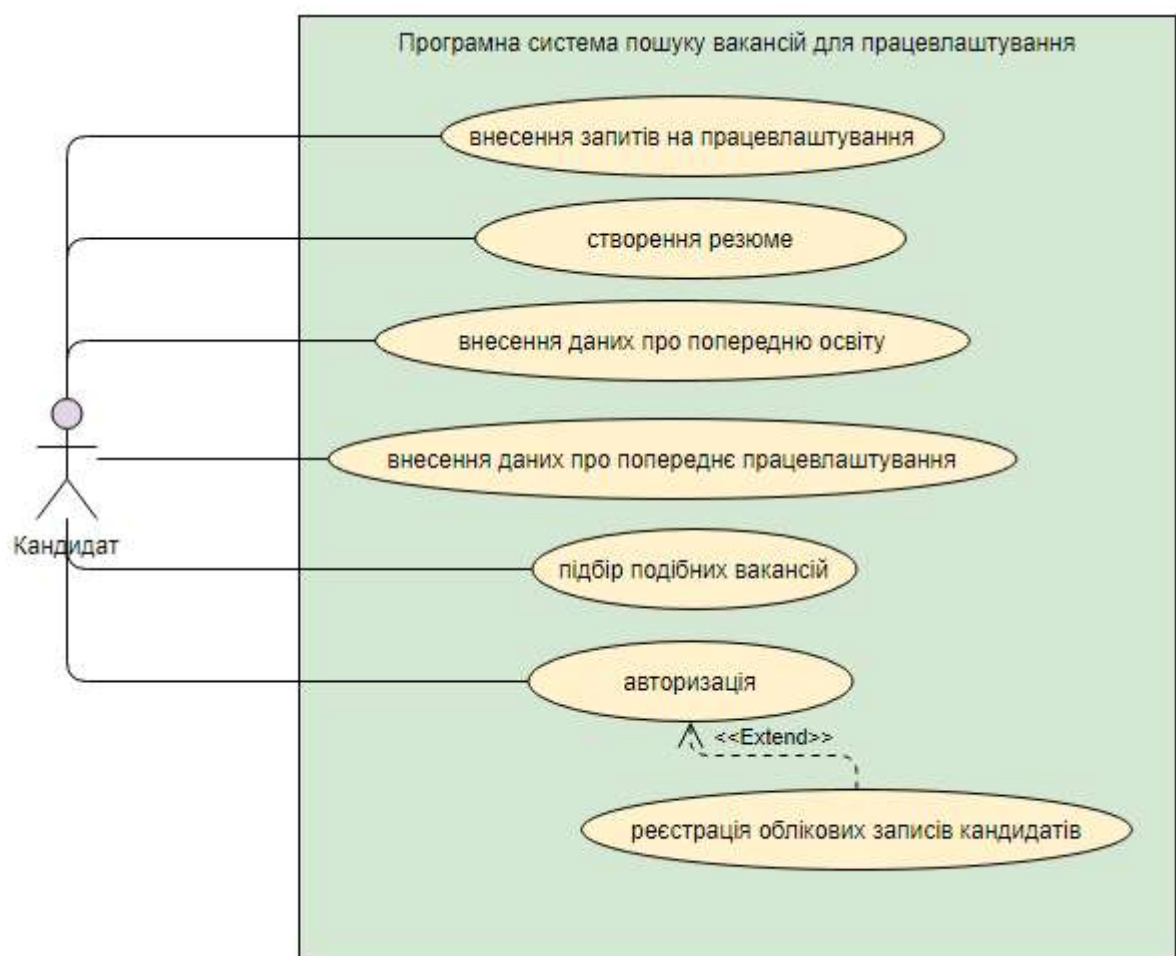


Рисунок 3.2 – Діаграма варіантів використання програми кандидатом

3.2 Вибір засобів і середовища проєктування і розроблення

Основні найактуальніші засоби пошуку вакансій для працевлаштування створені у вигляді вебсервісів. Тому і дане рішення було прийнято розробляти як вебзастосунок, до того ж вебзастосунок буде обробляти запити до сервера клієнтів, що знаходяться на різній віддаленості, що відповідає завданням предметної області.

У табл. 3.1 зведено результати порівняння мов програмування PHP і Python [14]-[16], які є на даний момент одними з найпопулярніших засобів веброзробки.

Таблиця 3.1 – Порівняння мов програмування

Критерій	PHP	Python
Бібліотеки	Загального призначення	Велика підтримка спільноти
Засоби роботи з даними	Присутні	Велика кількість бібліотек
Веб-програмування	Зручне з наявністю фреймворків	Зручне з наявністю фреймворків
Гнучкість засобів програмування	Гнучкий рівень	Вищий рівень

В результаті у якості мови програмування було обрано Python, оскільки вона має більш гнучкі засоби програмування, велику кількість бібліотек для роботи з даними, до того ж вебрішення, створені цією мовою, достатньо якісні.

Тож дану роботу буде присвячено в подальшому розробці вебзастосунку мовою програмування Python.

3.3 Проєктування бази даних

Системою керування базою даних було обрано MySQL через те, що вона є безкоштовною і при цьому повноцінним потужним рішенням.

Створені в результаті проєктування таблиці бази даних охоплюють наступні сутності:

- Vacancy – вакансія;
- Candidate – кандидат;
- Category – вакансія;
- Company – компанія;
- Request – запит на працевлаштування;
- JobExperience – попередня робота;
- Education – освіта;
- CV – резюме;
- User – користувач;
- CompanyUser – менеджер компанії;
- Requirements – вимоги до кандидатів;
- CandidateRequirements – значення вимог, що відповідають кандидату;
- VacancyRequirements – значення, задані в якості мінімальних для вимог вакансії.

Всі вимоги розділено на два різні представлення: з одного боку це вимоги, які в текстовому вигляді зберігаються в вакансії, а з іншого боку – це структуровані дані, які виділені окремо у вигляді характеристик і значень в Requirements і VacancyRequirements.

Усіх користувачів розділено на User та CompanyUser, оскільки менеджери компаній мають особливі характеристики, що визначають права їх роботи з даними компанії, окрім того компанія може мати багато таких облікових записів, відповідно всі інші варіанти можна вважати такими, що програють.

Таблиця Vacancy містить поля:

- Id – порядковий номер;
- Position – посада;
- Company_id – компанія;
- Salary – зарплатня;
- RequirementText – вимоги до кандидата;
- FullInfo – повний опис;
- Category_id – категорія вакансії;
- File – файл вакансії;
- Created – дата створення або оновлення вакансії;
- City – місто.

Поле Company_id визначає зовнішній ключ, необхідний для позначення того, якій компанії належить вакансія.

Таблиця Candidate містить поля:

- Id – порядковий номер;
- User_id – користувач системи;
- Name – повне ім'я;
- Phone – номер телефону;
- Email – електронна поштова адреса;
- Site – вебсайт за наявності зі сторінкою кандидата;
- Photo – фотокартка;
- EducationShort – рівень освіти;
- FullInfo – опис кандидата.

Поле User_id визначає зовнішній ключ, що визначає обліковий запис користувача, під яким працює даний кандидат.

Таблиця Company містить поля:

- Id – порядковий номер;
- Name – назва;
- Phone – номер телефону;
- Email – електронна поштова адреса;

- Site – вебсайт;
- Photo – зображення;
- FullInfo – опис компанії.

Таблиця Request містить поля:

- Id – порядковий номер;
- Candidate_id – кандидат;
- Vacancy_id – вакансія;
- RText – текст запиту від кандидата;
- RFile – шлях до файлу, поданого в якості рецензії;
- Answer – відповідь на запит від компанії;
- GetDate – дата і час подання запиту;
- Status – стан.

Поле Candidate_id визначає зовнішній ключ кандидата, який виконав запит на вакансію.

Поле Vacancy_id визначає зовнішній ключ вакансії, на яку було подано запит.

Таблиця JobExperience містить поля:

- Id – порядковий номер;
- Candidate_id – кандидат;
- StartDate – дата початку роботи;
- EndDate – дата завершення роботи;
- Company_id – компанія;
- Position – посада;
- Address – адреса;
- FullInfo – повний опис результатів.

Поле Candidate_id визначає зовнішній ключ кандидата, який працював за цією позицією.

Поле Company_id визначає зовнішній ключ, необхідний для позначення того, в якій компанії працював кандидат.

Таблиця Education містить поля:

- Id – порядковий номер;
- Candidate_id – кандидат;
- StartDate – дата початку навчання;
- EndDate – дата завершення навчання;
- Institution – навчальний заклад;
- Speciality – спеціальність;
- Qualify – отримана кваліфікація.

Поле Candidate_id визначає зовнішній ключ кандидата, який отмував дану освіту.

Таблиця CV містить поля:

- Id – порядковий номер;
- Candidate_id – кандидат;
- SearchPosition – посада, на яку претендує кандидат;
- SearchSalary – зарплатня, на яку претендує кандидат;
- FullInfo – повний опис;
- File – файл резюме, що додається.

Поле Candidate_id визначає зовнішній ключ кандидата, якому належить дане резюме.

Таблиця CompanyUser містить поля:

- Id – порядковий номер;
- User_id – користувач системи;
- Company_id – компанія;
- Rights – права на роботу з даними компанії;
- Phone – номере телефону;
- Email – електронна поштова адреса користувача.

Поле Company_id визначає зовнішній ключ компанії, якій належить даний обліковий запис користувача.

Поле User_id визначає зовнішній ключ користувача, для якого встановлюється дана залежність.

Схема бази даних (рис. 3.3) визначає відношення між таблицями.

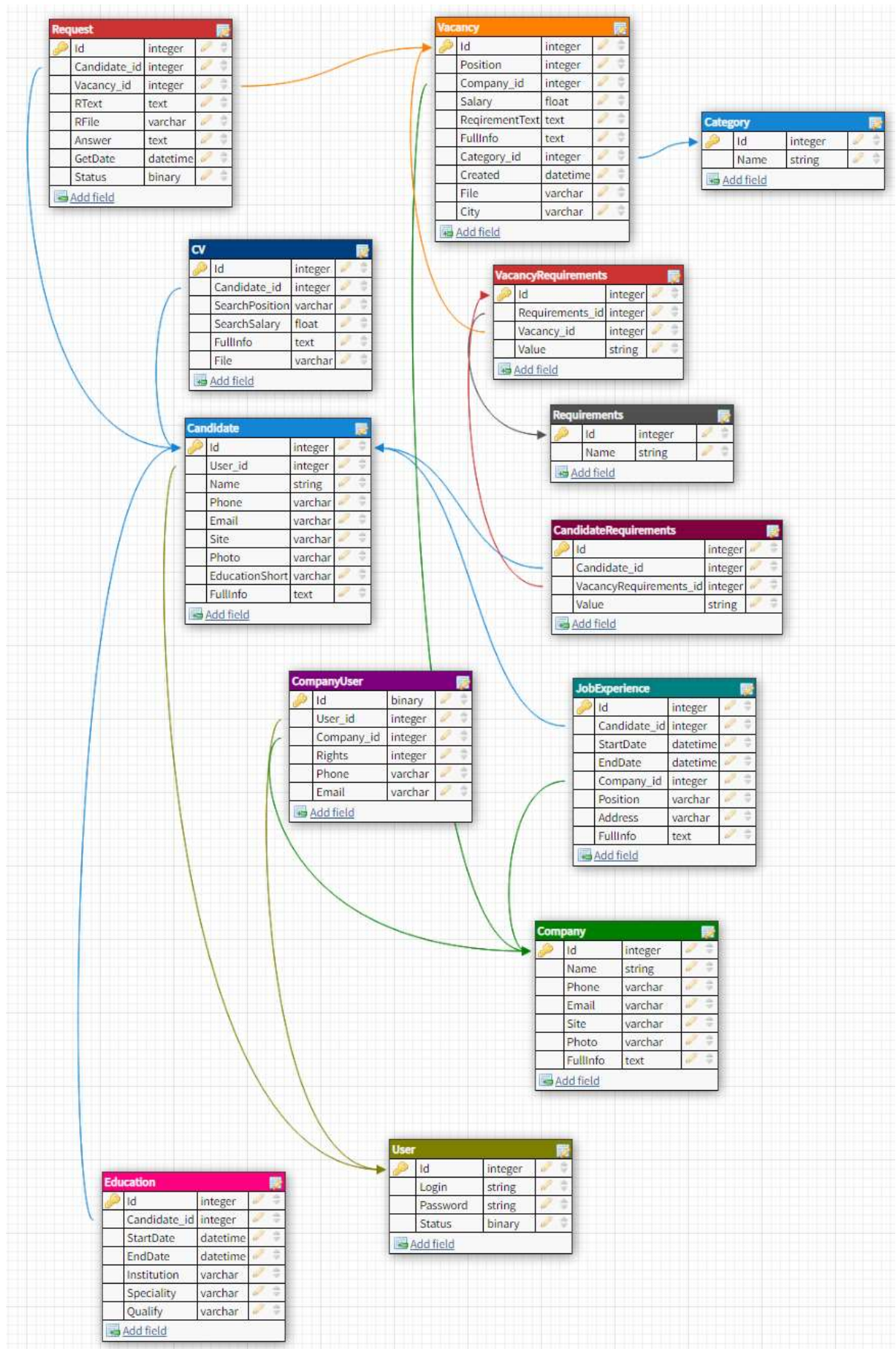


Рисунок 3.3 – Схема бази даних

Таблиця Category містить поля:

- Id – порядковий номер;
- Name – назва категорії.

Таблиця User містить поля:

- Id – порядковий номер;
- Login – логін;
- Password – пароль;
- Status – тип запису.

Таблиця Requirements містить поля:

- Id – порядковий номер;
- Name – вимога.

Таблиця VacancyRequirements містить поля:

- Id – порядковий номер;
- Requirements_id – вимога;
- Vacancy_id – вакансія;
- Value – значення.

Таблиця CandidateRequirements містить поля:

- Id – порядковий номер;
- Requirements_id – вимога;
- Candidate_id – кандидат;
- Value – значення.

Сутності було розроблено для забезпечення необхідності як структуризації параметрів у вигляді вимог, що представляються в вакансіях. При цьому в таблиці Requirements зокрема мають зберігатися ознаки, які було перелічено в розділі 2.

3.4 Структура програмного проєкту

Структуру програмного проєкту побудовано на основі постійної взаємодії підсистеми пошуку вакансій та вебзастосунку. Комунікація їх

відбувається як між собою під час виконання пошуку за вимогами користувача, так і з сервером бази даних за рахунок програмних інтерфейсів звернення до бази даних.

На рис. 3.4 представлено діаграму компонентів розгортання програмної системи в межах клієнт-серверної моделі.

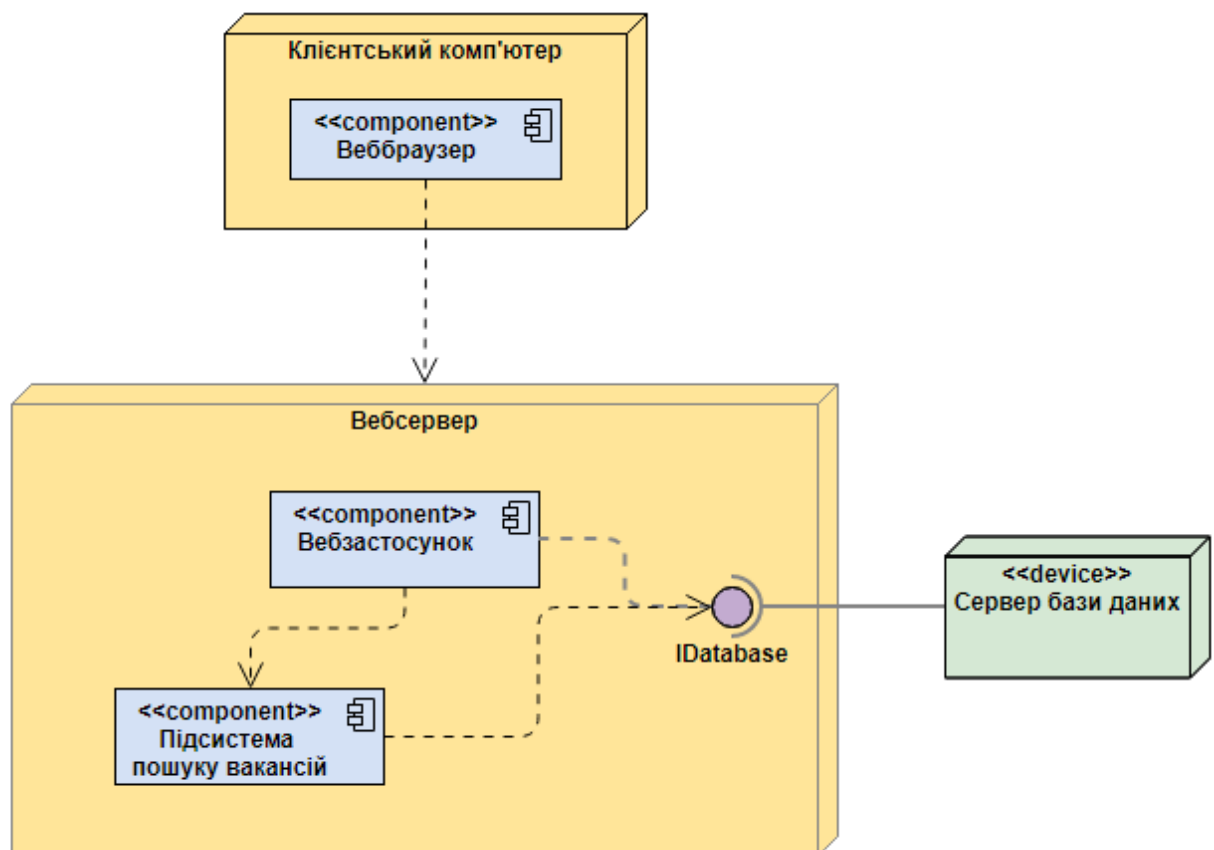


Рисунок 3.4 – Діаграма компонентів програмної системи

3.5 Висновки за розділом 3

Розглянуто основні сценарії використання системи користувачами на основі побудови діаграм варіантів використання для менеджера та кандидата на працевлаштування.

Вирішено проєктувати і реалізовувати програмне забезпечення в якості вебзастосунку в основі. Для реалізації обрано мову програмування Python.

Для забезпечення роботи з базою даних використано систему MySQL. Визначено таблиці бази даних, встановлено зв'язки між ними, побудовано схему бази даних. Визначено структуру програмного проекту з побудовою відповідної діаграми компонентів.

4 РОЗРОБКА ПРОГРАМИ

4.1 Опис логіки функціонування програми

Робота програми розпочинається зі звернення клієнтського застосунку до вебсервера, який у відповідь надсилає сторінку з відображенням переліку вакансій.

Функціонування програми побудовано на розподілі логіки роботи між кандидатами на працевлаштування та менеджерами компаній. В залежності від ролі в системі кожен з них отримує свою функціональність. При цьому для того щоб виконати деякі дії може знадобитися авторизуватися. Неавторизований користувач не може знайти подібні вакансії, оскільки він не може подати власний запит на працевлаштування.

Схему на рис. 4.1 побудовано на основі даної логіки, але з певними спрощеннями, тому що кожна з дій, представлених на схемі вимагає деякої сукупності дій, тобто схему можна було б глибше деталізувати, але логіка її побудови заключалася в тому, щоб відобразити основний принцип взаємодії користувача з вебзастосунком.

Після обрання ролі, тобто вибору, який саме варіант вебзастосунку буде відображатися, далі надається повний набір функцій через меню. Кожен користувач може обирати будь-яку з них стільки разів, скільки потрібно, після цього він може або змінити режим, або завершити роботу з програмою.

Така логіка взаємодії дозволила врахувати той факт, що більшість дій у менеджера та кандидата відрізняється, але з іншого боку кожному користувачу може знадобитися і та функціональність, і та, відповідно сенсу створювати два окремі вебзастосунки не було. Коригування доступу відбувається за рахунок авторизації.

Кандидат може працювати тільки з власним резюме і то тільки після авторизації. Але він може переключитися в режим менеджера і отримати доступ до переліку всіх резюме.

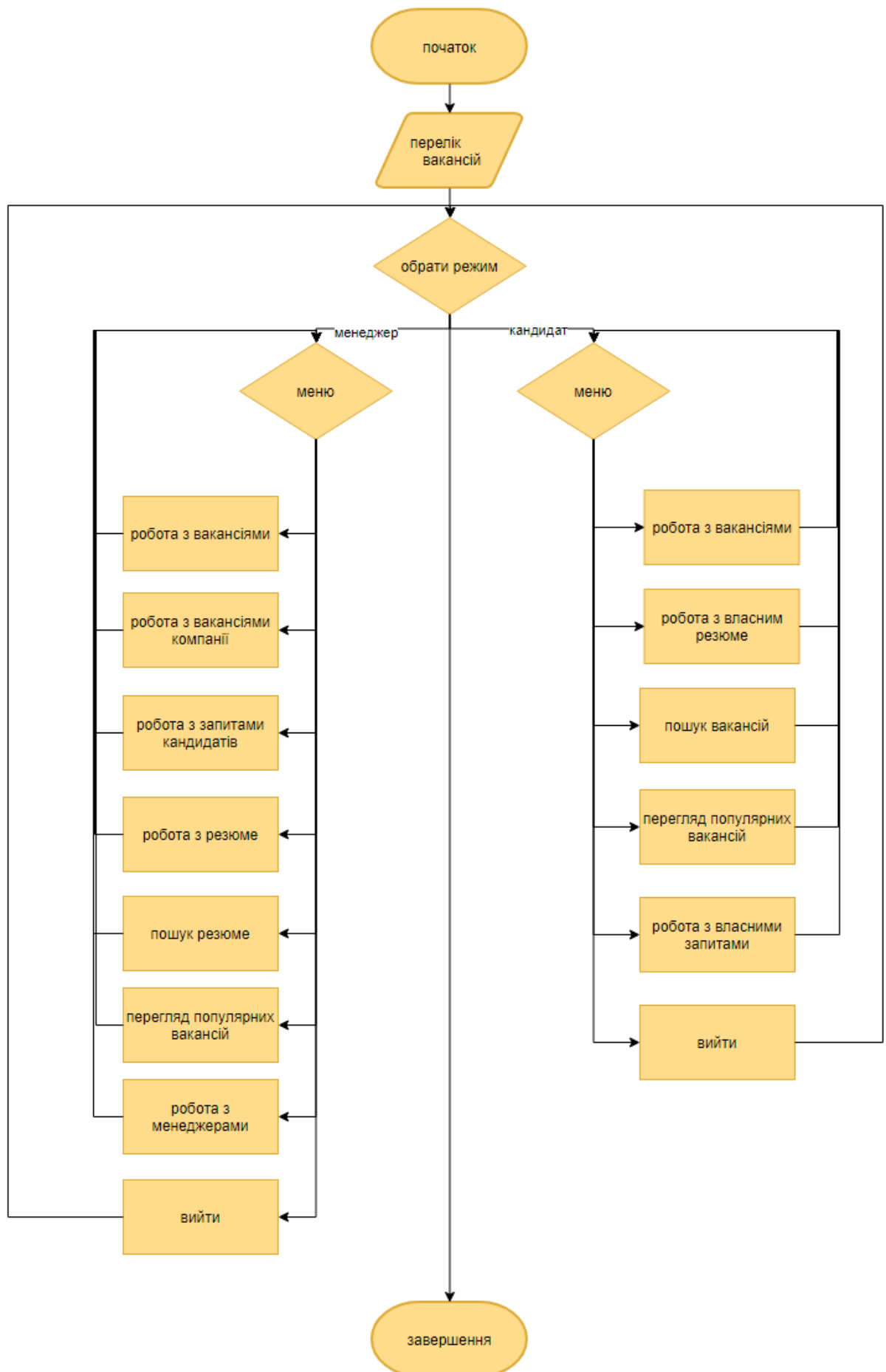


Рисунок 4.1 – Схема функціонування програми

4.2 Опис реалізації вебзастосунку

Вебзастосунок реалізовано за допомогою вебфреймворку Flask, який надає вбудований сервер розробки та швидкий налагоджувач, інтегровану підтримку модульного тестування, шаблонізатор Jinja2, реалізовану підтримку Unicode, Flask надає багато контролю на стадії розробки проєкту, загалом даний вебфреймворк дотримується принципів мінімалізму [17].

На основі його використання було створено вебсторінки для організації функціональності, описаної в розділі 3. Вебсторінки зокрема включають:

- addvacancy.html – додавання нової вакансії компанією;
- editvacancy.html – зміна вакансії;
- vacancies.html – перелік вакансій;
- cv.html – резюме кандидата;
- addcv.html – додавання власного резюме;
- searchvac.html – пошук вакансій;
- searchcv.html – пошук резюме;
- similar.html – відображення подібних вакансій, де відображаються результати, сформовані за допомогою розробленого методу;
- answer.html – створення відповіді на запит кандидата;
- cvs.html – перегляд переліку доступних резюме;
- rating.html – перегляд найбільш популярних вакансій;
- requests.html – перелік власних запитів кандидата;
- makerequest.html – створення запиту на роботу за вакансією.

Шаблонізація цих вебсторінок виконана за допомогою несення в код звернень наступним чином:

```
render_template('cvs.html', cvs=cvs)
```

Для забезпечення взаємодії з даними розроблено файл processor.py, який містить набір функцій, що забезпечують виконання запитів різних типів до бази даних. Звідси дані формуються для відображення в шаблонах сторінок, тут дані зберігаються в базу даних.

Розглянемо приклад основних дій, які виконуються в даному модулі.

Витягання даних вакансії забезпечується на основі відповідного SQL-запиту типу SELECT (на вибірку даних):

```
dbcurs = JobDataBase.cursor ()
dbcurs.execute ("SELECT * FROM Vacancy WHERE Vanacy.Id = " + str
(num))
result = dbcurs.fetchone ()
JobDataBase.commit ()
return result
```

Відповідно оновлення даних вакансії забезпечується шляхом виконання UPDATE-запиту до бази даних:

```
dbcurs = JobDataBase.cursor()
comm = "UPDATE Vacancy SET Position = %s, Company_id = %s,
Salary = %s, RequirementText = %s, FullInfo = %s, Category_id = %s, Created =
%s, File = %s, City = %s WHERE Id = %s"
data = (position, cid, salary, requirement, fullinfo, caid, created, file, city,
id)
dbcurs.execute(comm, data)
JobDataBase.commit()
```

Аналогічно підтримуються всі основні типи запитів до бази даних і доступ до них надається в результаті за допомогою відповідних програмних функцій.

4.3 Висновки за розділом 4

Описано логіку функціонування програми, побудовану схему функціонування на основі представлення меню вебзастосунку. Описано принципи розроблення вебзастосунку на основі використання вебфреймворку Flask. Розглянуто основні створені вебсторінки та принципи організації взаємодії з базою даних.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

5.1 Призначення програми

Програмне забезпечення призначене для виконання пошуку вакансій для працевлаштування здобувачами та пошуку резюме кандидатів представниками компаній (рекрутерами). Це забезпечується як наданням засобів для внесення таких даних, їх перегляду, так і наданням засобів пошуку, що включають як традиційний пошук, так і підбір подібних вакансій.

5.2 Умови виконання програми

Технічні засоби, на яких експлуатується програма, повинні включати наступний мінімальний набір:

- двоядерний процесор, частота не менше 2,4 ГГц;
- оперативна пам'ять – 4 Гб;
- не менше 2 Гб вільного простору на жорсткому диску.

Для експлуатації програми має бути виділено окремий сервер, через який буде відбуватися доступ до вебзастосунку. Окрім того на сервері має розташовуватися база даних та інстальована система керування базою даних MySQL, але для збереження їх може виділятися також окремий сервер бази даних.

5.3 Опис роботи користувача з програмою

Для виконання програми потрібно звернутися до вебзастосунку віддалено, вказавши адресу сервера.

На рис. 5.1 представлено варіант відображення вебсторінки для авторизованого в ролі кандидата користувача.

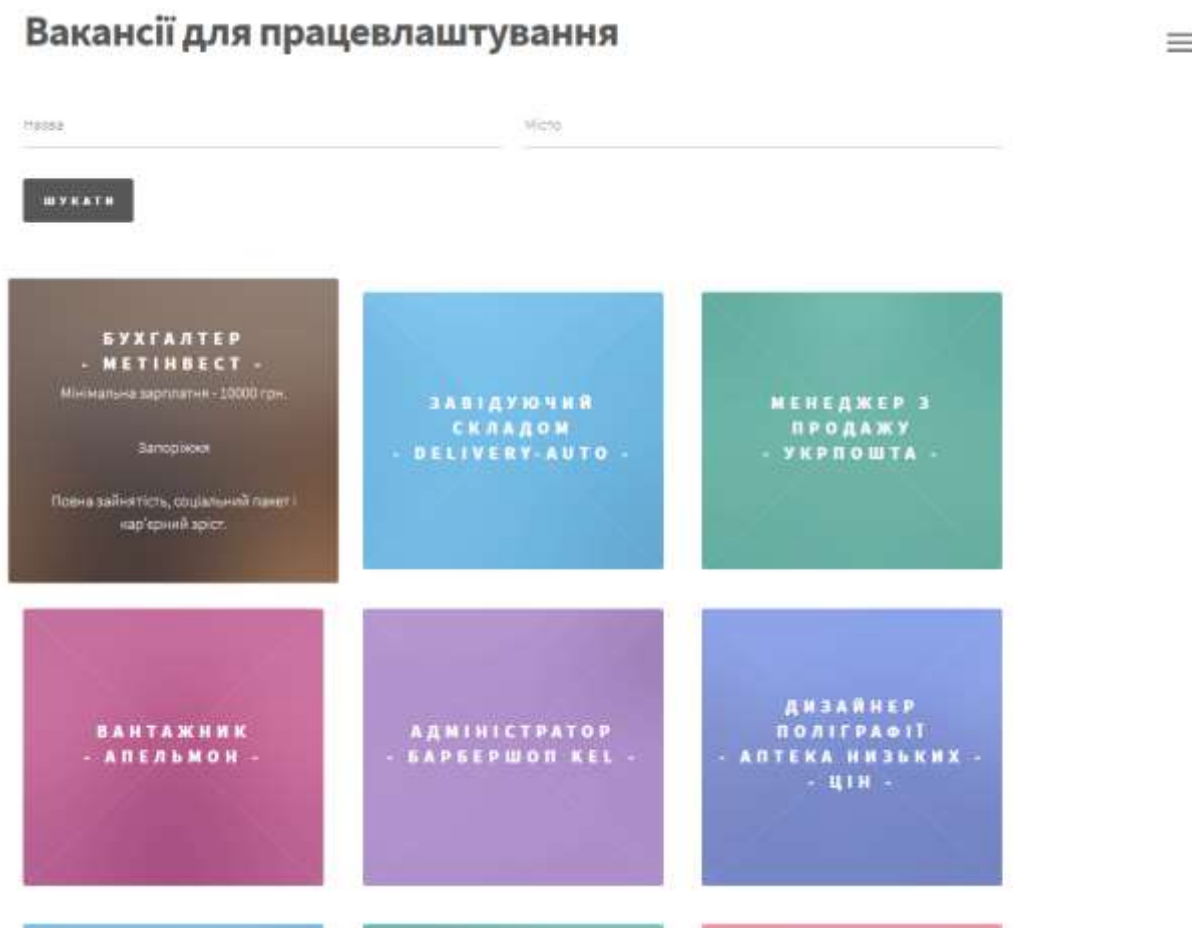


Рисунок 5.1 – Перегляд вакансій для працевлаштування

Після авторизації кандидату відкривається базова сторінка, на якій відображається перелік всіх доступних вакансій. Кожна вакансія відображається плиткою, на якій вказано посаду та компанію. Якщо навести курсор на плитку, то відобразиться повніша інформація про вакансію, зокрема мінімальна зарплатня, місто та перший абзац опису вакансії.

В правій стороні відображається меню, яке можна розкрити натисканням на іконку (рис. 5.2). Пункти меню для кандидата на працевлаштування (звичайного авторизованого користувача):

- пункт «Вакансії» дозволяє переглянути перелік всіх вакансій, як було розглянуто вище;
- пункт «Резюме» дозволяє переглянути власне резюме, змінити його, подати нову версію;

- пункт «Пошук» дозволяє виконати розширений пошук вакансій за набором параметрів;
- пункт «Популярні» дозволяє переглянути вакансії, за якими було подано найбільше заявок;
- пункт «Ваша вакансія» дозволяє переглянути вакансії, подібні на останню, на яку користувач подав заявку;
- пункт «Для роботодавця» дозволяє перейти в режим роботи роботодавця;
- пункт «Ваші запити» дозволяє переглянути запити, які було подано на працевлаштування авторизованим користувачем, та відповіді на них (за наявності);
- пункт «Вийти».

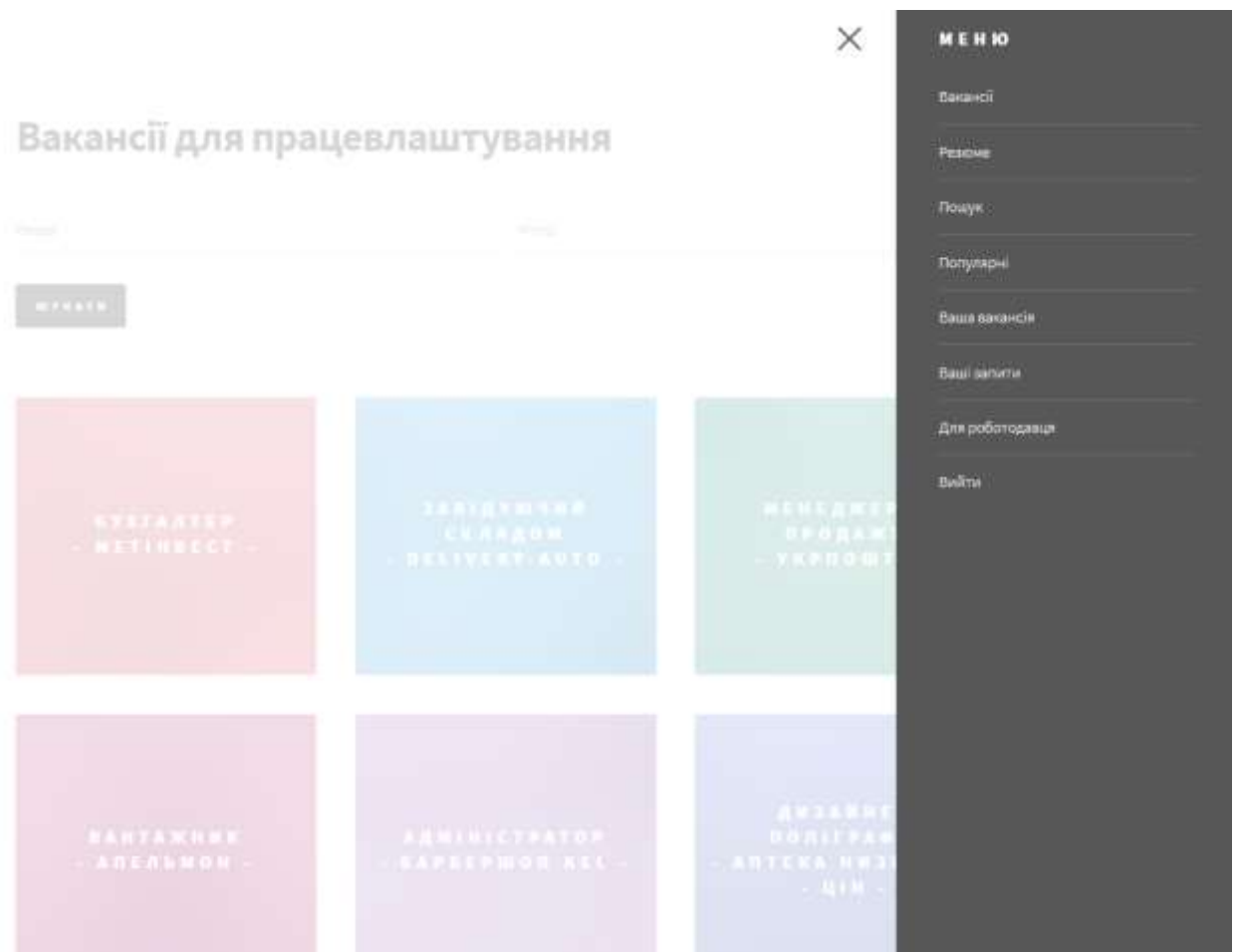


Рисунок 5.2 – Доступ до меню кандидатом

У варіанті роботи роботодавця меню відповідно включає наступні пункти:

- пункт «Вакансії» дозволяє переглянути перелік всіх вакансій на ринку;
- пункт «Власні вакансії» дозволяє переглянути вакансії, наявні у даній компанії, звідси їх також можна створювати;
- пункт «Запити» дозволяє створити відповіді на подані запити кандидатів;
- пункт «Резюме» дозволяє переглядати всі резюме на ринку;
- пункт «Пошук» дозволяє виконати розширений пошук резюме;
- пункт «Популярні» дозволяє переглянути вакансії, за якими було подано найбільше заявок;
- пункт «Менеджери» дозволяє керувати обліковими записами користувачів, що можуть працювати з даними компанії;
- пункт «Для кандидатів» дозволяє перейти в режим роботи кандидата (необхідна повторна авторизація);
- пункт «Вийти».

Під час відображення переліку всіх вакансій доступна також спрощена форма пошуку. Вона дозволяє визначити місто вакансії та посаду.

Форма пошуку вакансій в повному вигляді представлена на рис. 5.3.

Пошук вакансій

Посада

Місто

Мінімальна зарплата

☒ Останні

ШУКАТИ

Рисунок 5.3 – Форма пошуку вакансій

Для того щоб відкрити повну інформацію про вакансію, потрібно натиснути на неї і відкриється відповідна сторінка.

На рис. 5.4 наведено приклад результату пошуку вакансій.

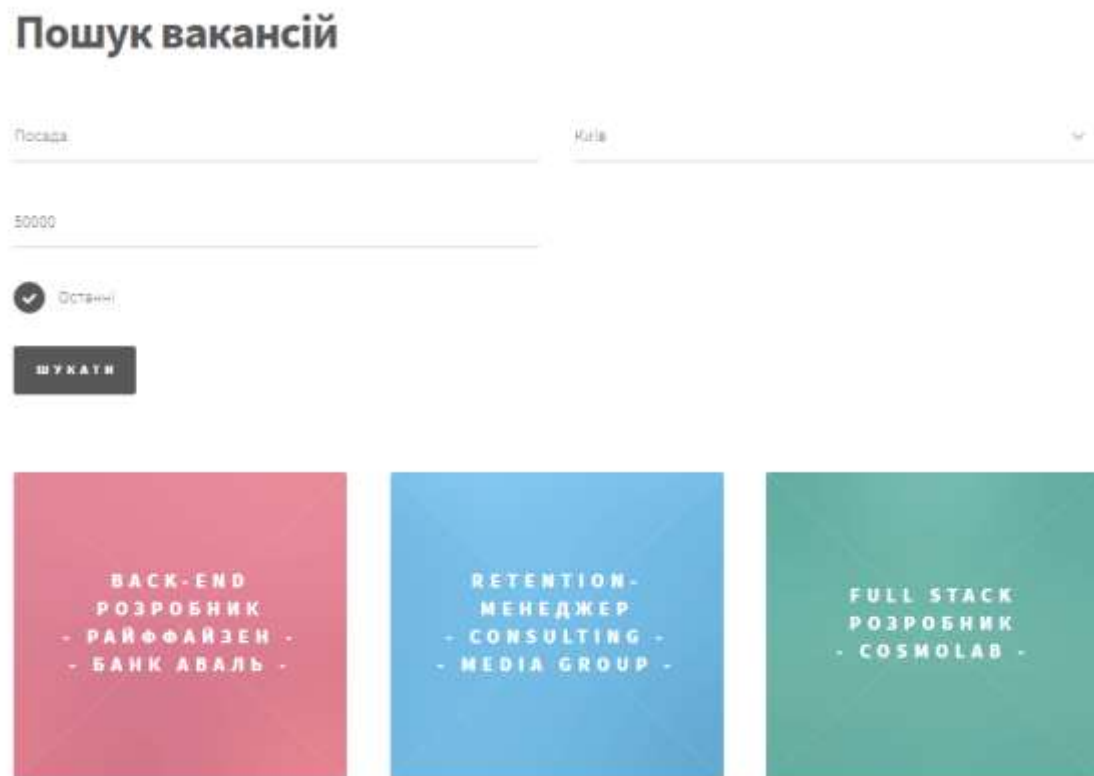


Рисунок 5.4 – Результат пошуку

У видачі розташовані тільки вакансії, мінімальна зарплатня за якими більша за вказану, зі вказаного міста.

У даній формі можна задати посаду, обрати місто з випадаючого переліку та вказати мінімальну зарплатню, за якою готовий працювати кандидат. Прапорець «Останні» призводить до видачі першими в результатах тих вакансій, які були додані пізніше. Окрім того не виводяться вакансії, дата створення або оновлення яких більше місяця.

На рис. 5.5 наведено приклад перегляду знайдених подібних вакансій. При цьому окрім переліку знайдених вакансій видаються останні вакансії, за якими було подано користувачем запит.

Останній поданий запит



Подібні вакансії



Рисунок 5.5 – Перегляд подібних вакансій

Приклад перегляду вакансії на працевлаштування в повному вигляді продемонстровано на рис. 5.6.

За кожною вакансією виводиться інформація про компанію, яка її виставила, місто, в якому вона актуальна, номер телефону, сайт компанії, коротке оголошення (воно використовується в роботі методу під час декодування саме цієї текстової інформації про вакансію), а також весь інший текст, співвіднесений з вакансією, який знаходиться в файлі вакансії і містить в тому числі відповідну розмітку.

Back-end розробник

КОМПАНІЯ:

- Authority Softline

МІСТО:

- Київ

ТЕЛЕФОН:

- +380958254937

САЙТ:

- <http://authoritysoft.com>

ОГОЛОШЕННЯ:

- Пропонуємо роботу в найбільш теплій компанії Києва, гнучкий графік, ваші професійні амбіції, наші найцікавіші проекти, софт, що є, - наш принцип.

ЗАРОБІТНА ПЛАТНЯ ВІД:

- 30 000 грн.

ОБОВ'ЯЗКОВІ ВИМОГИ

- 3 роки роботи в комерційній програмній розробці.
- Знання Java Core / EE.
- Знання баз даних: MySQL, PostgreSQL.
- Фреймворки: MVC, MVVM.

ПЛЮСИ

- Досвід роботи з платіжними системами.
- Розуміння архітектури мікросервісів.

ОБОВ'ЯЗКИ

- Аналіз коду.
- Модульне тестування.
- Аналіз технічних фінансів.

ПРОПОНУЄМО

- Гнучкий графік.
- Team building.
- Курси англійської мови.

ПОДАТИ ЗАЯВУ

Рисунок 5.6 – Перегляд вакансії на працевлаштування

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

6.1 Планування розробки програмного продукту

Результатом дипломної кваліфікаційної роботи магістра є програма, яка буде застосовуватися для пошуку вакансій для працевлаштування. В результаті очікується продаж 10000 копій програмного додатку за суму, що дорівнює 1 долар за копію, тобто 28,6 грн. за копію. Таке число проданих додатків буде досягнуто за рахунок конкурентної вартості порівняно з аналогами, що мають відповідну функціональність, зокрема програма «100 Best Real Remote Jobs!» коштує своїм користувачам 139,99 грн., таку ж суму встановлено на додаток «Job Solution». З розширенням кількості продажів планується розміщувати оголошення щодо пошуку вакансій, яких не буде на інших ресурсах, за рахунок чого привертати увагу до застосунку. Окрім того привертання уваги буде виконуватися за рахунок наявних засобів аналітики, відсутніх у аналогів.

6.2 Планування розробки програмного продукту

Для розрахунку витрат необхідно визначити етапи робіт та їх тривалість [18]. У розробці беруть участь програміст протягом 4 місяців та консультант протягом 0,25 місяця. Розробка програмного забезпечення починається 14 серпня та повинна бути закінчена до 7 грудня 2020 року.

Тривалість робіт визначають за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, людей [18].

У табл. 6.1 наведена трудомісткість проектування системи.

Таблиця 6.1 – Характеристика робіт з розробки програми

Найменування роботи	Трудомісткість		Виконавці	Тривалість, дні
	люд.-дні	% до підсумку		
Аналіз предметної області	2	2,35	Програміст	2
Узгодження результатів аналізу	6	7,06	Програміст Консультант	3
Визначення вимог до програмного продукту	5	5,88	Програміст	5
Узгодження вимог	4	4,71	Програміст Консультант	2
Документування вимог	3	3,53	Програміст	3
Проектування структури програми	20	23,53	Програміст	20
Розробка схеми функціонування програми	5	5,88	Програміст	5
Створення програмного коду	20	23,53	Програміст	20
Тестування та налагодження програмного продукту	5	5,88	Програміст	5
Складання програмної документації	15	17,65	Програміст	15
Підсумок	85	100		80

За даними табл. 6.1 складається зведений стрічковий графік розробки програмного продукту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення строків початку виконання всі види роботи, а в стовпцях – календарний період їх виконання [18].

Перша частина даного графіку наведена в табл. 6.2 і відповідає датам з 14 серпня по 16 жовтня 2020 року.

Таблиця 6.2 – Зведений графік планування розробки програмного продукту з 14.08 по 16.10

Найменування робіт	Календарний період, дні									
	14.08-14.08	17.08-21.08	24.08-28.08	31.08-4.09	7.09-11.09	14.09-18.09	21.09-25.09	28.09-2.10	5.10-9.10	12.10-16.10
Аналіз предметної області										
Узгодження результатів аналізу		 								
Визначення вимог до програмного продукту		   								
Узгодження вимог				 						
Документування вимог				  						
Проектування структури програми					   	   	   			
Розробка схеми функціонування програми									   	
Створення програмного коду										   

Друга частина графіку наведена в табл. 6.3 і відповідає датам з 19

жовтня по 7 грудня 2020 року.

Таблиця 6.3 – Зведений графік планування розробки програмного продукту з 19.10 по 7.12

Найменування робіт	Календарний період, дні							
	19.10-23.10	26.10-30.10	2.11-6.11	9.11-13.11	16.11-20.11	23.11-27.11	30.11-4.12	7.12-11.12
Створення програмного коду								
Тестування та налагодження програмного продукту								
Складання програмної документації								

6.3 Розрахунок основної заробітної плати

Розрахунок зарплати ведеться на підставі даних про трудомісткість, поданих у табл. 6.4.

Таблиця 6.4 – Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн.	Кількість місяців роботи	Сума заробітної платні, грн.
Програміст	1	5000	4	20000
Керівник	1	8000	0,25	2000
Підсумок	2			22000

6.4 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10 % від основної заробітної плати працівників і розраховують за формулою (6.2):

$$ЗП_{\text{доп}} = ЗП_{\text{осн}} \cdot 0,1 \quad [18]. \quad (6.2)$$

У даному випадку отримаємо:

$$ЗП_{\text{доп}} = 22000 \cdot 0,1 = 2200 \text{ грн}.$$

6.5 Єдиний соціальний внесок

Єдиний соціальний внесок становить 22% і береться від основної та додаткової заробітної плати за формулою (6.3):

$$ВТ_{\text{соц}} = (ЗП_{\text{осн}} + ЗП_{\text{доп}}) \cdot 0,22 \quad [18], \quad (6.3)$$

$$ВТ_{\text{соц}} = (22000 + 2200) \cdot 0,22 = 5324 \text{ грн}.$$

6.6 Визначення витрат на матеріали

Використовується 3 найменування матеріалів: диск DVD+RW – 10 грн., тонер для картриджа – 95 грн. і папір – 95 грн. (1 упаковка).

Витрати на матеріали вираховуються за наступною формулою (6.4):

$$ВТ_{\text{мат}} = \sum_{i=1}^n (Ц_i \cdot N_i \cdot (1 + K_{\text{м.з.}}) - Ц_{\text{іо}} \cdot N_{\text{іо}}), \quad (6.4)$$

де BT_{mat} – це витрати на матеріали, покупні напівфабрикати та комплектуючі вироби, грн.;

$K_{m.з.}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

C_i – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн.;

N_i – потреба в i -му матеріалі, напівфабрикаті і комплектуючому;

C_{io} – ціна зворотних відходів i -го найменування матеріалу, грн/;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих [18].

Обрахуємо:

$$C_{io} = 0, N_{io} = 0, K_{m.з.} = 0,05,$$

$$BT_{mat} = (1 + 0,05) \cdot (10 + 95 + 95) = 210 \text{ грн.}$$

Разом, витрати на матеріали становлять 210 грн.

6.7 Витрати на спеціальне обладнання

Середнє енергоспоживання ноутбука при подібній проєктній роботі складає 200 Ват, початкова ціна робочої машини – 15 000 грн.

Амортизаційні відрахування визначають за формулою (6.5):

$$A = \Phi_{\delta} \cdot \frac{H_a}{100}, \quad (6.5)$$

де Φ_{δ} – балансова вартість обчислювальної техніки, грн.;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, % [18].

Норма амортизації для персонального комп'ютеру (ПК) становить 25%.

Балансова вартість обчислювальної техніки становить:

$$\Phi_0 = 3000 \text{ грн.}$$

Тоді отримаємо амортизаційні відрахування:

$$A = 3000 \cdot 0,25 = 750 \text{ грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію та допоміжні матеріали за формулою (6.6):

$$C_e = N_n \cdot \Phi_{ef} \cdot K_{зч} \cdot K_{зн} \cdot \Pi_{ел.е}, \quad (6.6)$$

де N_n – номінальна потужність електронно-обчислювальної машини (ЕОМ), кВт;

Φ_{ef} – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{зч}$ – середній коефіцієнт завантаження по часу;

$K_{зн}$ – коефіцієнт завантаження по потужності;

$\Pi_{ел.е}$ – ціна однієї кВт·год електроенергії, грн./(кВт·год) [18].

Номінальна потужність робочої машини – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження по часу і по потужності рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,11 грн.

Отримуємо ціну експлуатації обладнання:

$$C_e = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 410,18 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою (6.7):

$$ЗП_{обсл} = \Phi ЗП_p \cdot (1 + K_{відр}) \cdot \frac{t_{обсл}}{\Phi_{эф.обсл}}, \quad (6.7)$$

де $\Phi ЗП_p$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн.;

$K_{відр}$ – коефіцієнт, що враховує відрахування на соціальне страхування та в інші фонди;

$t_{обсл}$ – час протягом року, необхідний на технічне обслуговування ЕОМ, год./рік;

$\Phi_{эф.обсл}$ – річний ефективний фонд часу обслуговуючого персоналу, год./рік [18].

Місячна заробітна плата обслуговуючого персоналу становить 3723 грн., а річний фонд заробітної плати відповідно дорівнює 44676 грн. Річний ефективний фонд робочого часу обслуговуючого ПК працівника дорівнює 1750 год./рік. На обслуговування одного ПК витрачається по 1 годині на місяць, що в рік становить 12 годин.

Отримуємо:

$$ЗП_{обсл} = 44676 \cdot (1 + 0,22) \cdot 12 / 1750 = 373,75 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і становить 90 грн.

Стаття «Інші витрати» приймається рівною 5% від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 1623,93 грн., 5% від суми становлять 81,20 грн.

Розраховані статті витрат на утримання та експлуатацію обладнання внесено до табл. 6.5.

Таблиця 6.5 – Кошторис витрат на утримання та експлуатацію обладнання

Найменування статей витрат	Сума, грн.
Амортизація обладнання	750
Експлуатація обладнання (крім витрат на поточний ремонт)	410,18
Заробітна плата основна та додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	373,75
Поточний ремонт обладнання	90
Інші витрати	81,20
Разом	1705,13

Витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми визначаються за формулою (6.8):

$$C_{mo} = B_{екс} \cdot t_{mo}, \quad (6.8)$$

де C_{mo} – витрати на оплату машинного часу, грн;

$B_{екс}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн/машино-год;

t_{mo} – машинний час цифровий ЕОМ для написання і налагодження даного програмного продукту, машино-год. [18].

Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин, в результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$B_{екс} = 1705,13 / 1800 = 0,95 \text{ грн} / \text{машино} - \text{год}.$$

ЕОМ експлуатується 70 днів в одну зміну, що становить в сумі 560 годин. Таким чином, витрати на оплату машинного часу складуть

$$C_{mo} = 0,95 \cdot 560 = 532 \text{ грн.}$$

6.8 Інші прямі витрати

До інших прямих витрат включаються витрати на використовуване при розробці системи комерційне програмне забезпечення: пайове програмне забезпечення, що використовується постійно при роботі ПК (Windows 10 Professional), – 5 289 грн. в сумі без податку на додану вартість.

Підрахунки виконуються за формулами (6.9)-(6.10):

$$S_{най.ПО} = \frac{Ц_{ПОWindows} \cdot T_{KTC}}{\Phi_{ef.KTC} \cdot T_{c.ПО}}, \quad (6.9)$$

$$S_{цел.ПО} = Ц_{ПОEclipse}, \quad (6.10)$$

де $S_{най.ПО}$ – витрати на пайове програмне забезпечення при розробці програмного забезпечення з розрахунків, грн.;

$S_{цел.ПО}$ – витрати на цільове програмне забезпечення, що купується виключно для розробки програмного забезпечення з розрахунків, грн.;

$Ц_{ПОWindows}$ – ціна за Windows (без податку на додану вартість), грн.;

$Ц_{ПОEclipse}$ – ціна за Eclipse (без податку на додану вартість), грн.;

T_{KTC} – машинний час комплексу технічних засобів, необхідний користувачеві для розробки системи, машино-год./рік;

$\Phi_{ef.KTC}$ – річний ефективний фонд часу роботи комплексу технічних засобів, машино-год./рік;

$T_{c.ПО}$ – термін служби пайового програмного забезпечення, років [18].

Обчислимо:

$$S_{най.ПО} = \frac{5289 \cdot 560}{1800 \cdot 5} = 329,09 \text{ грн},$$

$$S_{цел.ПО} = 0 \text{ грн},$$

$$S_{\Sigma} = 329,09 + 0 = 329,09 \text{ грн}.$$

6.9 Розрахунок накладних витрат

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведені в табл. 6.6.

Таблиця 6.6 – Калькуляція кошторисної вартості робіт з розробки програми та ціни програмного забезпечення

Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
Основна заробітна плата	22000	52,89
Додаткова заробітна плата	2200	5,29
Єдиний соціальний внесок	5324	12,80
Матеріали та комплектуючі	210	0,50
Витрати на спеціальне обладнання	532	1,28
Інші прямі витрати	329,09	0,79
Накладні витрати	11000	26,45
Разом	41595,09	100
Прибуток (30%)	12478,53	
Податок на додану вартість	10814,72	
Разом (з податком на додану вартість)	64888,34	

6.10 Розрахунок економічної ефективності програмного продукту

Очікується продаж 10000 додатків по одному долару за копію. При курсі долара 28,6 гривень за 1 долар, з урахуванням ризику 50 %, надходження від продажів складуть 143 тис. грн., що значно більше вартості програми. При цьому для потенційних користувачів встановлена вартість буде значно вигіднішою порівняно з вартістю існуючих аналогів. Вартість закупівлі 10000 екземплярів програми для замовника складає 286 тис. грн., а вартість закупівлі такої ж кількості екземплярів програм-аналогів (139,99 грн. за одиницю) складає 1399,9 тис. грн., що відповідно в 4,89 разів дорожче.

Отримані результати розрахунків наведені в табл. 6.7.

Таблиця 6.7 – Результати розрахунків

Показник	Сума, грн.
Надходження від продажів	143 000
Вартість розробки програми	64 888,34
Прибуток	78 111,66

Таким чином, розробка даного програмного забезпечення економічно доцільна.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

Сучасний розвиток науки та техніки привносить принципові нововведення у всі сфери матеріального виробництва, докорінно змінюючи знаряддя та предмети праці, технологію, методи обробки інформації, разом з тим, захопившись вдосконаленням засобів праці їх творці залишили поза увагою проблеми людини в рамках комп'ютерної революції [19].

Дослідження та програмна реалізація методів пошуку вакансій для працевлаштування передбачає роботу в обчислювальному центрі невеликої команди програмістів.

Небезпечні та шкідливі виробничі фактори за природою виникнення поділяються на такі групи: фізичні (підвищена і знижена температура повітря, надмірна запиленість і загазованість повітря та ін.); хімічні (вміст у робочій зоні шкідливих речовин та ін.); психофізіологічні (нервово-емоційні перевантаження, розумове напруження та ін.); біологічні (підвищений вміст у повітрі мікроорганізмів та ін.) [19].

7.2 Заходи забезпечення безпеки

При роботі в обчислювальному центрі головним небезпечним фактором є електрика: питання правильного електроживлення при об'єднанні комп'ютерів в локальні мережі мають важливе значення як для забезпечення електробезпеки користувачів, так і для безаварійної і безперебійної роботи обладнання [19].

Заходи щодо забезпечення електробезпеки розроблено відповідно до ДСТУ Б В.2.5-82:2016 «Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом» [20].

Для запобігання ураження людини струмом служить захисне заземлення: сучасні ПК включаються в розетку за допомогою триполюсної

вилки, один з полюсів якої використовується для заземлення, з ним електрично з'єднані металеві неструмоведучі частини ПК, які можуть опинитися під напругою, в обчислювальному центрі передбачається використання тільки триполюсних розеток, у яких відповідний полюс з'єднаний з шиною заземлення [20].

Кожен блок живлення комп'ютера або периферійного пристрою має мережевий фільтр, конденсатори цього фільтра призначені для шунтування високочастотних перешкод живлячої мережі на землю через дріт захисного заземлення [21].

Згідно з «Правилами улаштування електроустановок» опір захисного заземлення в обчислювальному центрі становить 3 Ом (за стандартом – не більше 4 Ом) [19].

Згідно з ГОСТ 12.2.007.0–75 «Вироби електротехнічні. Загальні вимоги безпеки» в приміщенні встановлено автоматичні вимикачі, пристрій заземлення контуру [22].

7.3 Заходи з забезпечення виробничої санітарії та гігієни праці

Заходи щодо забезпечення виробничої санітарії та гігієни праці для обчислювального центру обладнаного ПК з візуальними дисплейними терміналами розроблені відповідно до вимог Державних санітарних норм та правил «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу», Міністерства юстиції України 06.05.2014 р. за № 472/25249, Державних санітарних правил і норм 3.3.2.007-98 «Державні стандартні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [22].

Роботи, що виконуються сидячи і не потребують фізичного напруження, при яких витрати енергії складають до 139 Вт, для забезпечення оптимального рівня параметрів повітряного виробничого середовища

використовуємо Державні санітарні норми 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» [23] та ГОСТ 12.1.005-88 (1991) норми мікроклімату в приміщенні наведені в табл. 7.1 [21].

Таблиця 7.1 – Норми мікроклімату в приміщенні

Період року	Категорія робіт	Температура повітря, °С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодний	Легка	22–24	40–60	0,1
Теплий	Легка	23–25	40–60	0,1-0,2

В обчислювальному центрі передбачено: устрій системи водяного опалення приміщення для забезпечення необхідної температури повітря в холодний період року відповідно до Державних будівельних норм В.2.5-67:2013 «Опалення, вентиляція та кондиціювання» [24].

Вимоги до освітлення регламентовані Державними будівельними нормами В.2.5-28-2018 «Природне та штучне освітлення» [25], згідно з якими освітлення у приміщеннях з ЕОМ суміщене, при якому недостатнє за нормами природне освітлення доповнене штучним, природне освітлення бічне, одностороннє [25].

В обчислювальному центрі коефіцієнт природної освітленості має становити в середньому 1,1% (1,2% – у сонячний день та 1% – у хмарну погоду), це відповідає вимогам Державних будівельних норм В.2.5-28-2018, для забезпечення відносної постійності природного освітлення вікна обладнанні сонцезахисними металевими жалюзі з системою регулювання, в таких жалюзі коефіцієнт відбиття становить 0,7, робочі місця розташовано так, щоб у поле зору не потрапляли вікна або світлі поверхні світильників [24].

Найбільш прийнятними для приміщень, де працюють ПК, є люмінесцентні лампи ЛБ (білого світла) і ЛТБ (тепло-білого світла) потужністю 20, 40 або 80 Вт.

Розрахуємо кількість ламп необхідних для освітлення приміщення розміром 9·4·3,5м освітленістю $E_p=400\text{лк}$. Коефіцієнт відбиття стелі 70% і стін 50%. Для освітлення використовуються люмінесцентні лампи типу ЛБ в світильниках ЛПО.

Знаходимо індекс приміщення за формулою (7.1):

$$i = \frac{AB}{h(A+B)}, \quad (7.1)$$

де A – довжина приміщення, м;

B – ширина приміщення, м;

h – висота підвісу світильника над робочою поверхнею, м.

Розрахуємо:

$$i = \frac{9 \cdot 4}{3,5 \cdot (9+4)} = 1,94.$$

Приймаємо коефіцієнт запасу $k = 1,6$ і коефіцієнт нерівномірності освітлення 1.1, при індексі $i=0,8$ отримуємо $\eta=32\%$, вітильники розміщуємо в два ряди ($N_p = 2$), визначаємо необхідний світловий потік ламп в кожному ряду за формулою (7.2):

$$\Phi_{\Sigma} = \frac{E_H \cdot A \cdot B \cdot k_z \cdot z}{\eta}, \quad (7.2)$$

де E_H – рівень нормованого загального освітлення, лк;

A – довжина приміщення, м;

B – ширина приміщення, м;

k_z – коефіцієнт запасу (для люмінесцентних ламп $k_z = 1,4 \dots 1,7$);

z – коефіцієнт нерівномірності (мінімальної) освітленості (відношення середньої освітленості до мінімальної освітленості), як правило дорівнює (для люмінесцентних ламп $z = 1,1$);

η – коефіцієнт використання світлового потоку.

Розрахуємо:

$$\Phi_{\Sigma} = \frac{400 \cdot 36 \cdot 1,1 \cdot 1,6}{2 \cdot 0,32} = 39600 \text{ лм.}$$

Для виключення засвічення екранів дисплеїв прямими світловими потоками світильники загального освітлення мають в своєму розпорядженні збоку від робочого місця, паралельно лінії зору програміста і стіні з вікнами.

Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється стомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою, погіршується мовна комутація.

Згідно із технічним паспортом обладнання (ПК, монітори, принтери, клавіатури (при натисканні), маніпулятори типу «миша» (при натисканні)), що використовується для роботи, рівень шуму складає: 43 дБА, згідно із Державними санітарними нормами 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвук» нормоване значення рівня шуму більше, ніж фактичне, тож додаткових робіт для усунення надмірного шуму не проводилося, роботи в даному приміщенні з таким обладнанням відповідають нормативним вимогам [26].

На програміста, в період роботи за дисплеєм, здійснюється інтенсивне електромагнітне випромінювання, яке може стати причиною професійних захворювань, комп'ютери потрапляють за визначенням в 5-й діапазон (низькі частоти) [26].

Основним джерелом електромагнітного випромінювання, у тому числі рентгенівського, в приміщенні є електронно-променеві трубки старих моніторів, через це, для зниження потужності експозиційної дози рентгенівського випромінювання для роботи використовуються монітори з рідкокристалічними дисплеями [25].

Обладнання і організація робочого місця в обчислювальному центрі мають забезпечувати відповідність конструкції всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності [20].

Робочі стільці мають підйомно-повторним механізми та регулюються по висоті та куту нахилу сидіння і спинки, а також по відстані спинки від переднього краю сидіння, регулювання кожного параметра незалежне, легко здійснюється та має надійну фіксацію, екран монітору повинний знаходитися на оптимальній відстані 700 мм [20].

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ, велике значення має раціональна конструкція і розташовує елементів робочого місця, що важливе для підтримки оптимальної робочої пози людини-оператора [22].

В приміщенні обчислювального центру наявні 2 медичні аптечки першої допомоги.

Також приміщення обладнано шафами для зберігання документів, магнітних дисків, полицями, стелажми та тумбами.

Колір приміщень і меблів сприяє створенню комфортних умов для зорового сприйняття.

Це основні рішення, які було прийнято відносно заходів з забезпечення виробничої санітарії та гігієни праці.

7.4 Заходи з пожежної безпеки

Пожежі в обчислювальному центрі становлять особливу небезпеку, тому що пов'язані з великими матеріальними втратами. Як відомо, пожежа може виникнути при взаємодії горючих речовин, окислення і джерел запалювання, у приміщеннях обчислювальних центрів присутні всі три основні чинники, необхідні для виникнення пожежі, при цьому відзначено наступне:

- для відводу теплоти від ПК у обчислювальному центрі постійно діє потужна система кондиціонування, тому кисень, як окислювач процесів горіння, є в будь-якій точці обчислювального центру;

- обчислювальний центр, згідно ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» [27] відноситься до категорії «П-Па», а клас можливої пожежі, згідно Державних будівельних норм В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [28], визначається, як «А»;

- до засобів гасіння пожежі, призначених для локалізації невеликих загорянь, відносяться пожежні стовбури, внутрішні пожежні водопроводи, вогнегасники, сухий пісок, азбестові ковдри і т. п.;

- оскільки приміщення (дослідницької лабораторії, конструкторського бюро, тощо), що обладнане ПК з візуальними дисплейними терміналами, має площу 36 м², тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» [29] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 2 штук (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг. і більше, на 20 м² площі приміщення), відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м;

– усі працівники обчислювального центру під час прийняття на роботу і в процесі праці повинні проходити протипожежний інструктаж, перевірку знань з питань пожежної безпеки, евакуаційні виходи і проходи утримуються постійно вільними та нічим не захащується;

– відповідальний за протипожежний стан приміщень обчислювального центру повинен періодично (не рідше одного разу на місяць) перевіряти працездатність пристрою, який забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогнедимозатримувальних пристроїв, під час ремонту слід не допускати руйнування або порушення цілісності негорючих діафрагм у фальш-підлоговому просторі;

– відповідальний за пожежну безпеку приміщень обчислювального центру зобов'язаний негайно сповістити технічну службу у разі виявлення незадовільного технічного стану димових та теплових сповіщувачів автоматичної пожежної сигналізації, а також про несправності автоматичної установки об'ємного (газового) пожежогасіння [22], [27]-[29].

Усі працівники обчислювального центру повинні вміти привести в дію (в разі потреби) первинні засоби пожежогасіння. Регулярно, але не рідше одного разу на квартал, слід очищати від пилу ЕОМ, контрольновимірну апаратуру, кабельні траншеї та фальшпідлоговий простір за допомогою пиловсмоктувача [27].

7.5 Заходи забезпечення безпеки у надзвичайних ситуаціях

Головним і невід'ємним елементом всієї системи захисту населення і територій від надзвичайних ситуацій техногенного та природного характеру є інформація та оповіщення, для своєчасного попередження населення введені сигнали попередження населення у мирний і воєнний час: сигнал «Увага всім!» повідомляє населення про надзвичайну обстановку в мирний час і на випадок загрози нападу противника у воєнний час, сигнал подається

органами цивільного захисту за допомогою сирени і виробничих гудків, тривалі гудки означають попереджувальний сигнал, почувши їх, необхідно включити радіо, телевізор і прослухати текст інформації про дії населення після одержання сигналу, якщо немає радіо, телевізора або вони не працюють, слід з'ясувати значення і зміст інформації у сусідів або інших людей, які знають про неї, після одержання інформації необхідно виконати всі вказівки тексту інформації сигналу [30].

Успіх захисту населення залежатиме від дисциплінованості, своєчасної і правильної поведінки, суворого дотримання рекомендацій і вимог органів цивільного захисту [30].

ВИСНОВКИ

У дипломній кваліфікаційній роботі магістра виконано аналіз проблеми пошуку вакансій для працевлаштування. Для цього було виконано наступні завдання:

- розробка методу пошуку вакансій для працевлаштування;
- проєктування програми;
- розробка програми;
- розробка організаційно-економічної частини;
- розробка заходів з охорони праці та цивільної безпеки.

Розглянуто задачу пошуку вакансій для працевлаштування. Проаналізовано метод k-найближчих сусідів, принципи його застосування. Проаналізовано принципи роботи моделі word2vec. Визначено власний метод пошуку вакансій для працевлаштування. В якості міри схожості використано косинусну подібність.

Розглянуто основні сценарії використання системи користувачами на основі побудови діаграм варіантів використання для менеджера та кандидата на працевлаштування. Вирішено проєктувати і реалізовувати програмне забезпечення в якості вебзастосунку в основі. Для реалізації обрано мову програмування Python.

Для забезпечення роботи з базою даних використано систему MySQL. Визначено таблиці бази даних, встановлено зв'язки між ними, побудовано схему бази даних. Визначено структуру програмного проєкту з побудовою відповідної діаграми компонентів.

Програмно реалізовано вебзастосунок пошуку вакансій для працевлаштування, який дозволяє вносити дані наявних вакансій на працевлаштування, створювати резюме кандидатів, переглядати вакансії на працевлаштування, виконувати пошук вакансій, підбирати вакансії за резюме кандидата та за обраними вакансіями кандидатом, вносити зміни в вакансії,

вносити запити на працевлаштування, переглядати найбільш популярні за запитами на працевлаштування компанії.

Наукова новизна роботи полягає в розробленому методі пошуку вакансій для працевлаштування, що виконує пошук подібних вакансій на основі їх класифікації за допомогою методу k-найближчих сусідів, використовуючи в якості додаткових ознак закодовані за допомогою моделі word2vec значення, що дозволяє обґрунтовано приймати рішення про вибір вакансій або резюме для детального аналізу.

Усі завдання дипломної кваліфікаційної роботи магістра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Doyle, A. 10 Steps for Finding Your New Job [Electronic resource] / A. Doyle. – Access mode : <https://www.thebalancecareers.com/steps-to-find-a-new-job-2060725>.
2. Burnett, B. Designing Your Life [Text] : How to Build a Well-Lived, Joyful Life / Bill Burnett, Dave Evans. – New York : Knopf, 2016. – 272 p.
3. Bolles, R. N. What Color Is Your Parachute? 2020 [Text] : A Practical Manual for Job-Hunters and Career-Changers / R. N. Bolles. – Ten Speed Press, 2019. – 320 p.
4. Пармон, А. С. Сайты вакансий [Электрон. ресурс] : куда податься в поисках работы / А.С. Пармон. – Режим доступа : <https://www.kp.ru/guide/saity-vakansii.html>.
5. How to Use The Internet in Your Job Search [Electronic resource]. – Access mode : <https://www.myperspectives.com/career-center/jobs/search/conducting-an-online-job-search+&cd=1&hl=uk&ct=clnk&gl=ua>.
6. Polner, E. The 10 Best Job Search Websites of 2020 [Electronic resource] / E. Polner. – Access mode : <https://www.thebalancecareers.com/top-best-job-websites-2064080>.
7. Word Embedding Tutorial [Electronic resource] : word2vec using Gensim. – Access mode : <https://www.guru99.com/word-embedding-word2vec.html>.
8. Menshawy, A. Deep Learning By Example [Text] : A hands-on guide to implementing advanced machine learning algorithms and neural networks / A. Menshawy. – Birmingham : Packt Publishing, 2018. – 450 p.
9. Ali, Z. A simple Word2vec tutorial [Electronic resource] / Z. Ali. – Access mode : <https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1>.
10. Karani, D. Introduction to Word Embedding and Word2Vec [Electronic resource] / D. Karani. – Access mode :

<https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa>.

11. K-Nearest Neighbor (KNN) Algorithm for Machine Learning [Electronic resource]. – Access mode : <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>.

12. Schott, M. K-Nearest Neighbors (KNN) Algorithm for Machine Learning [Electronic resource] / M. Schott. – Access mode : <https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26>.

13. Chakure, A. K-Nearest Neighbors (KNN) Algorithm [Electronic resource] / A. Chakure. – Access mode : <https://quintagroup.com/cms/python/flask>.

14. Лутц, М. Изучаем Python [Текст] / М. Лутц. – СПб. : Символ-Плюс, 2010. – 1280 с.

15. Маккинли, У. Python и анализ данных [Текст] / У. Маккинли. – М. : ДМК Пресс, 2015. – 482 с.

16. Ленгсторф, Дж. PHP и jQuery для профессионалов / Дж. Ленгсторф. – М. : Вильямс, 2010. – 352 с.

17. Flask Python Framework [Electronic resource]. – Access mode : <https://quintagroup.com/cms/python/flask>.

18. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 8.05010301 «Програмне забезпечення автоматизованих систем» та 8.05010201 «Інформаційні технології проєктування» для усіх форм навчання [Текст] / В.В. Остапенко, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с.

19. Правила безпечної експлуатації електроустановок споживачів [Текст] : НПАОП 40.1-1.21-98. : На заміну ДНАОП 0.00.1.21-84 : чинний від 1998-01-09. – К. : Мінпраці України, 1998. – 89 с.

20. Основи охорони праці. [Текст] : Підручник. / О. І. Запорожець, О. С. Протоєрейський, Г. М. Франчук, І. М. Боровик. – К. : Центр учбової

літератури, 2009. – 264 с.

21. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс] : ДСанПіН 3.3.2.007-98. : Чинний від 1998-12-10. – К. : МОЗ України, 1998. – Режим доступу : <http://mozdocs.kiev.ua/view.php?id=2445>.

22. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [Текст]. – Харків : Форт, 2000. – 24 с.

23. Законодавство про охорону праці. Том 1 [Текст] / Харків : Форт, 2013. – 368 с.

24. ДБН В.2.5-28:2018. Природне та штучне освітлення. Посібник [Текст] : проблеми природного і штучного освітлення / К. : Етін, 2019. – 180 с.

25. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку [Текст]. – К. : МОЗ України, 1999. – 24 с.

26. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою [Текст] : ДСТУ Б В.1.1-36:2016. : На заміну НАПБ Б.03.002-2007 : чинний від 2017-01-01. – К. : Мінрегіонбуд України, 2016. – 66 с. – (Державний Стандарт України)

27. Пожежна безпека об'єктів будівництва. Загальні вимоги [Текст] : ДБН В.1.1-7:2016. : На заміну ДБН В.1.1.7-2002 : чинний від 2017-06-01. – К. : Мінрегіон України, 2017. – 47 с. – (Державні будівельні норми).

28. Жидецький, В. Ц. Основи охорони праці [Текст] : Підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.

29. Шоботов, В. М. Цивільна оборона [Текст] : Навчальний посібник / В. М. Шоботов. – Вид. 2-ге, перероб. – К. : Центр навчальної літератури, 2006. – 438 с.

30. Демиденко, Г. П. Повышение устойчивости работы объектов народного хозяйства в военное время [Текст] : учебное пособие / Г. П. Демиденко. – К. : Вища шк. : Головное изд-во, 1984. – 232 с.

ДОДАТОК А
Технічне завдання

Вступ

Для розв'язання проблеми пошуку вакансій для працевлаштування необхідно розробити метод, що буде забезпечувати розв'язання цієї проблеми, та програмно реалізувати його для забезпечення практичного впровадження.

A.1 Підстави для розробки

Розробка виконується на основі завдання на дипломну кваліфікаційну роботу магістра, яке було визначене за темою “Дослідження та програмна реалізація методів пошуку вакансій для працевлаштування” і затверджене наказом № 301 від 3 листопада 2020 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення призначене для виконання пошуку вакансій для працевлаштування здобувачами та пошуку резюме кандидатів представниками компаній (рекрутерами). Це забезпечується як наданням засобів для внесення таких даних, їх перегляду, так і наданням засобів пошуку, що включають як традиційний пошук, так і підбір подібних вакансій.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функціональні характеристики, виконання яких має забезпечуватися програмою:

- вносити дані наявних вакансій на працевлаштування;

- створювати резюме кандидатів;
- переглядати вакансії на працевлаштування;
- виконувати пошук вакансій;
- підбирати вакансії;
- вносити зміни в вакансії;
- вносити запити на працевлаштування;
- вносити відповіді на подані запити;
- реєструвати облікові записи кандидатів;
- створювати облікові записи менеджерів компанії;
- авторизувати облікові записи;
- вносити дані про попередню роботу та освіту кандидата;
- переглядати найбільш популярні за запитами на працевлаштування компанії.

A.3.2 Вимоги до надійності

Основним чином надійність роботи програми має забезпечуватися шляхом надійності надання доступу до серверу.

Окрім того програма повинна контролювати дані, які потрапляють їй на вхід або отримуються в результаті її роботи.

Вхідні дані охоплюють зокрема дані, отримані з бази даних, від користувача, між підсистемами програмної системи.

A.3.3 Умови експлуатації

Для експлуатації програми має бути виділено окремий сервер, через який буде відбуватися доступ до вебзастосунку. Окрім того на сервері має розташовуватися база даних та система MySQL, але для збереження їх може виділятися також окремий сервер бази даних.

Відповідно необхідно перевіряти, слідкувати і керувати роботою

сервера у відповідності з заданими вимогами до серверів.

A.3.4 Вимоги до складу та параметрів технічних засобів

Технічні засоби, на яких експлуатується програма, повинні включати наступний мінімальний набір:

- двоядерний процесор, частота не менше 2,4 ГГц;
- оперативна пам'ять – 4 Гб;
- не менше 2 Гб вільного простору на жорсткому диску.

A.3.5 Вимоги до транспортування та збереження

Передбачається, що програма буде зберігатися на сервері і таким чином буде виконуватися її розповсюдження.

A.3.6 Вимоги до програмної документації

Необхідно розробити програмні документи, що включають дане технічне завдання, текст програми, що оформлюються додатками пояснювальної записки до дипломної кваліфікаційної роботи магістра, та зокрема опис експлуатації та тестування програми.

A.3.7 Вимоги до маркування та упакування

Упаковка повинна бути цілісною у тому випадку, якщо програма зберігається у вигляді фізичної копії на носії інформації, включаючи зокрема флешки, компакт-диски, карти пам'яті.

А.4 Порядок контролю та приймання

Дипломна кваліфікаційна робота магістра повинна виконуватися у повній відповідності з календарним планом, виконання етапів має контролюватися керівником роботи та консультантами відповідних розділів. Останім етапом є захист роботи перед екзаменаційною комісією у дистанційній формі.

ДОДАТОК Б
Текст програми

```

import mysql.connector
import JobDataBase from DataBaseHandler

def CreateVacancyTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS Vacancy (
        Id INT NOT NULL AUTO_INCREMENT,
        Position VARCHAR(255) NOT NULL,
        Company_id INT,
        Salary FLOAT,
        RequirementText TEXT,
        FullInfo TEXT,
        Category_id INT,
        File VARCHAR(255),
        Created DATETIME,
        City VARCHAR(255),
        PRIMARY KEY (Id),
        FOREIGN KEY (Category_id) REFERENCES Category (id),
        FOREIGN KEY (Company_id) REFERENCES Company (id)

    ); """)
    JobDataBase.commit()

def addVacancyTb (position, cid, salary, requirement,
fullInfo, caid, file, created, city):
    dbcurs = JobDataBase.cursor()
    comm = """INSERT INTO
Vacancy (Position,
Category_id,
Salary,
RequirementText,
FullInfo,
Category_id,
File,
Created,
City)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s) """
    data = (position, cid, salary, requirement, fullInfo,
caid, file, created, city)

```

```

        dbcurs.execute(comm, data)
        JobDataBase.commit()

    def changeVacancy (num, position, cid, salary, requirement,
fullInfo, caid, file, created, city):
        dbcurs = JobDataBase.cursor()
        comm = """UPDATE Vacancy
SET Position = %s,
Category_id = %s,
Salary = %s,
RequirementText = %s,
FullInfo = %s,
Category_id = %s,
File = %s,
Created = %s,
City= %s
WHERE Id = %s"""
        data = (position, cid, salary, requirement, fullInfo,
caid, file, created, city, num)
        dbcurs.execute(comm, data)
        JobDataBase.commit()

def getVacancies ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM Vacancy)
    result = dbcurs.fetchall()
    JobDataBase.commit()
    return result

def getVacancybyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""SELECT *
FROM Vacancy
WHERE Vacancy.id = """+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data

```

```

def CreateCandidateTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS Candidate (
        Id INT NOT NULL AUTO_INCREMENT,
        User_id INT,
        Name VARCHAR (255),
        Phone VARCHAR (13),
        Email VARCHAR (100),
        Site VARCHAR (255),
        Photo VARCHAR (255),
        EducationShort VARCHAR (255),
        FullInfo TEXT,
        PRIMARY KEY (Id),
        FOREIGN KEY (User_id) REFERENCES User(id)

    ); """)
    JobDataBase.commit()


def getCandidates ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM Candidate)
    result = dbcurs.fetchall()
    JobDataBase.commit()
    return result


def getCandidatebyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT      *      FROM      Candidate      WHERE
Candidate.id = "+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data


def addCandidateTb (user_id, name, phone, email, site, photo,
educationshort, fullinfo):
    dbcurs = JobDataBase.cursor()
    comm = """INSERT INTO
Candidate (User_id,

```

```

Name,
Phone,
Email,
Site,
Photo,
EducationShort,
FullInfo)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"""
    data = (user_id, name, phone, email, site, photo,
educationshort, fullinfo)
    dbcurs.execute(comm, data)
    JobDataBase.commit()

def changeCandidate (num, user_id, name, phone, email, site,
photo, educationshort, fullinfo):
    dbcurs = JobDataBase.cursor()
    comm = """UPDATE Candidate
SET User_id = %s,
Name = %s,
Phone = %s,
Email = %s,
Site = %s,
Photo = %s,
EducationShort = %s,
FullInfo = %s
WHERE Id = %s"""
    data = (user_id, name, phone, email, site, photo,
educationshort, fullinfo, num)
    dbcurs.execute(comm, data)
    JobDataBase.commit()

def CreateCompanyTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS Company (
        Id INT NOT NULL AUTO_INCREMENT,
        Name VARCHAR (255),
        Phone VARCHAR (13),
        Email VARCHAR (100),
        Site VARCHAR (255),

```

```

        Photo VARCHAR (255),
        FullInfo TEXT,
        PRIMARY KEY (Id)
    ); """
    JobDataBase.commit()

```

```

def addCompanyTb (name, phone, email, site, photo, fullInfo):
    dbcurs = JobDataBase.cursor()
    comm = """INSERT INTO Company (Name,
Phone,
Email,
Site,
Photo,
FullInfo)
VALUES (%s, %s, %s, %s, %s, %s) """
    data = (name, phone, email, site, photo, fullInfo)
    dbcurs.execute(comm, data)
    JobDataBase.commit()

```

```

def changeCompany (num, name, phone, email, site, photo,
fullInfo):
    dbcurs = JobDataBase.cursor()
    comm = """UPDATE
Company SET Name = %s,
Phone = %s,
Email = %s,
Site = %s,
Photo = %s,
FullInfo = %s
WHERE Id = %s"""
    data = (name, phone, email, site, photo, fullInfo, num)
    dbcurs.execute(comm, data)
    JobDataBase.commit()

```

```

def getCompanies ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM Company)
    result = dbcurs.fetchall()

```

```

        JobDataBase.commit()
        return result

def getCompanybyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM Company WHERE Company.id =
"+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data

def CreateRequestTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS Request (
        Id INT NOT NULL AUTO_INCREMENT,
        Candidate_id INT,
        Vacancy_id INT,
        RText TEXT,
        RFile VARCHAR(255),
        Answer TEXT,
        GetDate DATETIME,
        Status TINYINT,
        PRIMARY KEY (Id),
        FOREIGN KEY (Candidate_id) REFERENCES Candidate (id),
        FOREIGN KEY (Vacancy_id) REFERENCES Vacancy (id)

    ); """)
    JobDataBase.commit()

def addRequestTb (candidate_id, vacancy_id, rText, rFile,
answer, getDate, status):
    dbcurs = JobDataBase.cursor()
    comm = """INSERT INTO Request (Candidate_id,
    Vacancy_id,
    RText,
    RFile,
    Answer,
    GetDate,
```

```

        Status)
        VALUES (%s, %s, %s, %s, %s, %s, %s) """
        data = (candidate_id, vacancy_id, rText, rFile, answer,
getDate, status)
        dbcurs.execute(comm, data)
        JobDataBase.commit()

    def changeRequest (num, candidate_id, vacancy_id, rText,
rFile, answer, getDate, status):
        dbcurs = JobDataBase.cursor()
        comm = """UPDATE Request
SET Candidate_id = %s,
Vacancy_id = %s,
RText = %s,
RFile = %s,
Answer = %s,
GetDate = %s,
Status = %s
WHERE Id = %s"""
        data = (candidate_id, vacancy_id, rText, rFile, answer,
getDate, status, num)
        dbcurs.execute(comm, data)
        JobDataBase.commit()

def getRequests ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM Request)
    result = dbcurs.fetchall()
    JobDataBase.commit()
    return result

def getRequestbyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM Request WHERE Request.id =
"+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data

```

```

def CreateJobExperienceTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS
JobExperience (
    Id INT NOT NULL AUTO_INCREMENT,
    Candidate_id INT,
    StartDate DATETIME,
    EndDate DATETIME,
    Company_id INT,
    Position VARCHAR(255),
    Address VARCHAR(255),
    FullInfo TEXT,
    PRIMARY KEY (Id),
    FOREIGN KEY (Candidate_id) REFERENCES Candidate (id),
    FOREIGN KEY (Company_id) REFERENCES Company (id)

); """)
    JobDataBase.commit()

def addJobExperienceTb (candidate_id, startdate, enddate,
company_id, position, address, fullinfo):
    dbcurs = JobDataBase.cursor()
    comm = """INSERT INTO JobExperience (Candidate_id,
StartDate,
EndDate,
Company_id,
Position,
Address,
FullInfo)
VALUES (%s, %s, %s, %s, %s, %s, %s) """
    data = (candidate_id, startdate, enddate, company_id,
position, address, fullinfo))
    dbcurs.execute(comm, data)
    JobDataBase.commit()

def changeJobExperience (num, candidate_id, startdate,
enddate, company_id, position, address, fullinfo):
    dbcurs = JobDataBase.cursor()
    comm = """UPDATE JobExperience

```

```

SET Candidate_id = %s,
StartDate = %s,
EndDate = %s,
Company_id = %s,
Position = %s,
Address = %s,
FullInfo = %s
WHERE Id = %s"""

        data = (candidate_id, startdate, enddate, company_id,
position, address, fullinfo, num)
        dbcurs.execute(comm, data)
        JobDataBase.commit()

def getJobExperiences ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM JobExperience")
    result = dbcurs.fetchall()
    JobDataBase.commit()
    return result

def getJobExperiencebyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM JobExperience WHERE
JobExperience.id = "+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data

def CreateEducationTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS Education (
        Id INT NOT NULL AUTO_INCREMENT,
        Candidate_id INT,
        StartDate DATETIME,
        EndDate DATETIME,
        Institution VARCHAR(255),
        Speciality VARCHAR(255),
        Qualify VARCHAR(255),

```

```

        PRIMARY KEY (Id),
        FOREIGN KEY (Candidate_id) REFERENCES Candidate (id)

    ); """
    JobDataBase.commit()

    def addEducationTb (candidate_id, startdate, enddate,
institution, speciality, qualify):
        dbcurs = JobDataBase.cursor()
        comm = """INSERT INTO Education (Candidate_id,
StartDate,
EndDate,
Institution,
Speciality,
Qualify)
VALUES (%s, %s, %s, %s, %s, %s) """
        data = (candidate_id, startdate, enddate, institution,
speciality, qualify))
        dbcurs.execute(comm, data)
        JobDataBase.commit()

    def changeEducation (num, candidate_id, startdate, enddate,
institution, speciality, qualify):
        dbcurs = JobDataBase.cursor()
        comm = """UPDATE Education
SET Candidate_id = %s,
StartDate = %s,
EndDate = %s,
Institution = %s,
Speciality = %s,
Qualify = %s
WHERE Id = %s"""
        data = (candidate_id, startdate, enddate, institution,
speciality, qualify, num)
        dbcurs.execute(comm, data)
        JobDataBase.commit()

    def getEducations ():

```

```

        dbcurs = JobDataBase.cursor()
        dbcurs.execute("SELECT * FROM Education")
        result = dbcurs.fetchall()
        JobDataBase.commit()
        return result

def getEducationbyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM Education WHERE
Education.id = "+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data

def CreateCVTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS CV (
        Id INT NOT NULL AUTO_INCREMENT,
        Candidate_id INT,
        SearchPosition VARCHAR(255),
        SearchSalary FLOAT,
        FullInfo TEXT,
        File VARCHAR(255),
        PRIMARY KEY (Id),
        FOREIGN KEY (Candidate_id) REFERENCES Candidate (id)

    ); """)
    JobDataBase.commit()

def addCVTb (candidate_id, searchposition, searchsalary,
fullinfo, file):
    dbcurs = JobDataBase.cursor()
    comm = """INSERT INTO CV (Candidate_id,
SearchPosition,
SearchSalary,
FullInfo,
File)
VALUES (%s, %s, %s, %s, %s) """

```

```

        data = (candidate_id, searchposition, searchsalary,
fullinfo, file))
        dbcurs.execute(comm, data)
        JobDataBase.commit()

    def changeCV (num, candidate_id, searchposition,
searchsalary, fullinfo, file):
        dbcurs = JobDataBase.cursor()
        comm = """UPDATE CV
SET Candidate_id = %s,
SearchPosition = %s,
SearchSalary = %s,
FullInfo = %s,
File = %s
WHERE Id = %s"""
        data = (candidate_id, searchposition, searchsalary,
fullinfo, file, num)
        dbcurs.execute(comm, data)
        JobDataBase.commit()

def getCVs ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM CV)
    result = dbcurs.fetchall()
    JobDataBase.commit()
    return result

def getCVbyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM CV WHERE CV.id =
"+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data

def CreateCompanyUserTb():
    dbcurs = JobDataBase.cursor()

```

```

        dbcurs.execute("""CREATE TABLE IF NOT EXISTS CompanyUser
(
        Id INT NOT NULL AUTO_INCREMENT,
        User_id INT,
        Company_id INT,
        Rights INT,
        Phone VARCHAR(13),
        Email VARCHAR(100),
        PRIMARY KEY (Id),
        FOREIGN KEY (Company_id) REFERENCES Company (id),
        FOREIGN KEY (User_id) REFERENCES User (id)

); """)
        JobDataBase.commit()

def addCompanyUserTb (user_id, company_id, rights, phone,
email):
        dbcurs = JobDataBase.cursor()
        comm = """INSERT
INTO CompanyUser (User_id,
Company_id,
Rights,
Phone,
Email)
VALUES (%s, %s, %s, %s, %s) """
        data = (user_id, company_id, rights, phone, email)
        dbcurs.execute(comm, data)
        JobDataBase.commit()

def changeCompanyUser (num, user_id, company_id, rights,
phone, email):
        dbcurs = JobDataBase.cursor()
        comm = """UPDATE CompanyUser
SET User_id = %s,
Company_id = %s,
Rights = %s,
Phone = %s,
Email = %s
WHERE Id = %s"""

```

```

data = (user_id, company_id, rights, phone, email, num)
dbcurs.execute(comm, data)
JobDataBase.commit()

def getCompanyUsers ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM CompanyUser)
    result = dbcurs.fetchall()
    JobDataBase.commit()
    return result

def getCompanyUserbyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT      *      FROM      CompanyUser      WHERE
CompanyUser.id = "+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data

def CreateCategoryTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS Category (
        Id INT NOT NULL AUTO_INCREMENT,
        Name VARCHAR(255),
        PRIMARY KEY (Id)
    ); """)
    JobDataBase.commit()

def addCategoryTb (name):
    dbcurs = JobDataBase.cursor()
    comm = "INSERT INTO Category (Name) VALUES (%s)"
    data = (name))
    dbcurs.execute(comm, data)
    JobDataBase.commit()

def changeCategory (num, name):

```

```

dbcurs = JobDataBase.cursor()
comm = "UPDATE Category SET Name = %s WHERE Id = %s"
data = (name, num)
dbcurs.execute(comm, data)
JobDataBase.commit()

def getCategories ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM Category")
    result = dbcurs.fetchall()
    JobDataBase.commit()
    return result

def getCategorybyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM Category WHERE Category.id
= "+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data

def CreateUserTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS Category (
        Id INT NOT NULL AUTO_INCREMENT,
        Login VARCHAR(255),
        Password VARCHAR(255),
        Status TINYINT,
        PRIMARY KEY (Id)
    ); """)
    JobDataBase.commit()

def addUserTb (login, password, status):
    dbcurs = JobDataBase.cursor()
    comm = "INSERT INTO User (Login,
Password,
Status) VALUES (%s, %s, %s)"

```

```

        data = (name))
        db curs.execute(comm, data)
        JobDataBase.commit()

def changeUser (num, name):
    db curs = JobDataBase.cursor()
    comm = "UPDATE User SET Name = %s WHERE Id = %s"
    data = (name, num)
    db curs.execute(comm, data)
    JobDataBase.commit()

def getUsers ():
    db curs = JobDataBase.cursor()
    db curs.execute("SELECT * FROM User)
    result = db curs.fetchall()
    JobDataBase.commit()
    return result

def getUserbyid(num):
    db curs = JobDataBase.cursor()
    db curs.execute("SELECT * FROM User WHERE User.id =
"+str(num))
    data = db curs.fetchone ()
    JobDataBase.commit()
    return data

def CreateRequirementsTb():
    db curs = JobDataBase.cursor()
    db curs.execute("""CREATE TABLE IF NOT EXISTS Requirements
(
    Id INT NOT NULL AUTO_INCREMENT,
    Name VARCHAR(255),
    PRIMARY KEY (Id)
); """)
    JobDataBase.commit()

```

```

def addRequirementsTb (name):
    db curs = JobDataBase.cursor()
    comm = "INSERT INTO Requirements (Name) VALUES (%s)"
    data = (name))
    db curs.execute(comm, data)
    JobDataBase.commit()

def changeRequirements (num, name):
    db curs = JobDataBase.cursor()
    comm = "UPDATE Requirements SET Name = %s WHERE Id = %s"
    data = (name, num)
    db curs.execute(comm, data)
    JobDataBase.commit()

def getRequirements ():
    db curs = JobDataBase.cursor()
    db curs.execute("SELECT * FROM Requirements)
    result = db curs.fetchall()
    JobDataBase.commit()
    return result

def getRequirementsbyid(num):
    db curs = JobDataBase.cursor()
    db curs.execute("SELECT      *      FROM      Requirements      WHERE
Requirements.id = "+str(num))
    data = db curs.fetchone ()
    JobDataBase.commit()
    return data

def CreateVacancyRequirementsTb():
    db curs = JobDataBase.cursor()
    db curs.execute("""CREATE      TABLE      IF      NOT      EXISTS
VacancyRequirements (
        Id INT NOT NULL AUTO_INCREMENT,
        Requirements_id INT,
        Vacancy_id INT,
        Value VARCHAR(255),

```

```

        PRIMARY KEY (Id),
        FOREIGN KEY (Vacancy_id) REFERENCES Vacancy (id),
        FOREIGN KEY (Requirements_id) REFERENCES Requirements
(id)

); """
JobDataBase.commit()

def addVacancyRequirementsTb (requirements_id, vacancy_id,
value):
    dbcurs = JobDataBase.cursor()
    comm = """INSERT INTO VacancyRequirements
(Requirements_id,
Vacancy_id,
Value)
VALUES (%s, %s, %s) """
    data = (requirements_id, vacancy_id, value)
    dbcurs.execute(comm, data)
    JobDataBase.commit()

def changeVacancyRequirements (num, requirements_id,
vacancy_id, value):
    dbcurs = JobDataBase.cursor()
    comm = """UPDATE VacancyRequirements
SET Requirements_id = %s,
Vacancy_id = %s,
Value = %s
WHERE Id = %s"""
    data = (requirements_id, vacancy_id, value, num)
    dbcurs.execute(comm, data)
    JobDataBase.commit()

def getVacancyRequirements ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM VacancyRequirements")
    result = dbcurs.fetchall()
    JobDataBase.commit()
    return result

```

```

def getVacancyRequirementsbyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM VacancyRequirements WHERE
VacancyRequirements.id = "+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data

```

```

def CreateCandidateRequirementsTb():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("""CREATE TABLE IF NOT EXISTS
CandidateRequirements (
    Id INT NOT NULL AUTO_INCREMENT,
    Requirements_id INT,
    Candidate_id INT,
    Value VARCHAR(255),
    PRIMARY KEY (Id),
    FOREIGN KEY (Vacancy_id) REFERENCES Vacancy (id),
    FOREIGN KEY (Requirements_id) REFERENCES Requirements
(id)

); """)
    JobDataBase.commit()

```

```

def addCandidateRequirementsTb (requirements_id,
candidate_id, value):
    dbcurs = JobDataBase.cursor()
    comm = "INSERT INTO CandidateRequirements
(Requirements_id,
Candidate_id,
Value) VALUES (%s, %s, %s)"
    data = (requirements_id, vacancy_id, value))
    dbcurs.execute(comm, data)
    JobDataBase.commit()

```

```

def getCandidateRequirements ():
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM CandidateRequirements)

```

```
result = dbcurs.fetchall()
JobDataBase.commit()
return result
```

```
def getCandidateRequirementsbyid(num):
    dbcurs = JobDataBase.cursor()
    dbcurs.execute("SELECT * FROM CandidateRequirements WHERE
CandidateRequirements.id = "+str(num))
    data = dbcurs.fetchone ()
    JobDataBase.commit()
    return data
```

ДОДАТОК В
Слайди презентації

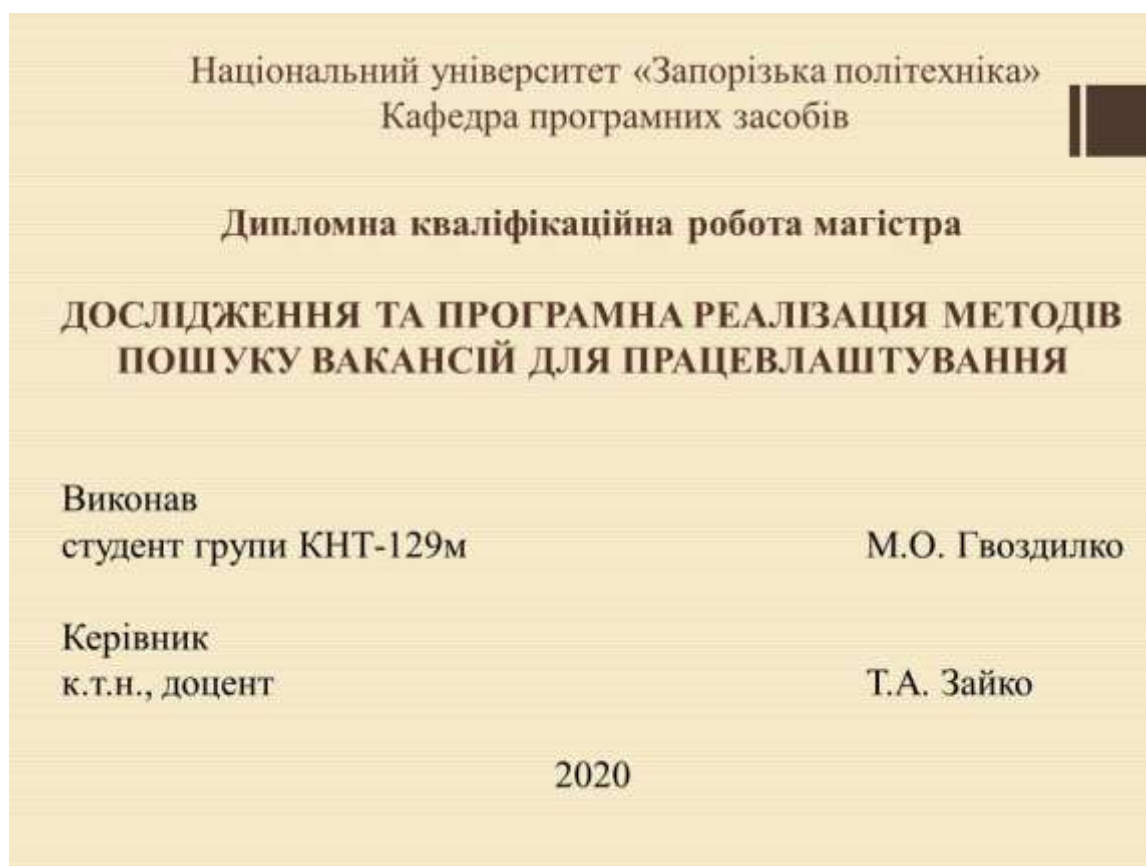


Рисунок В.1 – Слайд 1

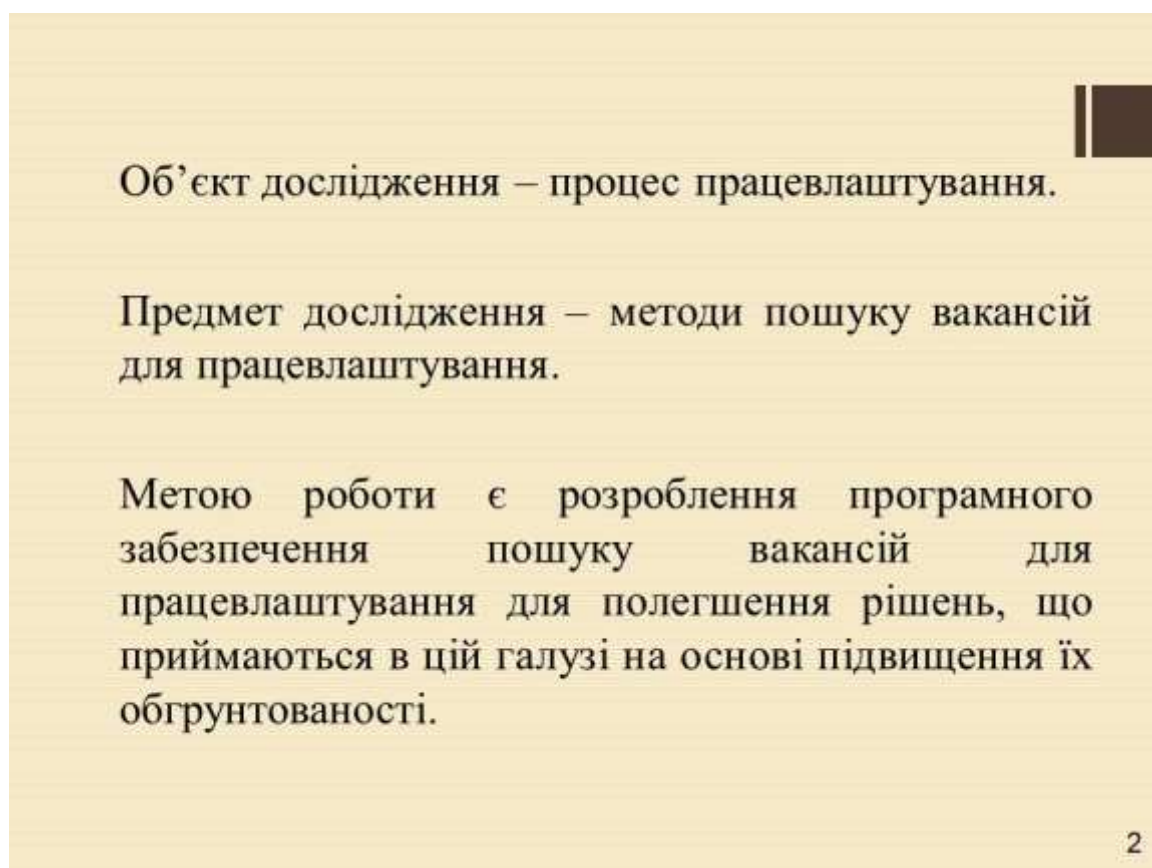


Рисунок В.2 – Слайд 2

Завдання

- ✓ розробка методу пошуку вакансій для працевлаштування;
- ✓ проектування програми;
- ✓ розробка програми;
- ✓ розробка організаційно-економічної частини;
- ✓ розробка заходів з охорони праці та цивільної безпеки.

3

Рисунок В.3 – Слайд 3

Метод пошуку вакансій для працевлаштування

На першому етапі роботи методу реалізується виконання кодування текстових даних для кожної вакансії за допомогою моделі word2vec.

На другому етапі формується набір значень, кожне з яких характеризує відповідну вакансію і характеризується зі сформованого набору описаних вище ознак і кодового представлення тексту вакансії після виконання нормалізації.

Ознаки, що визначають вакансію, зокрема включають:

- вид зайнятості;
- рівень посади;
- вік;
- стать;
- розмір очікуваної зарплати;
- досвід роботи;
- рівень освіти.

На наступному етапі формується вибірка даних, що представляє вакансії, які зацікавили користувачів. Кожен екземпляр – це набір вакансій, які зацікавили користувача з номером, що відповідає номеру екземпляра.

На наступному етапі подається нова вакансія, для якої визначається шляхом застосування методу k-найближчих сусідів вакансії, які можуть зацікавити здобувача, який подав запит на вхідну вакансію.

Обчислення міри схожості запропоновано виконувати на основі косинусної міри:

$$s_{i,j}^x = \cos(\eta_i^x m_j^x)$$

де $s_{i,j}^x$ – подібність за матрицею, що представляє набір значень ознак, які були описані вище;

m_i^x – набір значень ознак для i-ої вакансії;

m_j^x – набір значень ознак для j-ої вакансії.

За розрахованими значеннями визначаються K найближчих сусідів, на основі приналежності яких до виділеного класу і визначається, до якого класу належить дана вакансія. Відповідно в результаті найближчі вакансії з того ж самого класу і видаються як результат роботи методу.

4

Рисунок В.4 – Слайд 4



Рисунок В.5 – Слайд 5

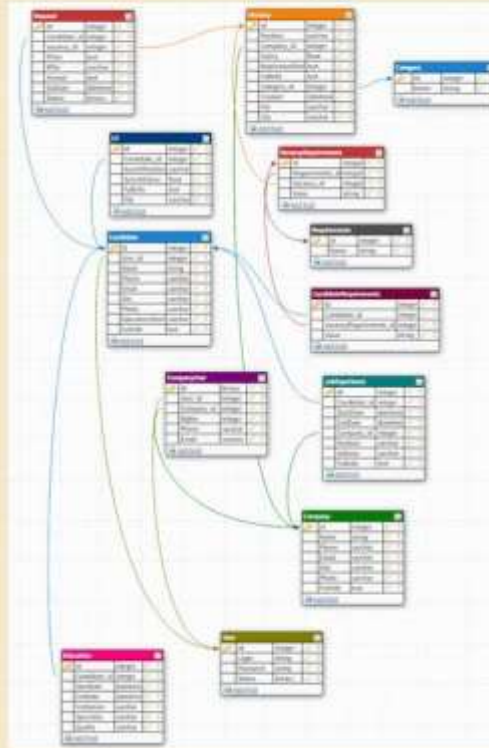
Вибір мови програмування

Критерій	PHP	Python
Бібліотеки	Загального призначення	Велика підтримка спільноти
Засоби роботи з даними	Присутні	Велика кількість бібліотек
Веб-програмування	Зручне з наявністю фреймворків	Зручне з наявністю фреймворків
Гнучкість засобів програмування	Гнучкий рівень	Вищий рівень

6

Рисунок В.6 – Слайд 6

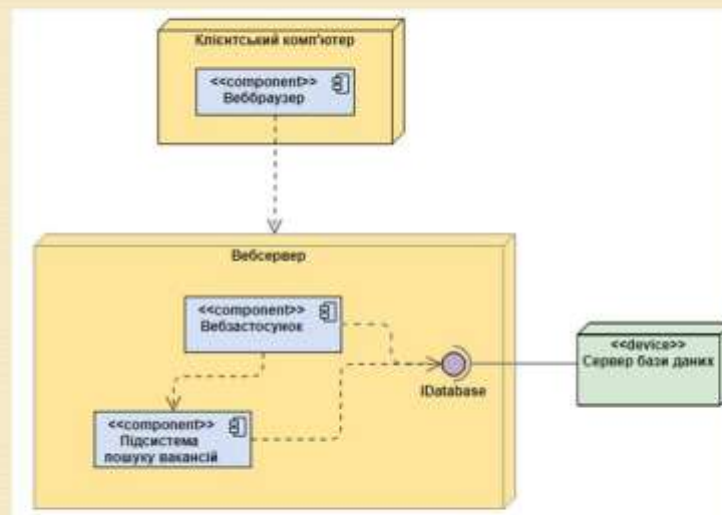
Схема бази даних



7

Рисунок В.7 – Слайд 7

Діаграма компонентів



8

Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9

Пошук вакансій для працевлаштування

Пошук вакансій

Пошук

Місто

30000

☒ Останні

ШУКАТИ

BACK-END
РОЗРОБНИК
- РАЙОФАЙЗЕН -
- БАНК АВАЛЬ -

RETENTION-
МЕНЕДЖЕР
- CONSULTING -
- MEDIA GROUP -

FULL STACK
РОЗРОБНИК
- COSMOLAB -

Рисунок В.10 – Слайд 10

Перегляд вакансій

Back-end розробник

КОМПАНІЯ:

- asksoft.com

МІСТО:

- Київ

ТЕЛЕФОН:

- +380442344027

САЙТ:

- <http://asksoft.com>

ОГОЛОШЕННЯ:

- Постійна робота у навігаційній компанії міста, технічний графік, відсутність командних змін.

ЗАРОБІТНА ПЛАТНЯ ВІД:

- 30 000 грн

ОСОВ'ЯЗКОВІ ВИМОГИ

- Здатність вивести програмний код.
- Знання Java / EE.
- Знання баз даних: MySQL, PostgreSQL.
- Знання: JSP, MVC.

ПЛЮСИ

- Додаткова інформація про компанію.
- Розуміння принципів роботи.

ОСОВ'ЯЗКИ

- Англійська.
- Навчання.
- Інформаційні ресурси.

ПРОПОНУЄМО

- Технічний графік.
- Team building.
- Інформаційні ресурси.

ПОДАТИ ЗАЯВУ

Рисунок В.11 – Слайд 11

Пошук подібних вакансій

Останній поданий запит

JAVA РОЗРОБНИК
- ASK-SOFT -

Подібні вакансії

SENIOR WEB-РОЗРОБНИК
- SOFT TRADE -

JAVA РОЗРОБНИК
- SOFTPRO -

JAVA РОЗРОБНИК
- COSMOLAB -

Рисунок В.12 – Слайд 12

Висновки

Програмно реалізовано вебзастосунок пошуку вакансій для працевлаштування, який дозволяє вносити дані наявних вакансій на працевлаштування, створювати резюме кандидатів, переглядати вакансії на працевлаштування, виконувати пошук вакансій, підбирати вакансії за резюме кандидата та за обраними вакансіями кандидатом, вносити зміни в вакансії, вносити запити на працевлаштування, переглядати найбільш популярні за запитами на працевлаштування компанії.

Наукова новизна роботи полягає в розробленому методі пошуку вакансій для працевлаштування, що виконує пошук подібних вакансій на основі їх класифікації за допомогою методу k-найближчих сусідів, використовуючи в якості додаткових ознак закодовані за допомогою моделі word2vec значення, що дозволяє обґрунтовано приймати рішення про вибір вакансій або резюме для детального аналізу.

13

Рисунок В.13 – Слайд 13