

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ВИЗНАЧЕННЯ ДОСТОВІРНОСТІ ВІДГУКІВ ПРО ПРАЦЕДАВЦІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
VERIFYING THE REVIEWS ABOUT EMPLOYERS

Виконав: студент(ка) 2 курсу, групи КНТ-211м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

Полумієнко Д.О.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Коцур М.І.

(прізвище та ініціали)

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

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Полумієнка Дмитра Олександровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів визначення достовірності відгуків про працедавців
Research and Software Implementation of Methods for Verifying the Reviews about Employers

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “4” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) грудень 2022 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

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	Льовкін В.М., доцент		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “1” вересня 2022 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	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	13 тиждень	

Студент(ка)

(підпис) Полумієнко Д.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Льовкін В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
99 с., 25 рис., 3 табл., 3 дод., 17 джерел.

ПОШУК РОБОТИ, ПРАЦЕДАВЕЦЬ, ВІДГУК, ЗАДАЧА
КЛАСИФІКАЦІЇ, ДОСТОВІРНІСТЬ.

Об'єктом виконаного дослідження є процес пошуку роботи, а предметом – методи визначення достовірності відгуків про працедавців.

Мета дослідження полягала в розробці програмного забезпечення надання доступу до відгуків про працедавців для підтримки шукачів роботи під час вибору працедавців.

Матеріали, методи та технічні засоби: мова Python, система для роботи з базами даних MySQL, фреймворк веброзробки Django, пакет scikit-learn.

Результати. Результати роботи охоплюють програмне забезпечення, що підтримує як пошук роботи через доступ до оголошень про вакансії, подання запитів на роботу, керування цими запитами, так і отримання інформації про працедавців через відгуки, а зокрема і перехід від оголошення до відгуків за цією посадою. Однією з відмінностей роботи є визначення достовірності відгуків, що дозволяє шукачу роботи розуміти, на скільки варто довіряти відомостям з відгуку.

Висновки. Науковою новизною роботи є метод визначення достовірності відгуків про працедавців, який дозволяє класифікувати відгуки про працедавців щодо їх достовірності або недостовірності, виконуючи векторизацію тексту відгуку за допомогою моделі TF-IDF і використовуючи алгоритм RF для реалізації класифікації.

Галузь використання – ринок праці у розрізі процесу пошуку роботи і вибору працедавців.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 99 pages, 25 figures, 3 tables, 3 appendixes, 17 sources.

JOB SEARCH, EMPLOYER, REVIEW, CLASSIFICATION PROBLEM, RELIABILITY.

The object of the research is the job search process, and the subject is methods of determining the reliability of reviews about employers.

The purpose of the study was to develop software to provide access to employer reviews to support job seekers in selecting employers.

Materials, methods and technical means: Python language, MySQL database system, Django web development framework, scikit-learn package.

The results. Job outcomes include software that supports both job searching by accessing job postings, submitting job requests, managing those requests, and getting information about employers through reviews, including transitioning from an ad to reviews for that position. One of the features of the work is the determination of the reliability of the reviews, which allows job seeker to understand the truthworth level of information from a review.

Conclusions. The scientific novelty of the work is the method for determining the reliability of reviews about employers, which allows classifying reviews about employers regarding their reliability or unreliability by performing vectorization of review text using the TF-IDF model and using the RF algorithm to implement the classification.

The field of use is the labor market in terms of the job search process and the selection of employers.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Значення відгуків про працедавців для процесу пошуку роботи	10
1.2 Аналіз можливостей програмного забезпечення надання доступу до відгуків про працедавців і пошуку роботи	11
1.3 Висновки за розділом 1	15
2 Матеріали і методи.....	17
2.1 Задача визначення достовірності відгуків про працедавців	17
2.2 Оброблення текстів природніми мовами.....	17
2.3 Алгоритм класифікації Random Forest.....	19
2.4 Метод визначення достовірності відгуків про працедавців	20
2.5 Висновки за розділом 2	22
3 Проектування програмного забезпечення	23
3.1 Вибір фреймворку для розробки вебдодатків.....	23
3.2 Визначення функціональних вимог до програми.....	24
3.3 Визначення структури бази даних	29
3.4 Висновки за розділом 3	43
4 Основні рішення щодо реалізації компонентів системи.....	44
4.1 Логіка функціонування програми	44
4.2 Класи вебдодатку	46
4.3 Висновки за розділом 4	51
5 Експлуатація, тестування та експериментальне дослідження програми	52
5.1 Призначення програми	52
5.2 Умови виконання програми	52
5.3 Виконання програми.....	52
5.4 Експериментальне дослідження	57
5.5 Висновки за розділом 5	58

Висновки	59
Перелік джерел посилання	61
Додаток А Технічне завдання	63
Додаток Б Текст програми	68
Додаток В Слайди презентації.....	92

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

RF – Randrom Forest;

TF-IDF – Term Frequency-Inverse Document Frequency;

БД – база даних.

ВСТУП

На ринку праці взаємодіють дві сторони: працедавці і потенційні працівники. При цьому працедавці бажають знайти потрібних їм працівників, а працівники – знайти такого працедавця, який би повністю задовольнив їх вимоги. Враховуючи останнє, поточні працівники можуть також шукати роботу, тобто знаходитись в дослідженні ринку, в пошуку роботи, продовжуючи працювати на іншій роботі. Тому працівники потребують інформації про те, чи є робота компанії достатньо цікавою, приносить відповідне задоволення, з точки зору реальних працівників цієї компанії. Отримання такої інформації є важливими джерелом прийняття рішень потенційних працівників про те, чи слід оформлювати трудові відносини з такою компанією.

Програму, яка зможе підтримувати процес пошуку роботи, необхідно створити в дипломній кваліфікаційній роботі магістра. При цьому потрібно враховувати, що представлені відгуки про працедавця можуть спотворити уявлення потенційних працівників без перевірки їх достовірності. Тому окрім просто можливості створити відгук про працедавця, і іншому працівнику – побачити такий відгук, програма повинна надати можливість різного роду контролю за тим, хто і як висловив свою думку. На це має бути спрямовано окрему увагу, адже тільки в синергії різних способів підтвердження інформації і наявності самої інформації про думки працівників стосовно своєї роботи у певного працедавця можна створити програмне забезпечення, яке буде мати дійсно практичну користь і приверне увагу різних учасників ринку праці в підсумку. У випадку, якщо таке програмне забезпечення дійсно зможе привернути увагу учасників ринку, це в підсумку може сприяти тому, щоб учасники ринку ставилися ще відповідальніше стосовно виконання своєї роботи з керування людським капіталом, сприяючи таким чином захисту інтересів усіх сторін.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Значення відгуків про працедавців для процесу пошуку роботи

Відгуки давно стали важливим елементом пошуку. Коли люди купляють товари, знімають житло, вони користуються відгуками покупців або клієнтів, які вже скористалися такими можливостями. З розвитком інтернет-бізнесу відгуки активно ставали частиною цього розвитку. Це ж саме стосується і ринку праці, де відгуки поступово стають важливою складовою, хоча раніше також були нею, адже люди обирали компанії, в яких працюють, отримуючи про них інформацію часто від інших людей. Але з застосуванням програмного забезпечення поширеність такої інформації зростає.

Результати дослідження [1] вказують на наступне:

- відгуки про компанію впливають на рішення шукачів розпочати або уникнути процесу подачі заявки у відповідну компанію, тому скепсис компаній щодо вебсайтів відгуків співробітників є необґрунтованим;
- компаніям слід усвідомити сильний вплив, який відгуки мають на їх онлайн репутацію в очах шукачів роботи та клієнтів: згідно з дослідженнями, підсумованими Glassdoor, 62 % шукачів роботи кажуть, що їхнє сприйняття компанії покращується після того, як роботодавець відповідає на відгук, однак, згідно з нещодавнім аналізом, лише 12% компаній, представлених на Glassdoor, взаємодіють із вебсайтом, створюючи обліковий запис, який, серед інших функцій дозволяє їм відповідати на відгуки;
- конструктивні відповіді компанії повертають потенційно втрачену довіру: менеджери мають відповідати на негативні відгуки належним чином, окрім того варто завчасно запобігати негативним відгукам про компанію, адже якщо компанія опиниться в ситуації, коли існують негативні відгуки, наступні позитивні відгуки призведуть до розбіжностей між відгуками, а отже, не створять бажаної позитивної репутації компанії [1].

Тому в подальшому потрібно проаналізувати, які можливості щодо даного процесу вже існують в програмному забезпеченні, як це

використовується і яким чином можна сприяти розвитку цієї галузі в подальшому.

1.2 Аналіз можливостей програмного забезпечення надання доступу до відгуків про працедавців і пошуку роботи

Результати аналізу вказують, по-перше, що в Україні зокрема популярні вебсайти пошуку роботи не містять можливості переглянути і залишити відгук про працедавця (як наприклад, work.ua [2]), але і спеціалізованого популярного вебсайту з репутацією, де можна отримати достатньо достовірну репутацію про компанії-працедавців, немає.

У світі тим не менш є різні інструменти для того, щоб таку інформацію отримати. Це і вебсайти, які направлені саме на роботу з відгуками і які мають серйозну репутацію в цій сфері, це і вебсайти пошуку роботи, де є така можливість як додаткова.

Glassdoor (рис. 1.1) – це вебсайт, який надає доступ до відгуків про працедавців і відповідно дозволяє створювати такі відгуки працівникам, які на поточний час або в минулому працювали в цій компанії [3]. За кожним відгуком окрім його тексту встановлюється оцінка за 5-бальною шкалою. Оцінка також окремо виставляється за балансом роботи/життя, за культурою і цінностями, за кар'єрними можливостями, за керуванням, за фінансовою стороною.

Також за компаніями надається інформація про доступні посади, про заробітну платню. Надається можливість шукати компанії, шукати самі роботи (посади), отримуючи по ним інформацію. Виставлені оцінки розбиваються на категорії і надається можливість переглянути узагальнену оцінку для працівників за етнічною ознакою, за статтю тощо.

Careerbliss (рис. 1.2) є подібним вебсайтом, який окрім того надає можливість шукати вакансії, при цьому вебсайт працює як певний агрегатор, оскільки після короткого опису вакансії перенаправляє на вебсайт відповідної компанії [4].



Companies

Keyword or Company

Q

Jobs | Salaries | Reviews | Companies | Careers | Tips & Trends

Home > Best Buy > Employee Reviews

Best Buy

★★★★★ (1,256 reviews)

Submit Review

Industry: Electronics and Appliance Stores · HQ: Minneapolis, MN · <https://www.bestbuy.com>

Overview | Jobs | Salaries | Reviews

Best Buy Employee Job Reviews in the United States

Browse Best Buy Reviews by Job Title →

Job Title

Location

All

All

Sort by

Date

Rating

★★★★★

23%

★★★★☆

46%

★★★★☆

24%

★★★☆☆

7%

★★☆☆☆

0%

3.8

Average Rating

(based on 723 Best Buy Review Ratings)

Ratings by Category

Company Culture

3.7

People You Work With

4.4

Rewards You Receive

3.4

Way You Work

4.0

Growth Opportunities

3.2

Person You Work For

4.1

Support You Get

3.8

Work Setting

4.0

Advertisement

Spotlight

Home Theater Installer

"Best Buy has been an amazing place to both learn and grow within this field."

Posted 13 years ago in Sterling, VA

Best Buy Employee

★★★★☆

in Charlotte, NC

"It can be fun and has great benefits and the discount is awesome. Hard time finding mentors or anyone to help train or move up for that matter. They don't seem to care about furthering you in any type of career that is fulfilling and worth while. The gap between entry level and any type of leadership has been stretched more and more over the years to make it incredibly difficult to move up."

Person You Work For

4 / 5

People You Work With

4 / 5

Work Setting

4 / 5

Support You Get

4 / 5

Rewards You Receive

3 / 5

Growth Opportunities

2 / 5

Company Culture

2 / 5

Way You Work

4 / 5

Advertisement

Be the first to see new Best Buy jobs

1 Best Buy

2 Your Location

3 Your email address

Subscribe

You can cancel email alerts at any time.

Trending Jobs

Clinical Product Owner (Remote)

PNT Data - Middletown, CT

VP of Sales & Marketing

Nationwide Technical Resources Corp - Bohemia, NY

Media Strategist

Known - New York City, NY

Program Specialist, Recommendations

Staples - Framingham, MA

Best Buy Reviews by Location

Richfield, MN

28 reviews

Minneapolis, MN

14 reviews

Houston, TX

12 reviews

Miami, FL

9 reviews

Los Angeles, CA

8 reviews

Orlando, FL

7 reviews

San Jose, CA

6 reviews

Eden Prairie, MN

6 reviews

Orange, CA

6 reviews

Рисунок 1.2 – Відгуки про працедавців на вебсайті Careerbliss

У всьому іншому ці вебсайти дуже подібні між собою. В даному випадку дещо інші складові оцінки виявлені, хоча вони дуже подібні (люди, з якими працюєте, спосіб роботи, винагороди, культура компанії, можливості росту, люди, на яких працюєте, отримувана підтримка, робочі умови).

На відміну від попередніх варіантів, Indeed (рис. 1.3) навпаки є всесвітньовідомим вебсайтом пошуку роботи і містить в першу чергу вакансії [5]. Але окрім вакансій надаються ще й відгуки про працедавців, тому цей вебсайт було окремо розглянуто під час аналізу.

The screenshot displays the Indeed website interface. At the top, the Indeed logo is on the left, and navigation links like 'Find jobs', 'Company reviews', 'Find salaries', 'Upload your resume', 'Sign in', and 'Employers / Post Job' are on the right. Below the header is a large banner for 'ARMY VALUES' with various military-related terms like 'LOYALTY', 'DUTY', 'SELFLESS SERVICE', 'RESPECT', 'HONOR', 'INTEGRITY', 'PERSONAL COURAGE', and 'LOYALTY' repeated in different colors and sizes. Below the banner is the U.S. Army company profile. It shows a 4.3 rating from 79.6K reviews, a 'Follow' button, and a 'Write a review' button. Below the profile is a 'Review this company' button. A detailed review for 'Truck driver' is shown, including a 4.0 rating, a star rating, and the text of the review. A 'View all 79,567 reviews' button is at the bottom of the review section. On the right, there is a 'Claimed Profile' status and a 'Want to know more about working here?' section with an 'Ask a Question' button.

Рисунок 1.3 – Відгуки про працедавців на вебсайті Indeed

У відгуку може бути встановлено і посаду, на якій працював працівник у компанії, також зазначається оцінка, текст відгуку. Можна окремо виділити позитивні і негативні риси.

Порівняння програми з дипломної кваліфікаційної роботи магістра з аналогами виконано в таблиці 1.1.

Таблиця 1.1 – Порівняння програмних аналогів

Критерій	Glassdoor	Indeed	Empviewer
Пошук вакансій	—	+	+
Подання запитів на роботу	—	+	+
Перегляд відгуків про працедавців	+	+	+
Додавання власних відгуків про працедавців	+	+	+
Визначення достовірності відгуків	—	—	+
Створення збережених списків	+	+	+

1.3 Висновки за розділом 1

Проведений аналіз стану відгуків про працедавців у галузі ринку праці продемонстрував, що вони мають достатньо великий вплив на шукачів роботи. Аналіз аналогів вказав на те, що підтвердженню даних відгуків приділяється недостатня увага: відсутні механізми автоматичного підтвердження достовірності відгуків, недостатньо задіяні можливості підтвердження і відповідей представників компаній на ці відгуки.

Завдання роботи для забезпечення виконання мети охоплюють:

- проаналізувати програмні засоби надання доступу до відгуків про працедавців;
- виконати дослідження методів визначення достовірності відгуків про працедавців;

- виконати проєктування програмного забезпечення надання доступу до відгуків про працедавців;
- реалізувати програмне забезпечення надання доступу до відгуків про працедавців.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача визначення достовірності відгуків про працедавців

Задача визначення достовірності відгуків про працедавців є задачею класифікації, розв'язання якої в автоматизованому режимі дозволяє без участі працедавця зрозуміти, на скільки достовірною є інформація, зазначена у відгуку, чи їй взагалі не варто довіряти.

Для визначення достовірності відгуку потрібно отримати спостереження, яке характеризує цей відгук, і за отриманими даними визначити вихідну ознаку – клас достовірності. Логічно використовувати бінарний розподіл, тобто відгук бо слід вважати достовірним, або недостовірним. Якщо відгук недостовірний, то відповідна довідкова інформація дозволить захистити інтереси працедавця, якщо у відгуку зазначена недостовірна негативна інформація, або навпаки захистити інтереси працівника, якщо у відгуку приведена недостовірна позитивна інформація, коли можливо штучно збільшуються позитивні якості компанії.

Відгук про працедавця характеризується своїм текстом, а також набором інших ознак, що визначають відношення працівника до роботи в компанії. Набір цих ознак має бути в підсумку визначений при розв'язанні задачі, щоб в підсумку можна було накопичувати дані саме за потрібними ознаками і мати повністю значення за кожною потрібною ознакою за кожним відгуком для випадків, коли щонайменше працівник погодився внести всю потрібну інформацію.

2.2 Оброблення текстів природніми мовами

Обробка природної мови – це гілка штучного інтелекту, яка допомагає комп'ютерам розуміти, інтерпретувати та отримувати значення людської мови: починається з попередньої обробки даних, яка, по суті, є сортуванням і очищенням даних, щоб привести їх до загальної структури, зрозумілої для

алгоритму (попередня обробка текстових даних має на меті відформатувати текст у спосіб, який модель може зрозуміти та навчитися, щоб імітувати людське розуміння), далі за допомогою векторизації тексту обробка природної мови перетворює тексти на щось, що може зрозуміти машина (вектори чисел), ці перетворені частини передаються в систему для створення алгоритму машинного навчання [6]-[7].

Існують два основні методи, які дозволяють векторизувати текст: модель «торби слів» та модель Term Frequency-Inverse Document Frequency (TF-IDF).

Модель «торби слів» перетворює кожен текст на основі словника, визначаючи, скільки разів кожне слово зі словника входить в заданий текст [8]-[9]. Тобто фактично ґрунтується на інформації про те, як часто слова повторюються.

Модель TF-IDF навпаки намагається виділити унікальні слова.

Для обчислення TF-IDF використовуються такі обчислення (2.1)-(2.3):

$$TF - IDF = TF \cdot IDF, \quad (2.1)$$

$$TF = \frac{WN}{DW}, \quad (2.2)$$

$$IDF = \log\left(\frac{DN}{DWN}\right), \quad (2.3)$$

де TF – частота слова порівняно з частотою слів у тексті;

IDF – фактор, який зменшує важливість запитів, які є спільними для багатьох текстів;

WN – кількість разів, яку слово зустрілось у тексті;

DW – кількість слів у документі;

DN – кількість документів;

DWN – кількість документів, які містять слово [10].

2.3 Алгоритм класифікації Random Forest

Random Forest (RF) – це керований алгоритм машинного навчання, у якому для розв’язання задачі класифікації, а також для розв’язання задачі прогнозування, для якої він також може використовуватися, обчислення цілого ансамблю дерев рішень об’єднуються для отримання одного остаточного результату [11]-[14].

Випадковий ліс складається з великої кількості цих дерев рішень, які працюють разом як так званий ансамбль, при цьому кожне окреме дерево рішень робить прогноз, наприклад, результат класифікації, і ліс використовує результат, який підтримується більшістю дерев рішень, як прогноз для всього ансамблю, а загалом секрет алгоритму RF полягає в так званому принципі мудрості натовпу, де основна ідея полягає в тому, що рішення багатьох завжди краще, ніж рішення окремої людини чи одного дерева рішень [13].

У алгоритмі RF застосовуються наступні правила:

- різні дерева рішень навчаються за допомогою навчальних даних: цей набір даних містить спостереження та характеристики, які будуть вибрані випадковим чином під час розділення вузлів, у системі RF використовуються різні дерева рішень;

- у дереві рішень є три типи вузлів: вузли рішень, листові вузли та кореневі вузли, при цьому листовий вузол кожного дерева представляє остаточний результат, створений цим конкретним деревом рішень, кінцевий результат обирається за допомогою процедури більшості голосів, у цій ситуації кінцевий вихід системи RF є результатом, обраним більшістю дерев рішень;

- ансамбль означає процес злиття багатьох моделей, а ансамблеві підходи поділяються на дві категорії: пакетування (bagging) – створює окрему навчальну підмножину із заміною на вибіркові навчальні дані, а кінцевий результат базується на більшості голосів, а посилення (boosting) –

перетворює слабких учнів на хороших, генеруючи послідовні моделі з максимально можливою точністю [11].

На результати роботи алгоритму RF можна вплинути наступним чином на основі використання відповідних гіперпараметрів:

- підвищення прогнозуючої сили: гіперпараметр `n_estimators` визначає кількість дерев, які будує алгоритм, при цьому більша кількість дерев підвищує продуктивність і робить прогнози більш стабільними, але це також уповільнює обчислення, гіперпараметр `max_features` визначає максимальну кількість ознак, які випадковий ліс розглядає для розділення вузла, гіперпараметр `min_sample_leaf` визначає мінімальну кількість листків, необхідних для розбиття внутрішнього вузла;

- збільшення швидкості моделі: гіперпараметр `n_jobs` визначає, скільки процесорів дозволено використовувати, відповідно якщо він має значення одиниці, він може використовувати лише один процесор, гіперпараметр `random_state` робить вихід моделі відтворюваним. Модель завжди даватиме однакові результати, якщо вона має певне значення `random_state` і якщо їй надано ті самі гіперпараметри та ті самі дані навчання, гіперпараметр `oob_score` є методом перехресної перевірки лісу (у вибірці приблизно одна третина даних не використовується для навчання моделі та може бути використана для оцінки її ефективності, ці спостереження називаються спостереженнями з мішка) [14].

2.4 Метод визначення достовірності відгуків про працедавців

Метод визначення достовірності відгуків про працедавців дозволяє розв'язати відповідну задачу як задачу класифікації. Для цього потрібно зібрати всі наявні дані про відгуки про працедавців, щодо яких вже встановлено достовірність, і сформувати з них вибірку даних, що охоплює наступні ознаки:

- текст відгуку, до якого обов’язково мають бути включені позитивні і негативні риси, якщо вони визначаються окремо від загального тексту відгуку;
- оцінка відгуку: використовується загальна оцінка, встановлена за 5-бальною шкалою;
- середня оцінка компанії-працедавця за всіма наявними на даний момент відгуками (також використовується відповідно 5-бальна шкала);
- кількість відгуків про працедавця – поточна наявна кількість відгуків, яку було створено про працедавця;
- клас достовірності відгуку: достовірний або недостовірний відгук.

Оскільки в даній сфері не завжди відгук про працедавця асоціюється з конкретною посадою, то посаду не було винесено до набору ознак, хоча за наявності повних даних для основної кількості відгуків таких даних варто дослідити застосування даної ознаки окремо.

На наступному етапі потрібно виконати попередню обробку тексту відгуку. Під час даного процесу необхідно вилучити всі зайві символи, символи, які не входять до базового алфавіту (на практиці використовувався англійський алфавіт).

Після попередньої обробки має бути виконано векторизацію текстів відгуків за допомогою використання TF-IDF.

На наступному етапі має бути застосовано алгоритм RF для навчання моделі за допомогою вибірки даних, яку було створено на попередніх етапах. Рекомендовано здійснювати навчання на основі логіки, описаної в статті [15].

Після успішного завершення навчання реалізується етап застосування створеної моделі для даних, які ще не вивчались. Це нові відгуки про працедавців, які були отримані вже після навчання або для яких не відомо клас достовірності. Для кожного такого відгуку також має реалізовуватися зведення до форми, за якою вище було представлено структуру ознак (окрім останньої ознаки), а також попереднє оброблення тексту відгуку і векторизація за допомогою TF-IDF. Результат роботи етапу – клас

достовірності, який повернула модель для отриманого відгуку про працедавця.

2.5 Висновки за розділом 2

Задачу визначення достовірності відгуків про працедавців сформульовано як задачу класифікації. Для розв'язання цієї задачі розглянуто методи оброблення природної мови, включаючи модель «торби слів» і модель TF-IDF. Розглянуто алгоритм RF щодо його застосування для розв'язання задачі класифікації.

Науковою новизною роботи є метод визначення достовірності відгуків про працедавців, який дозволяє класифікувати відгуки про працедавців щодо їх достовірності або недостовірності, виконуючи векторизацію тексту відгуку за допомогою моделі TF-IDF і використовуючи алгоритм RF для реалізації класифікації.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір фреймворку для розробки вебдодатків

Виконуючи вибір інструментів, що мають застосовуватися для вирішення задач проєктування і реалізації програмного забезпечення, було виконано вибір на користь мови Python, враховуючи необхідність реалізації методу з розділу 2, що вимагає добре відлагоджених реалізованих методів і засобів, включаючи RF і TF-IDF. До того ж Python дозволяє на високому рівні реалізувати і сам вебдодаток: за сукупністю цих факторів було обрано цю мову, а подальший вибір зведено до вибору фреймворку для розробки вебдодатків (табл. 3.1), яких існує декілька. Серед Django і CherryPy був обраний Django.

Таблиця 3.1 – Порівняння фреймворків для розробки вебдодатків

Показник / фреймворк	Django	CherryPy
Доступність механізму об'єктно-реляційного перетворення	Так	Ні
Адміністративна панель	Так	Ні
Засоби підтримки безпеки за замовчуванням	Так, постійно розширюються	Ні
Аутентифікація користувачів	Так, надаються широкі засоби керування записами користувачів	Так
Підтримувані рушії шаблонів	За замовчуванням, можна змінити	За замовчуванням немає, можна підключити

3.2 Визначення функціональних вимог до програми

Звичайний користувач, який не виконав авторизацію, розглядається як актор з роллю гостя (рис. 3.1).

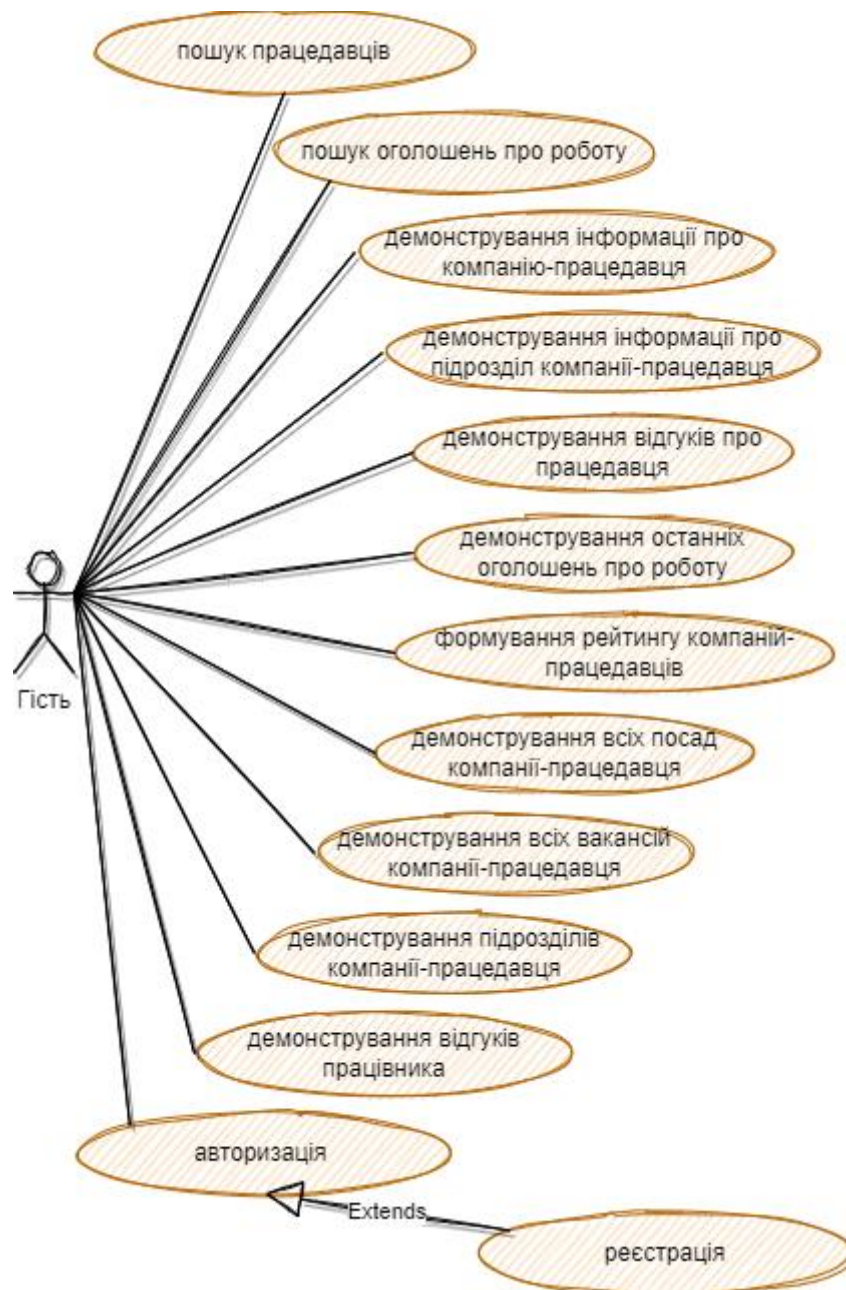


Рисунок 3.1 – Діаграма варіантів використання програми гостем

Гість взаємодіє з програмою через сценарії:

- пошук працедавців;
- пошук оголошень про роботу;

- демонстрування інформації про компанію-працедавця;
- демонстрування інформації про підрозділ компанії-працедавця;
- демонстрування відгуків про працедавця;
- демонстрування останніх оголошень про роботу за критерієм;
- формування рейтингу компаній-працедавців;
- демонстрування всіх посад компанії-працедавця;
- демонстрування всіх вакансій компанії-працедавця;
- демонстрування підрозділів компанії-працедавця;
- демонстрування відгуків працівника;
- авторизація і реєстрація записів користувачів.

Користувач, який авторизувався в програмі визначається як актор з роллю працівника (рис. 3.2). Його основною діяльністю є пошук роботи і надання інформації про працедавців, у яких він працював раніше. Працівник взаємодіє з програмою через сценарії:

- публікація відгуків про працедавців, що включає визначення достовірності відгуку про працедавця;
- пошук працедавців;
- пошук оголошень про роботу;
- демонстрування інформації про компанію-працедавця;
- демонстрування інформації про підрозділ компанії-працедавця;
- демонстрування відгуків про працедавця;
- демонстрування останніх оголошень про роботу за критерієм;
- формування рейтингу компаній-працедавців;
- демонстрування всіх посад компанії-працедавця;
- демонстрування всіх вакансій компанії-працедавця;
- демонстрування підрозділів компанії-працедавця;
- створення списку збережених компаній-працедавців або оголошень про роботу;
- додавання або вилучення компаній-працедавців або оголошень зі збереженого списку;

- демонстрування збереженого списку;
- демонстрування збережених списків користувачем.

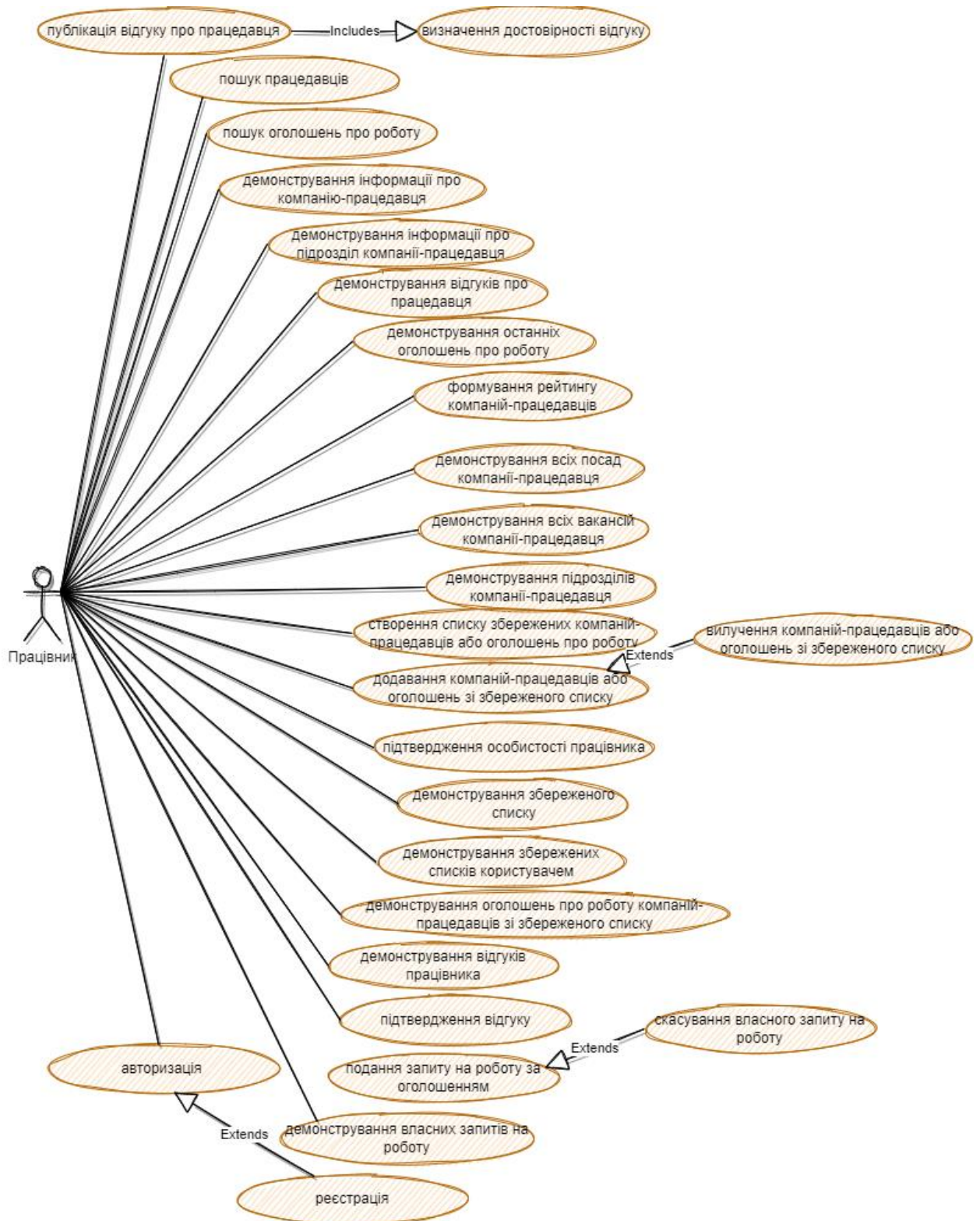


Рисунок 3.2 – Діаграма варіантів використання програми працівником

Також до сценаріїв належать:

- демонстрування оголошень про роботу компаній-працедавців зі збереженого списку;
- демонстрування відгуків працівника;
- підтвердження особистості працівника;
- підтвердження відгуку;
- подання запиту на роботу за оголошенням;
- скасування власного запиту на роботу;
- демонстрування власних запитів на роботу;
- авторизація і реєстрація записів користувачів.

Усі компанії-працедавці повинні мати своїх представників у програмі, якими є менеджери (рис. 3.3). Менеджер взаємодіє з програмою через сценарії:

- демонстрування інформації про власну компанію-працедавця;
- змінення інформації про власну компанію-працедавця;
- визначення менеджерів для представлення власної компанії-працедавця;
- демонстрування інформації про підрозділ компанії-працедавця;
- змінення інформації про підрозділ компанії-працедавця;
- демонстрування відгуків про власну компанію-працедавця;
- демонстрування рейтингу компаній-працедавців;
- визначення посади у власній компанії-працедавці;
- змінення посади у власній компанії-працедавці;
- демонстрування всіх посад власної компанії-працедавця;
- демонстрування всіх вакансій власної компанії-працедавця;
- демонстрування підрозділів власної компанії-працедавця;
- демонстрування відгуків працівника;
- підтвердження або скасування (як альтернативний сценарій) запиту на роботу;
- визначення достовірності відгуку про власну компанію-працедавця;

- публікація оголошення про роботу, що є узагальненим сценарієм до публікації оголошення про роботу на основі шаблону;
- визначення шаблону оголошення про роботу;
- змінення шаблону оголошення про роботу;
- демонстрування запитів на оголошення про роботу.

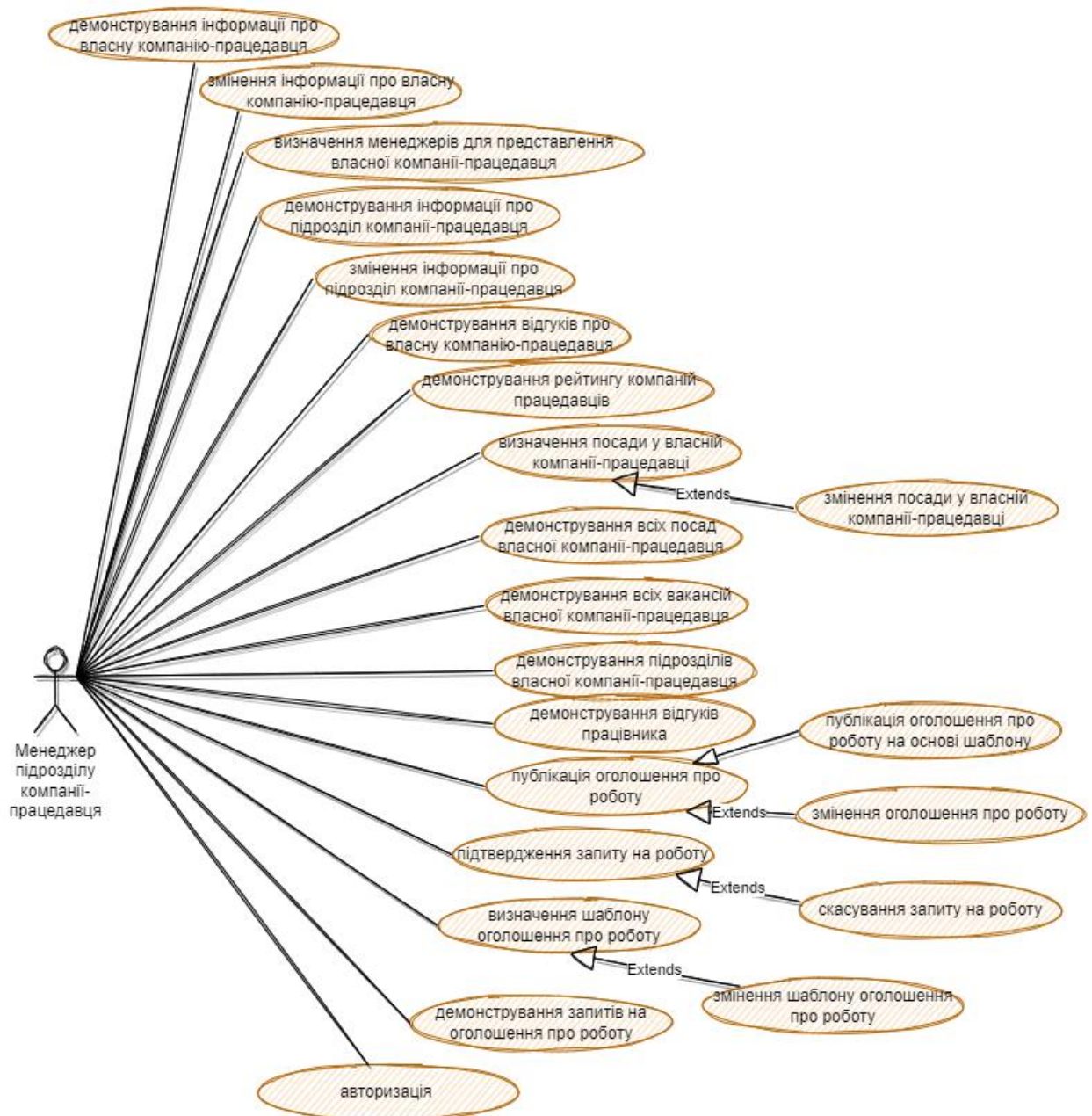


Рисунок 3.3 – Діаграма варіантів використання програми менеджером підрозділу компанії-працедавця

Також до сценаріїв належать:

- змінення оголошення про роботу, що як альтернативні сценарії передбачає подовження дії оголошення про роботу, відкликання оголошення про роботу;

- авторизація.

Користувачі програми можуть також виступати у ролі особливого актора – адміністратора. Адміністратор має можливість працювати зі всіма сутностями, переглядаючи і вносячи відповідні зміни, що загалом у даному випадку можна розглядати як типовий сценарій управління даними. Окрім нього до сценаріїв адміністратора входять:

- офіційне підтвердження особистості працівника;
- скасування недостовірності відгука про працедавця.

3.3 Визначення структури бази даних

Структура бази даних (БД) побудована через застосування системи MySQL. Вона застосовується для практичних проєктів різного масштабу.

Дані стосовно сфер діяльності зберігаються в таблиці Sphere з полями:

- id – номер сфери діяльності;
- titled_as – назва сфери діяльності.

Дані стосовно міст, де можуть знаходитись відділення компаній, зберігаються в таблиці City з полями:

- id – номер міста;
- name – назва цього міста;
- country – країна, в якій знаходиться місто.

Дані стосовно компаній-працедавців зберігаються в таблиці Employer з полями:

- id – номер компанії-працедавця;
- titled_as – назва компанії-працедавця;
- overall – повний опис компанії-працедавця;

- photo – зображення або фотографія, яка ілюструє компанію-працедавця;
- phone – номер телефону для зв'язку з компанією-працедавцем;
- site – вебсайт компанії-працедавця;
- score – усереднена оцінка компанії-працедавця;
- strategyscore – усереднена оцінка стратегії, місії компанії-працедавця.

Дані стосовно підрозділів компаній-працедавців зберігаються в таблиці Department з полями:

- id – номер підрозділу компанії-працедавця;
- titled_as – назва підрозділу компанії-працедавця;
- city – місто, в якому розташований підрозділ компанії-працедавця;
- address – адреса підрозділу компанії-працедавця;
- employer – компанія-працедавець, до якої належить підрозділ;
- overall – повний опис підрозділу;
- score – усереднена оцінка підрозділу компанії-працедавця;
- photo – зображення або фотографія, яка ілюструє підрозділ компанії-працедавця;
- phone – номер телефону для зв'язку з підрозділом компанії-працедавцем;
- site – вебсайт підрозділу компанії-працедавця;
- score – усереднена оцінка компанії-працедавця;
- humanscore – усереднена оцінка стратегії, місії компанії-працедавця.

Серед цих полів є поле city, що є зовнішнім ключем, який належить до таблиці City. Цей зв'язок визначається типом один до багатьох, адже місто може презентувати підрозділи різних компаній-працедавців, а один підрозділ розташований тільки в одному місті.

Ще один зовнішній ключ – це поле employer, що є зовнішнім ключем, який належить до таблиці Employer. Цей зв'язок визначається типом один до

багатьох, адже компанія-працедавець може мати різні підрозділи, а кожен підрозділ – це підрозділ конкретної компанії-працедавця.

Дані стосовно менеджерів компаній-працедавців зберігаються в таблиці `Employermanager` з полями:

- `id` – номер менеджера компанії-працедавця;
- `user` – запис користувача менеджера компанії-працедавця;
- `department` – відділ, якому належить цей менеджер компанії-працедавця;
- `phone` – номер телефону для зв'язку з менеджером компанії-працедавцем;
- `name` – ім'я менеджера компанії-працедавця;
- `surname` – прізвище менеджера компанії-працедавця.

Серед цих полів є поле `department`, що є зовнішнім ключем, який належить до таблиці `Department`. Цей зв'язок визначається типом один до багатьох, адже підрозділ компанії-працедавця може бути презентований різними менеджерами, а один менеджер може працювати тільки в одному підрозділі однієї компанії-працедавці.

Дані стосовно відгуків про роботу на працедавця зберігаються в таблиці `Review` з полями:

- `id` – номер відгуку про роботу на працедавця;
- `overall` – повний текст, що описує враження працівника про роботу на працедавця;
- `department` – підрозділ компанії-працедавця, де працював працівник;
- `positives` – позитивні сторони виконуваної роботи, компанії працедавця;
- `negatives` – негативні сторони виконуваної роботи, компанії працедавця;
- `score` – загальна оцінка в балах, виставлена про роботу на працедавця саме за посадою (професією);

- employerscore – загальна оцінка в балах, виставлена про всю компанію працедавця загалом;
- departmentscore – загальна оцінка в балах, виставлена про підрозділ компанії працедавця загалом;
- interestscore – оцінка в балах, виставлена працівником роботі відносно того, на скільки вона є цікавою;
- financescore – оцінка в балах, виставлена працівником роботі відносно того, на скільки вона є фінансово привабливою;
- strategyscore – оцінка в балах, виставлена працівником роботі відносно того, на скільки він вважає стратегію компанії правильною;
- humanscore – оцінка в балах, виставлена працівником роботі відносно того, на скільки привабливим для нього є колектив;
- workposition – посада, на якій в компанії працював працівник;
- employee – працівник, який залишив цей відгук;
- anonum – позначка про необхідність анонімного відображення відгуку (без вказання авторства);
- workbegin – час, з якого почав працювати на цій посаді працівник;
- workend – час, до якого працював на цій посаді працівник;
- reviewed – час написання відгуку;
- truth – позначка про клас достовірності, до якого було віднесено відгук.

Серед цих полів є поле employee, що є зовнішнім ключем, який належить до таблиці Employee. Цей зв'язок визначається типом один до багатьох, адже працівник може написати відгуки про різних працедавців, а один відгук може стосуватися і бути написаним тільки одним працівником.

Ще один зовнішній ключ – це поле workposition, що належить до таблиці Workposition. Цей зв'язок також визначається типом один до багатьох, адже на одну посаду в конкретній компанії може бути написано різну кількість відгуків, а кожен відгук – це відгук про одну займану посаду цим працівником.

Поле department – зовнішній ключ з таблиці Department. Цей зв'язок також визначається типом один до багатьох, адже на роботу, яка виконувалась в одному підрозділ, може бути написано багато відгуків, а кожен відгук – це відгук, що стосується одного підрозділу.

Дані стосовно базових користувачів, які в поточний момент є працівника або були працівниками, а зараз знаходяться в пошуку нової роботи, зберігаються в таблиці Employee з полями:

- id – номер працівника, зареєстрованого в програмі;
- user – запис користувача-працівника;
- name – ім'я користувача-працівника;
- surname – прізвище користувача-працівника;
- phone – номер телефону для зв'язку з користувачем-працівником;
- overall – повний опис для презентації себе користувачем-працівником, своїх вмінь;
- cv – резюме користувача-працівника;
- site – вебсайт користувача-працівника (фактично портфоліо, резюме або інша деталізована інформація про користувача, яка може використовуватися під час пошуку роботи);
- reviews_created – сумарна кількість відгуків про працедавців, які створив користувач-працівник;
- reviews_approved – сумарна кількість відгуків про працедавців, які створив користувач-працівник і які після того були схвалені працедавцем;
- reviews_confirmed – сумарна кількість відгуків про працедавців, які створив користувач-працівник і які були підтверджені іншими користувачами;
- reviews_official_confirmed – сумарна кількість відгуків про працедавців, які створив користувач-працівник і які були підтверджені саме офіційно підтвердженими користувачами;
- confirmations – сумарна кількість підтверджень ідентичності користувача-працівника іншими користувачами;

– official_confirmations – сумарна кількість підтверджень ідентичності користувача-працівника офіційно підтвердженими користувачами;

– official – показник того, чи підтверджений цей користувач-працівник офіційно, тобто представниками служби підтримки.

Дані стосовно посад в компанії-працедавці зберігаються в таблиці Workposition з полями:

– id – номер посади в компанії-працедавці;
 – profession – професія, якій відповідає посада;
 – employer – компанія-працедавець, в якій використовується і для якої визначена в програмі ця посада;

– score – усереднена оцінка в балах, виставлена працівниками;
 – interestscore – усереднена оцінка в балах, виставлена працівниками роботи за цією посадою відносно того, на скільки вона є цікавою;

– financescore – усереднена оцінка в балах, виставлена працівниками роботи за цією посадою відносно того, на скільки вона є фінансово привабливою;

– strategyscore – усереднена оцінка в балах, виставлена працівниками роботи за цією посадою відносно того, на скільки вони вважають стратегію компанії правильною;

– humanscore – усереднена оцінка в балах, виставлена працівниками роботи за цією посадою відносно того, на скільки привабливими вони вважають колектив (людські якості).

Серед цих полів є поле employer, що є зовнішнім ключем, який належить до таблиці Employer. Цей зв'язок визначається типом один до багатьох, адже компанія-працедавець може мати різні посади, а кожна посада фіксується відносно конкретної компанії-працедавця, а не загалом (для загальних визначень у межах всій компаній одразу існує сутність професії).

Ще один зовнішній ключ виражений полем profession, належить до таблиці Profession. Цей зв'язок визначається типом один до багатьох, адже за

професією визначаються різні посади в компаніях-працедавцях, а кожна посада може визначатися тільки однією професією одночасно.

Дані стосовно професій, за якими відбувається робота, зберігаються в таблиці Profession з полями:

- id – номер професії;
- titled_as – назва професії.

Дані стосовно збережених користувачем списків зберігаються в таблиці Savedlist з полями:

- id – номер збереженого користувачем (працівником) списку;
- named_as – назва, яку встановив користувач збереженому списку;
- employee – користувач-працівник, якому належить цей список;
- offered – позначка, яка визначає, чи в цей список будуть входити оголошення (якщо True), чи компанії-працедавці (якщо False).

Серед цих полів є поле employee, що є зовнішнім ключем, який належить до таблиці Employee. Цей зв'язок визначається типом один до багатьох, адже працівник може мати декілька збережених списків, а кожен список може бути створеним і відповідно належати тільки одному працівнику.

Кожен список може містити збережені компанії-працедавці або оголошення про роботу.

Дані стосовно того, які саме компанії-працедавці були збережені в списку працівника, зберігаються в таблиці SavedlistEmployer з полями:

- id – номер внесення однієї з компаній-працедавців в один зі списків;
- employer – компанія-працедавець, яку було збережено в списку;
- savedlist – збережений список користувача-працівника, до якого було внесено компанію;
- saved – позначка про час збереження цієї компанії-працедавця в цьому списку.

Створена ця таблиця була для того, щоб реалізувати зв'язок між таблицями Savedlist і Employer, який має тип багато до багатьох, адже одна

компанія-працедавець може входити в збережені списки різних користувачів, а кожен список може містити декілька компаній-працедавців, для чого і створюється в принципі.

Дані стосовно того, які саме оголошення про роботу були збережені в списку працівника, зберігаються в таблиці SavedlistWorkoffer з полями:

- id – номер внесення одного з оголошень про роботу в один зі списків;
- workoffer – оголошення про роботу, яке було збережено в списку;
- savedlist – збережений список оголошень про роботу, до якого було внесено це оголошення;
- saved – позначка про час збереження цього оголошення в цьому списку.

Створена ця таблиця була для того, щоб реалізувати зв'язок між таблицями Savedlist і Workoffer, який має тип багато до багатьох, адже одне оголошення про роботу може входити в збережені списки різних користувачів, а кожен список може містити декілька оголошень про роботу, для чого і створюється в принципі.

Дані стосовно підтвердження ідентичності користувачів-працівників зберігаються в таблиці IdentityConfirmation з полями:

- id – номер підтвердження ідентичності користувачів-працівників іншими користувачами;
- employee1 – користувач-працівник, ідентичність якого підтверджено;
- employee2 – користувач-працівник, який підтвердив ідентичність іншого користувача-працівника.

Поле employee1 є зовнішнім ключем, який належить до таблиці Employee. Цей зв'язок визначається типом один до багатьох, адже ідентичність працівника може бути підтверджена багатьма іншими працівниками, а кожна перевірка стосується одного конкретного користувача-працівника.

Поле `employee2` є зовнішнім ключем, який належить до таблиці `Employee`. Цей зв'язок визначається типом один до багатьох, адже один працівник може підтвердити ідентичність багатьох інших працівників, а кожна перевірка виконується одним користувачем-працівником.

Дані стосовно підтвердження відгуків зберігаються в таблиці `ReviewConfirmation` з полями:

- `id` – номер підтвердження відгуку;
- `review` – відгук про працедавця користувача-працівника, який було підтверджено;
- `employee` – користувач-працівник, який підтвердив відгук.

Поле `employee` є зовнішнім ключем, який належить до таблиці `Employee`. Цей зв'язок визначається типом один до багатьох, адже один працівник може підтвердити декілька відгуків про працедавців, а кожна перевірка виконується одним користувачем-працівником.

Поле `review` є зовнішнім ключем, який належить до таблиці `Review`. Цей зв'язок визначається типом один до багатьох, адже один відгук може бути підтверджений різними користувачами-працівниками, а кожна перевірка стосується тільки одного відгуку.

Дані стосовно підтверджень відгуків самим працедавцем, про якого вони створені, зберігаються в таблиці `Approve` з полями:

- `id` – номер підтвердження відгуку компанією-працедавцем;
- `review` – відгук, який було підтверджено;
- `employermanager` – менеджер компанії-працедавця, який підтвердив даний відгук;
- `positive` – реакція на відгук працедавця: повністю підтверджений, частково підтверджений, не згоден;
- `answer` – відповідь на відгук.

Поле `review` є зовнішнім ключем, який належить до таблиці `Review`. Цей зв'язок визначається типом один до одного, адже один відгук може бути

підтверджений тільки менеджером тієї компанії-працедавця, про яку він написаний, а кожне підтвердження стосується тільки одного відгуку.

Поле `employermanager` є зовнішнім ключем, який належить до таблиці `Employermanager`. Цей зв'язок визначається типом один до багатьох, адже один менеджер компанії-працедавця може підтвердити різні відгуки про працедавців, а кожне підтвердження виконується одним менеджером компанії-працедавця.

Дані стосовно оголошень про роботу зберігаються в таблиці `Workoffer` з полями:

- `id` – номер оголошення про роботу;
- `workposition` – посада, про вакансію якої створено оголошення;
- `department` – підрозділ компанії-працедавця, вакансія в якому виникла;
- `state` – стан вакансії: підготовлена, відкрита, недостовірна, продовжена, закрита, заархівована;
- `skills` – вміння, які необхідні для роботи за цією вакансією;
- `duties` – обов'язки під час роботи за цією вакансією;
- `conditions` – умови праці за цією вакансією;
- `education` – мінімальний рівень освіти, необхідний для роботи за вакансією;
- `experience` – вимоги до досвіду роботи кандидата;
- `requirements` – усі інші вимоги до кандидата;
- `overall` – загальний повний текст оголошення про роботу;
- `minsalary` – мінімальна заробітна платня за цією вакансією;
- `maxsalary` – максимальна заробітна платня за цією вакансією;
- `offered` – позначка про час створення цього оголошення;
- `updated` – позначка про час останнього оновлення цього оголошення;
- `till` – позначка про час, з якого вакансія вважається закритою.

Серед цих полів є поле `department`, що є зовнішнім ключем, який належить до таблиці `Department`. Цей зв'язок визначається типом один до

багатьох, адже підрозділ компанії-працедавця може мати різні оголошення про вакансії, а одне оголошення про вакансію визначається тільки для конкретного підрозділу однієї компанії-працедавця, тобто не можна визначити одне оголошення для різних підрозділів, потрібно створити окреме оголошення.

Дані стосовно шаблонів оголошень про роботу зберігаються в таблиці WorkofferTemplate з полями:

- id – номер шаблону оголошення про роботу;
- titled_as – назва шаблону оголошення про роботу;
- employer – компанія-працедавець, якій належить цей шаблон;
- skills – вміння за роботою, які визначені в шаблоні оголошення;
- overall – шаблонний текст оголошення;
- duties – обов’язки за роботою, які визначені в шаблоні оголошення;
- conditions – умови праці за роботою, які визначені в шаблоні оголошення;
- education – мінімальний рівень освіти за роботою, які визначені в шаблоні оголошення;
- experience – вимоги до досвіду роботи кандидата за роботою, які визначені в шаблоні оголошення;
- requirements – усі інші вимоги до кандидата за роботою, які визначені в шаблоні оголошення.

Серед цих полів є поле employer, що є зовнішнім ключем, який належить до таблиці Employer. Цей зв’язок визначається типом один до багатьох, адже компанія-працедавець може мати різні шаблони оголошень про роботу, а кожен шаблон належить одній конкретній компанії-працедавцю.

Дані стосовно тегів, віднесених до шаблонів оголошень про роботу, зберігаються в таблиці WorkofferTemplateTags з полями:

- id – номер співвіднесення тега з шаблоном оголошень про роботу;
- tag – тег, який встановлений для шаблону оголошення про роботу;

– `workoffertemplate` – шаблон оголошення про роботу, якому встановлено цей тег.

Створена ця таблиця була для того, щоб реалізувати зв'язок між таблицями `WorkofferTemplate` і `Tag`, який має тип багато до багатьох, адже один шаблон оголошення про роботу може мати різні теги, а кожен тег повинен використовуватися в різних шаблонах оголошень.

Дані стосовно тегів, віднесених до оголошень про роботу, зберігаються в таблиці `WorkofferTags` з полями:

- `id` – номер співвіднесення тега з оголошенням про роботу;
- `tag` – тег, який встановлений для оголошення про роботу;
- `workoffer` – оголошення про роботу, якому встановлено цей тег.

Створена ця таблиця була для того, щоб реалізувати зв'язок між таблицями `Workoffer` і `Tag`, який має тип багато до багатьох, адже одне оголошення про роботу може мати різні теги, а кожен тег повинен використовуватися в різних оголошеннях, оскільки теги необхідні, щоб швидко знаходити відповідним чином угруповані оголошення.

Дані стосовно тегів зберігаються в таблиці `Tag` з полями:

- `id` – номер тега;
- `name` – текст тега.

Дані стосовно запитів на роботу за існуючими оголошеннями зберігаються в таблиці `Workask` з полями:

- `id` – номер запиту на роботу за існуючим оголошенням;
- `employee` – користувач-працівник, який оформив запит на роботу;
- `workoffer` – оголошення про роботу, на яке оформлено запит;
- `overall` – текстовий опис запиту на роботу;
- `resume` – доданий файл резюме;
- `state` – стан запиту;
- `asked` – позначка про час, коли був створений запит.

Поле `employee` є зовнішнім ключем, який належить до таблиці `Employee`. Цей зв'язок визначається типом один до багатьох, адже один

працівник може мати різні запити на роботу, а кожен запит стосується конкретного працівника.

Ще один зовнішній ключ – це поле `workoffer`, що належить до таблиці `Workoffer`. Цей зв'язок також визначається типом один до багатьох, адже одне оголошення про роботу може мати декілька запитів на роботу, а кожен запит оформляється на конкретне оголошення про роботу.

Дані стосовно сфер діяльності компанії-працівника зберігаються в таблиці `EmployerSphere` з полями:

- `id` – номер співвідношення компанії-працівника зі сферою діяльності;
- `employer` – компанія-працівник, яка належить до цієї сфери діяльності;
- `sphere` – сфера діяльності, до якої належить компанія-працівник.

Створена ця таблиця була для того, щоб реалізувати зв'язок між таблицями `Employer` і `Sphere`, який має тип багато до багатьох, адже одна компанія-працедавець може працювати в різних сферах, а кожна сфера повинна бути представлена різними компаніями-працедавцями.

Дані стосовно сфер діяльності, що відносяться до професії, зберігаються в таблиці `ProfessionSphere` з полями:

- `id` – номер співвідношення професії зі сферою діяльності;
- `profession` – професія, яка належить до цієї сфери діяльності;
- `sphere` – сфера діяльності, до якої належить професія.

Створена ця таблиця була для того, щоб реалізувати зв'язок між таблицями `Profession` і `Sphere`, який має тип багато до багатьох, адже одна професія може бути пов'язана з різними сферами, а кожна сфера повинна бути представлена різними професіями.

Схема БД, яка містить ці таблиці та зв'язки між ними, подана на рис. 3.4.

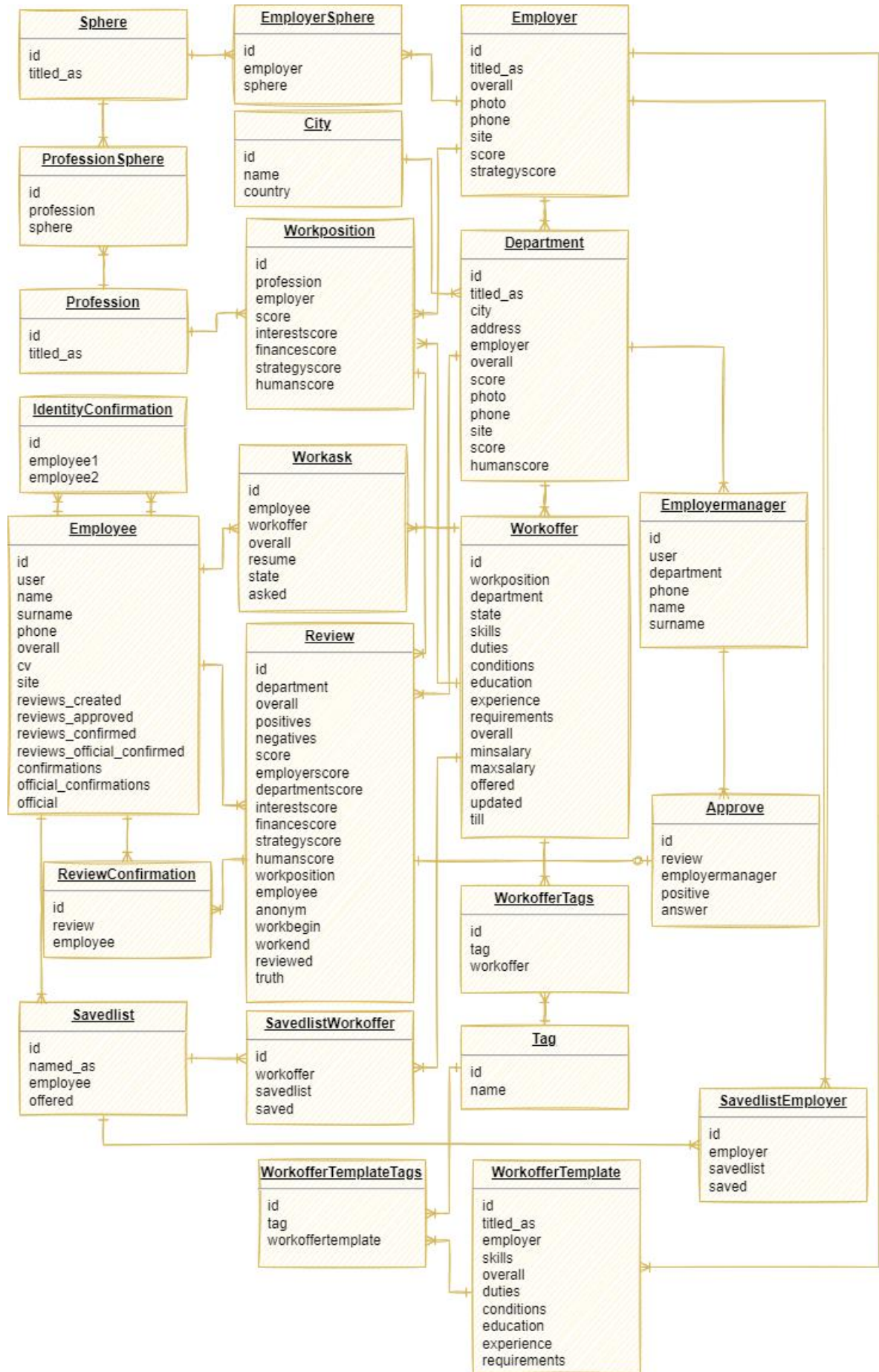


Рисунок 3.4 – Схема БД

3.4 Висновки за розділом 3

Основними вибраними засобами проєктування і розробки є мова Python і пакет scikit-learn, адже необхідно забезпечити роботу алгоритму RF, система для роботи з БД MySQL. Моделювання сценаріїв було реалізовано за діаграмами використання для менеджера підрозділу компанії-працедавця, працівника, гостя. Зважаючи на всеохоплення можливостей внесення, зміни і перегляду даних адміністратором, окрема діаграма для нього не виносилась в результати роботи. Описано повністю структуру кожної таблиці БД і зв'язки між цими таблицями.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Логіка функціонування програми

Фактично функціональність програми застосовується для того, щоб користувач, який шукає роботу, знайшов необхідну йому вакансію, при цьому отримавши потрібну для прийняття рішення інформацію про те, наскільки придатною для нього є ця посада, цей працедавець. Тобто сама по собі робота з оголошенням про роботу, створеними працедавцями, та інша частина роботи, на яку направлені особливості даної програми, безпосередньо пов'язані з об'єктом оголошення про роботу. А тому порядок роботи зі станами цього об'єкта визначає фактично і ту логіку, яка закладена в порядок функціонування програми і всіх її складових.

Робота над будь-яким оголошенням починається з того, що менеджер компанії-працедавця його створює (рис. 4.1). Лише після того, як менеджер відкриває цю вакансію, тобто розпочинається конкурс, оголошення стає відкритим, тоді воно фактично публікується і його бачать всі інші користувачі.

Відкрите оголошення може перебувати в такому стані до того моменту, поки строк, до якого проводиться конкурс, не вийде. Перевірка цього моменту відбувається шляхом запуску скрипта, який шляхом взаємодії з БД змінює стан оголошення, якщо строк вийшов. Після того, як строк вийшов, оголошення стає закритим. Якщо ж оголошення знаходиться у стані відкритого, але менеджер компанії-працедавця збільшив строк завершення конкурсу, то оголошення переходить до стану продовженого. Якщо строк був зменшений, то стан не змінюється.

Після переходу оголошення до стану закритого, якщо не пройшов місяць з дня завершення строку конкурсу, менеджер компанії-працедавця може продовжити строк конкурсу і таким чином перевести оголошення до стану продовженого.

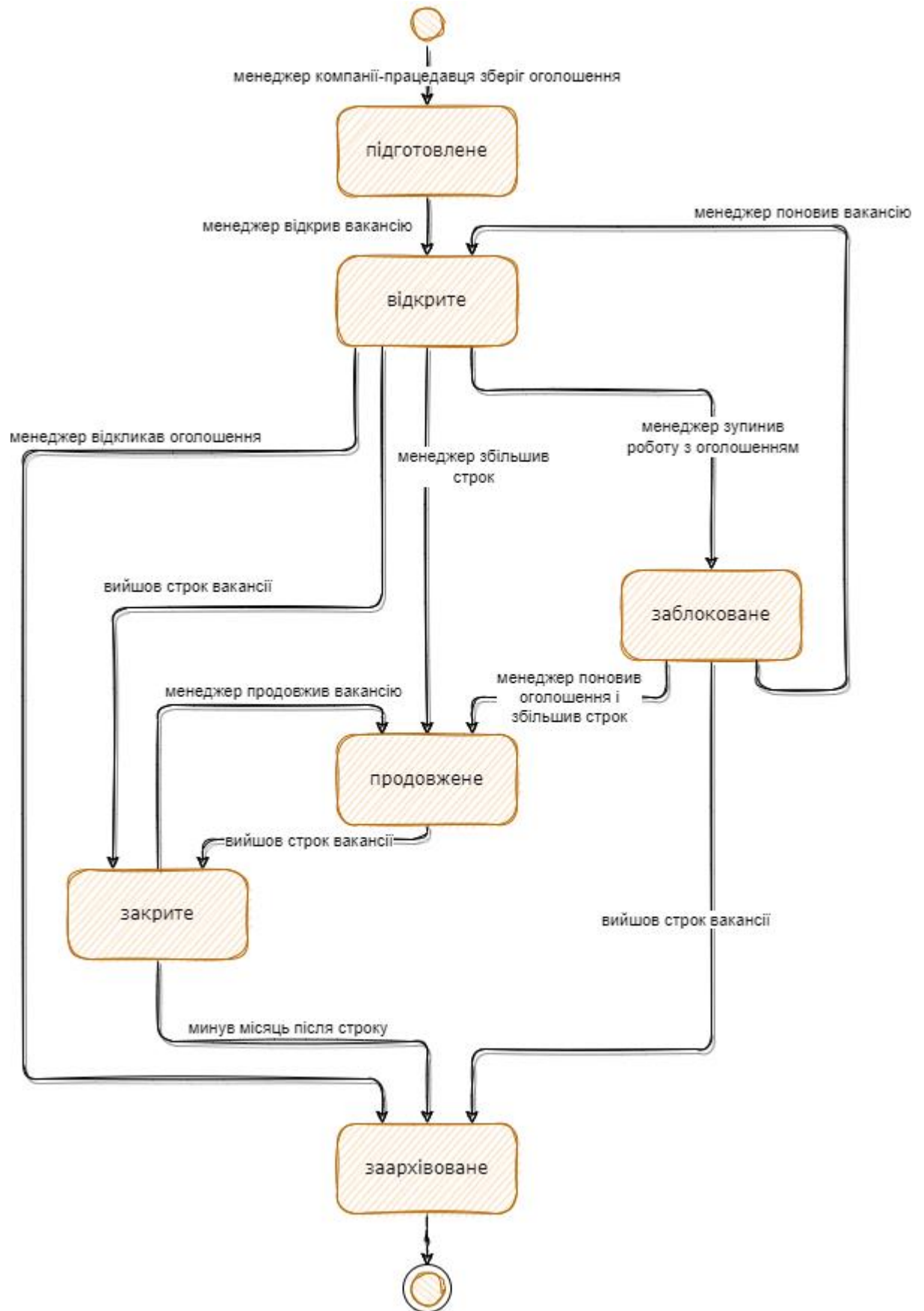


Рисунок 4.1 – Діаграма станів оголошення про роботу

Якщо з дати завершення конкурсу минув місяць, то оголошення автоматично переводиться до стану заархівованого і більше не зможе змінюватися ніяким чином.

Відкрите оголошення в будь-який момент менеджер може відкликати і таким чином перевести його до стану заархівованого. Але також він може закрити до нього доступ, щоб через певний час до нього повернутися. У такому випадку він переводить оголошення до стану заблокованого, а з цього стану тоді в залежності від встановлених строків може або повернути його до відкритого стану, або до продовженого.

4.2 Класи вебдодатку

Оскільки однією з основних сутностей роботи є оголошення про роботу, то опис розпочато з відповідних класів, які стосуються цієї сутності. Базовим класом є клас моделі `Workoffer`, але він фактично повторює структуру таблиці БД, яку було описано в попередньому розділі. Вебсторінки для роботи з оголошеннями формуються на основі шаблонів, а ось дані для заповнення цих шаблонів готуються класами подань.

Сама публікація оголошення керується класом подання `WorkofferPublication`, який підтримується на основі відповідної форми `FormWorkoffer`. Це форма, яка за допомогою відповідних полів дозволяє заповнювати всю необхідну інформацію в оголошенні про роботу, яку надає працедавець через одного зі своїх менеджерів.

Клас має `get`-метод, в якому реалізується зокрема підтримка форми через наповнення шаблонів і посад, які можуть використовуватися. Це зроблено в даному методі, а не в формі, оскільки набір цих об'єктів залежить від того, менеджер якого саме працедавця зайшов у систему. В залежності від цього вибираються шаблони, які належать цій компанії-працедавцю, та посади, які вже визначені в цій компанії-працедавці. Шаблони створені за допомогою шаблону [16].

Також клас має `post`-метод, в якому відбувається перевірка валідності заповнення форми і можливості збереження об'єкта на основі цих даних. Якщо форма невалідна, то відбувається повернення до форми заново. Якщо ж форма валідна, то програмно вноситься номер підрозділу, до якого належить менеджер компанії-працівника, що увійшов у систему. Цей підрозділ вноситься як підрозділ, за яким оголошено вакансію в оголошенні. Також визначається стан оголошення як підготовлений і зберігаються всі теги, які було пов'язано з цим оголошенням на основі зв'язку багато до багатьох. У разі успішності всіх цих дій, після цього завантажується сторінка з деталізованим представленням цього оголошення, тобто надається можливість переглянути в готовому до публікації вигляді оголошення.

Змінювання оголошення керується через клас подання `WorkofferModification`. Базовою формою для цього подання і відповідного шаблону сторінки є та ж сама форма, що визначена і для публікації оголошення. Також тією самою є і вебсторінка, тобто вона створена на основі того самого шаблону.

Клас має `get`-метод, в якому реалізується зокрема підтримка початкового заповнення форми на основі даних об'єкта, змінення якого виконується. Під час цього заповнення перевіряється, чи належить авторизований менеджер компанії-працедавця до того ж самого підрозділу, з яким асоційовано оголошення. Тільки у тому випадку, якщо це так, виконується наповнення шаблонів і посад, які можуть використовуватися для цієї компанії-працедавця, і відповідний контекст даних передається для заповнення шаблону сторінки. Якщо ж ні, то повертається помилка. Так забезпечено захист від того, щоб користувач, який не має відповідних прав, виконав зміну оголошення. При цьому звичайно менеджери компанії-працедавця мають слідкувати за доступом всіх своїх колег, щоб не надати відповідні права користувачу, який таких прав не повинен мати. Ці дії також реалізовано для менеджерів через класи подань демонстрування всіх менеджерів підрозділу компанії-працедавця (підрозділ авторизованого

менеджера) `DepartmentEmployerManagers`, реєстрації менеджерів компанії-працедавця `EmployerManagerRegister`. Через останній клас подання може бути скасовано реєстрацію менеджера в компанії або навпаки доданого користувача, який до цього зареєструвався як працівник в ролі менеджера.

Клас має `post`-метод, в якому спочатку відбувається перевірка того, чи знаходиться оголошення в стані, для якого можна вносити зміни. Якщо оголошення знаходиться в стані підготовленого, відкритого, продовженого або закритого, то робота над змінами допускається і дані завантажуються з форми зі збереженням дати завершення конкурсу, яка була внесена в оголошення раніше. Якщо це інший стан, то повертається помилка. Після цього (у випадку правильних станів) відбувається перевірка валідності заповнення форми і можливості збереження об'єкта на основі цих даних. Якщо форма невалідна, то відбувається повернення до форми заново. Якщо ж форма валідна, то разом зі збереженням змінених даних відбувається змінення стану оголошення. Якщо було збільшено дату завершення конкурсу, а оголошення знаходиться в стані відкритого, то воно переводиться до стану продовженого. Якщо оголошення знаходиться в стані закритого і дата завершення конкурсу збільшується, то воно також переводиться до стану продовженого. Також відбувається декілька перевірок, щоб стан не було штучно спотворено. У разі успішності всіх цих дій, після цього завантажується сторінка з деталізованим представленням цього оголошення, тобто надається можливість переглянути в готовому до публікації вигляді оголошення.

Окремо клас подання `WorkofferCancel` керує скасуванням оголошення, яке є відкритим на даний момент, але менеджер цієї компанії-працедавця бажає його заархівувати.

Клас подання `WorkofferDetail` дозволяє переглянути повну інформацію стосовно оголошення про роботу. Для цього окрім самого оголошення витягається множина тегів, асоційованих з цим оголошенням.

Клас подання `ManagerWorkoffers` керує демонстрацією всіх оголошень, які доступні менеджеру компанії-працедавця. Визначається підрозділ, до якого належить авторизований користувач і за цим підрозділом витягаються всі оголошення.

Клас подання `DepartmentWorkoffers` керує демонстрацією всіх оголошень, які доступні за конкретно визначеним через параметр `get`-запиту номером підрозділу. За цим підрозділом витягаються всі оголошення, впорядковуючи від останніх, але витягаючи тільки ті оголошення, які є відкритими або продовженими, адже це подання для працівника, а не для менеджера як попереднє. Аналогічно клас подання `EmployerWorkoffers` виконує те саме наповнення, але обираючи оголошення, що належать не заданому підрозділу компанії, а всій компанії одразу.

Спеціалізований клас подання `EmployerDepWorkoffers` також керує списком оголошень за компанією-працедавцем, але об'єднує їх за посадами. На сторінці можна тоді відобразити посади, а за кожною посадою показати, в яких саме підрозділах вона доступна у вигляді вакансії.

Сам пошук оголошень підтримується класом подання `FindOffers`. Встановлення значень параметрів пошуку виконується через параметри `get`-запиту: параметр `prof` визначає номер професії, за якою потрібно виконати пошук (вибираються оголошення, що стосуються тільки цієї професії, якщо вона задана), параметр `salary` визначає мінімальну заробітну платню, яка має бути не більшою ніж мінімальна заробітна платня, вказана в оголошенні, параметри `minasks` і `maxasks` визначають відповідно мінімальну і максимальну кількість запитів на роботу за оголошенням (щоб працівник міг відсіяти оголошення, які не привернули нічию увагу, і оголошення, які надто популярні), параметр `minscore` визначає мінімальну усереднену оцінку, яку повинна мати посада, за якою створено оголошення, параметр `miniscore` визначає мінімальну усереднену оцінку, яку повинна мати посада, за якою створено оголошення, відносно того, на скільки вона є цікавою, параметр `minfscore` визначає мінімальну усереднену оцінку, яку повинна мати посада,

за якою створено оголошення, відносно того, на скільки вона є фінансово привабливою, параметр `minsscore` визначає мінімальну усереднену оцінку, яку повинна мати посада, за якою створено оголошення, відносно того, на скільки ця компанія має правильну стратегію, параметр `minhscore` визначає мінімальну усереднену оцінку, яку повинна мати посада, за якою створено оголошення, відносно того, на скільки привабливим є колектив (людські якості), параметр `city` визначає місто (за його номером), в якому повинен розташовуватися підрозділ компанії-працедавця (якщо значення задано), параметр `tags` визначає теги, які повинні бути встановлені для оголошення, параметр `spheres` визначає сфери діяльності, до яких має відноситися компанія-працедавець (останні два параметри можуть мати набір значень, перелічених через кому).

Клас `WorkofferAsks` керує наповненням запитами на роботу за заданим оголошенням про роботу. Номер оголошення передається через параметр `off` get-запиту. Перевіряється також, що підрозділ, з яким асоційовано авторизованого менеджера компанії-працедавця, співпадає з підрозділом, для якого опублікована вакансія в цьому оголошенні. Спочатку в списку повертаються запити, які були подані останніми.

Клас подання `LastCityWorkoffers` керує наповненням сторінки відтворення останніх оголошень про роботу в заданому місті. Місто передається своїм номером через параметр `city` get-запиту. Допускаються тільки оголошення, які знаходяться в стані відкритих або продовжених, які асоціюються з підрозділами компаній, розташованими в заданому місті. Надається 1000 останніх оголошень за часом публікації.

Клас подання `LastTagWorkoffers` керує наповненням сторінки відтворення останніх оголошень про роботу, що відмічені заданим тегом. Тег задається своєю назвою. Місто передається своїм номером через параметр `city` get-запиту. Допускаються тільки оголошення, які знаходяться в стані відкритих або продовжених. Надається 1000 останніх оголошень за часом публікації.

Клас подання LastSphereWorkoffers керує наповненням сторінки відтворення останніх оголошень про роботу, що визначені за вакансіями, які стосуються професій в заданій сфері діяльності. Сфера діяльності задається своїм номером через параметр sphere get-запиту. Допускаються тільки оголошення, які знаходяться в стані відкритих або продовжених. Надається 1000 останніх оголошень за часом публікації.

4.3 Висновки за розділом 4

Повністю реалізовано програмне забезпечення у відповідності зі сценаріями роботи користувачів всіх ролей. Оскільки така робота вимагала координації складових програмного забезпечення між собою, було визначено можливі стани оголошення і переходи між цими станами.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма повинна надавати можливість працедавцям створювати оголошення про роботу, працівникам – інформацію про свої оцінки компаній, на які вони працюють, а користувачам, які знаходяться в пошуку роботи або вивчають ринок праці, – отримувати інформацію про потенційних працедавців і знаходити роботу, яка може відповідати їх вимогам з більшою вірогідністю.

5.2 Умови виконання програми

Сервер повинен забезпечувати таку обчислювальну потужність: на сервері має бути забезпечено щонайменше процесор, який має не менше ніж 4 ядра, частоту від 2 ГГц, оперативна пам'ять має бути не меншою від 4 Гб, на жорсткому диску має бути доступно не менше ніж 2 Гб.

5.3 Виконання програми

Під час роботи з вебдодатком необхідно враховувати, що за замовчуванням відображаються сторінки в режимі гостя, але після авторизації режим змінюється. Гість має обмежені функції працівника. Під час авторизації автоматично завантажується сторінка авторизації працівника, але з неї можна перейти до сторінки авторизації менеджера.

Менеджер підрозділу компанії-працедавця має об'єднані функції у основні розділи меню, до яких входять «Компанія», з якого можна підійти до вкладених підрозділів «Інформація» для перегляду всієї інформації про компанію з можливістю змінювання, «Підрозділ» для виконання аналогічних дій за самим підрозділом, «Посади» для демонстрування, додавання і

змінювання посад в підрозділі компанії, «Менеджери» для керування доступом інших менеджерів до роботи з підрозділом компанії.

Розділ «Рейтинг» дозволяє обрати варіант рейтингу (за професію або за сферою діяльності) і відобразити відповідні найкращі компанії за введеною професією або сферою діяльності.

Розділ «Відгуки» дозволяє одразу переглянути всі відгуки про компанію-працедавця, менеджером якої є авторизований користувач. Звідси можна перейти до підтвердження відгуків, під час якого можна зокрема надати розширену відповідь, відреагувавши на вказані проблеми. Це дозволяє у випадку аргументованої відповіді знизити негативний вплив від відгуку.

Розділ «Оголошення» містить набір вкладених підрозділів, включаючи «Вакансії», де міститься сам список оголошень про роботу, які вже було створено за цим підрозділом компанії-працедавця, «Запити», де містяться запити за оголошеннями, які були подані працівниками, «Шаблони», де містяться всі шаблони, які можуть використовуватися під час публікації оголошень, в компанії. З підрозділу «Вакансії» можна перейти до сторінки зі всіма опублікованими вакансіями, а звідти через кнопку «Створити» перейти до сторінки з внесенням даних оголошення (рис. 5.1). При цьому це сторінка тільки підготовки оголошення. Для того щоб оголошення з'явилося у працівників, потрібно після цього зайти в цей розділ в знайшовши серед оголошень потрібне, ще не опубліковане, натиснути біля нього на кнопку «Опублікувати», яка в такому випадку буде відображатися.

На сторінці створення оголошення можна використати один з шаблонів з випадаючого списку, а після цього натиснути на кнопку «Використати», наявні з шаблону позиції будуть автоматично задані в полях форми оголошення. На самій формі можна обрати з випадаючого списку посаду, яка є в цьому підрозділі компанії (або за кнопкою «Додати» одразу перейти до її внесення), вибрати рівень освіти, задати вміння, обов'язки, досвід роботи, заробітну платню, текст оголошення і дату завершення.

Empviewer

Компанія ▾РейтингВідгукиОголошення ▾Про себе

Вийти

Підготовка оголошення

Оголошення ○ Створення

Оберіть шаблон



Мінімальний

Використати

Посада

Вебдизайнер

Додати

Вміння

Обов'язки

Освіта

Вища освіта

Обов'язки

Досвід роботи

Інші вимоги

Мінімальна зарплатня

0

Максимальна зарплатня

0

Текст оголошення

Завершення конкурсу

дд.мм.рррр

Зберегти

Рисунок 5.1 – Вебсторінка підготовки оголошення про роботу

Працівники мають розділи меню, до яких входять «Компанії» (з підрозділами «Рейтинг» та «Пошук» для виконання пошуку потрібної), «Робота» (з підрозділами «Останні» для відображення останніх опублікованих оголошень за обраним містом, тегом або сферою діяльності та «Пошук» для виконання пошуку оголошень про роботу), «Запити» для перегляду списку власних запитів на оголошення та реагування на ці запити від працедавців, «Відгуки» для перегляду власних створених відгуків.

Розділ «Списки» дозволяє перейти до списків працедавців або оголошень, після чого відображається набір таких списків. Обравши потрібний, можна перейти до списку відповідних компаній або оголошень. А зі сторінки списку компаній є кнопка для відображення оголошень за цими компаніями (рис. 5.2).

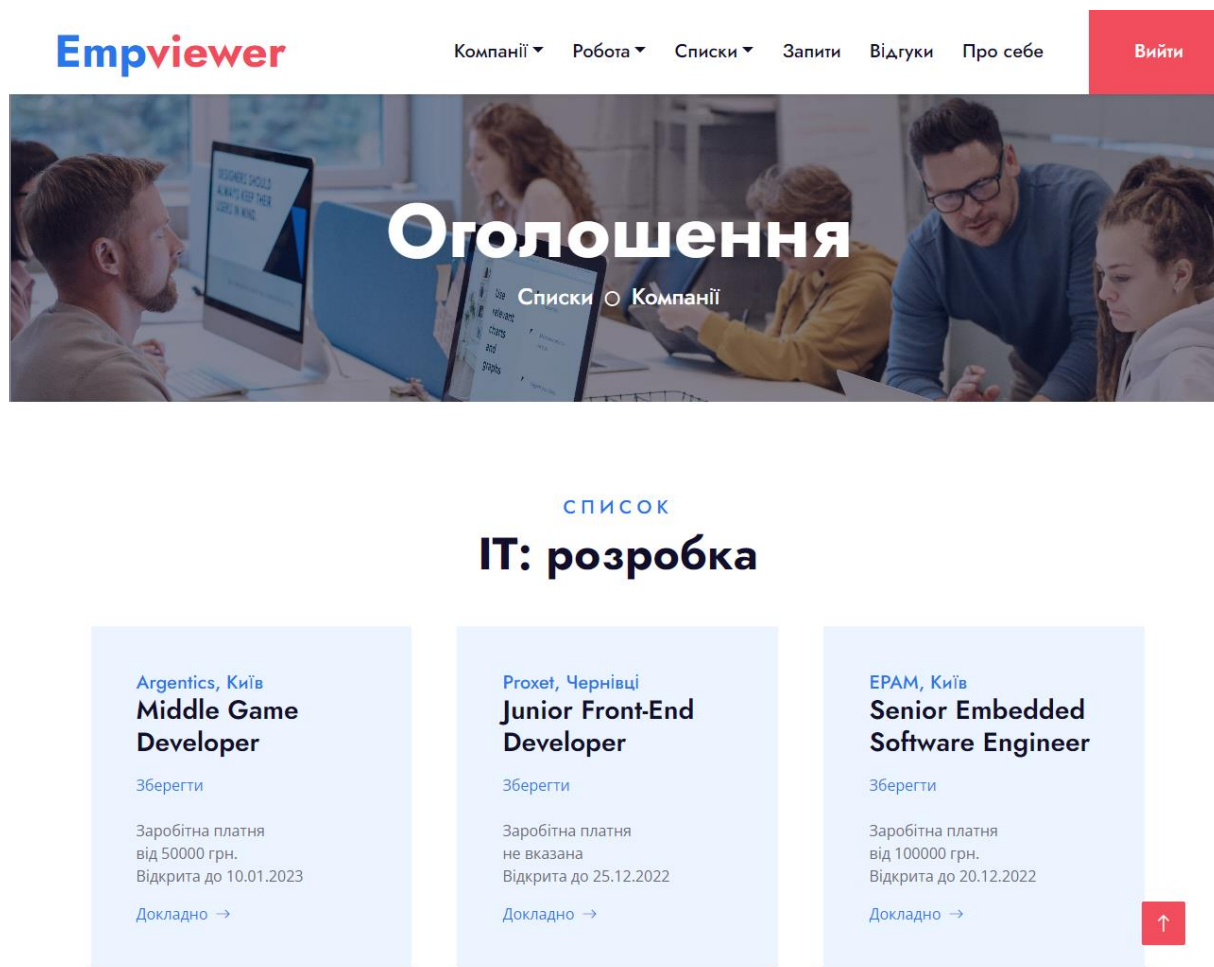


Рисунок 5.2 – Вебсторінка демонстрації оголошень про роботу в компаніях-працедавцях зі збереженого списку

За кожним оголошенням доступне посилання для збереження оголошення в списку. На наступній сторінці потрібно буде обрати необхідний список зі всіх доступних у працівника. Можна також перейти до сторінки самої компанії, де окрім інформації про компанію міститься короткий набір за підрозділами, включаючи відгуки (рис. 5.3), посади, оголошення та підрозділи. За відповідними кнопками можна перейти до сторінок повного перегляду відповідних підрозділів.

За відгуками міститься їх список, звідки за кожним можна перейти до окремої сторінки з повною інформацією, включаючи час роботи, якщо він був вказаний, оцінки за окремими критеріями. Для кожного класифікованого як недостовірний відгук відображається напис «недостовірний».

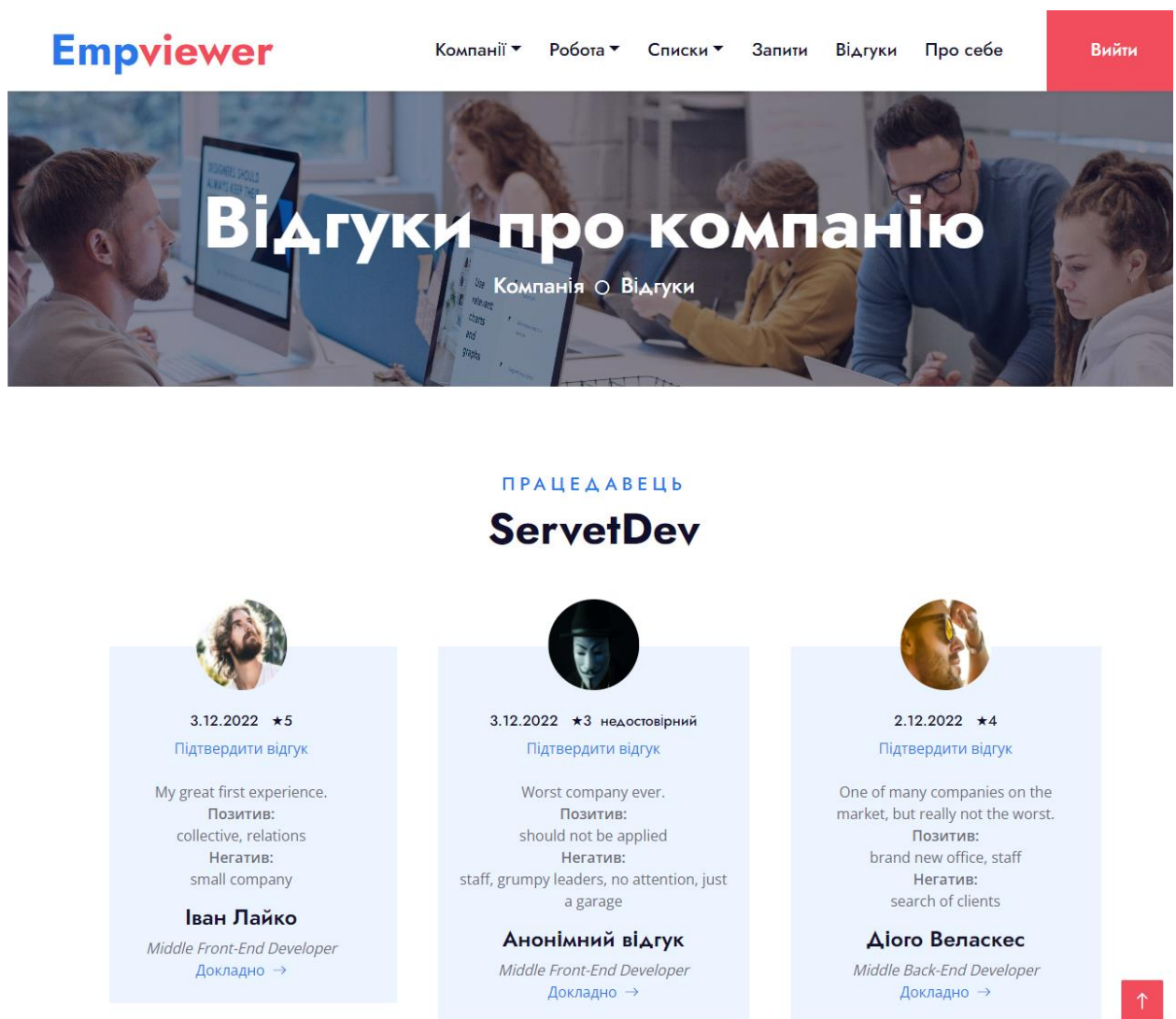


Рисунок 5.3 – Демонстрація відгуків про компанію-працедавця

5.4 Експериментальне дослідження

Для дослідження ефективності застосування методу визначення достовірності відгуків про працедавців було виконано експериментальне дослідження, де даний метод було порівняно з наївним байєсовим класифікатором за точністю класифікації. Для дослідження використано вибірку [17].

Вибірка містить велику кількість атрибутів, включаючи номер компанії, номер відгуку, теги опису роботи, підрозділ, загальний рейтинг (оцінку) компанії, кількість переглядів, логін автора відгуку (велика частина відгуків анонімна), поточна робота автора, номер користувача-автора, дата публікації відгуку, описові мітки роботи, місто, де виконується робота, посада, тип, назва компанії, кількість працівників у компанії, галузь, в якій працює компанія, оцінка компанії автором загалом і за критеріями (кар'єрний зріст, фінансова сторона, баланс роботи і життя, задоволеність роботою, культура компанії, розширення навичок, безпека роботи), місце компанії в різного роду рейтингах, текст відгуку та інші атрибути. Загалом вибірка містить більше ніж 487 тисяч відгуків.

Метод визначення достовірності відгуків про працедавців призвів до точності класифікації на тестовій вибірці в 82,87 % (табл. 5.1), що в 1,6 рази більше ніж результат наївного байєсового класифікатора.

Таблиця 5.1 – Результати експериментального дослідження

Метод класифікації	Точність, %
Наївний байєсів класифікатор	51,61
Метод визначення достовірності відгуків про працедавців	82,87

5.5 Висновки за розділом 5

Проведене дослідження разом з тестуванням програми підтвердили реалізацію всіх функцій програми, які було заплановано, у повній мірі та високу точність класифікації відгуків про працедавців щодо їх достовірності на основі використання вибірки даних.

ВИСНОВКИ

Виконання дипломної кваліфікаційної роботи магістра направлено на розробку програмного забезпечення надання доступу до відгуків про працедавців для підтримки шукачів роботи під час вибору працедавців.

Результати роботи охоплюють програмне забезпечення, що підтримує як пошук роботи через доступ до оголошень про вакансії, подання запитів на роботу, керування цими запитами, так і отримання інформації про працедавців через відгуки, а зокрема і перехід від оголошення до відгуків за цією посадою.

Однією з відмінностей роботи є визначення достовірності відгуків, що дозволяє шукачу роботи розуміти, на скільки варто довіряти відомостям з відгуку. При цьому забезпечується це різними інструментами: програмно результатами використання інтелектуального методу, підтвердженням відгуків іншими користувачами, для яких передбачено також можливість офіційного підтвердження особи, підтвердженням або реагуванням на відгуки зі сторони самих працедавців.

Відгуки про працедавців визначають посаду, на якій працював або працює працівник, та містять цілий ряд інформації, що включає сам текст відгуку, конкретний підрозділ компанії-працедавця, де працював працівник, посаду, позитивні й негативні сторони виконуваної роботи, загальну оцінку в балах виконуваних роботах, оцінку компанії-працедавцю, оцінку підрозділу, а також оцінки за окремими показниками: оцінка відносно того, на скільки робота є цікавою, оцінка відносно того, на скільки робота є фінансово привабливою, оцінка роботи відносно того, на скільки стратегія компанії є правильною, оцінка роботи відносно того, на скільки привабливим є колектив. Відгук може бути поданий як анонімний, а може бути з зазначенням авторства.

Науковою новизною роботи є метод визначення достовірності відгуків про працедавців, який дозволяє класифікувати відгуки про

працедавців щодо їх достовірності або недостовірності, виконуючи векторизацію тексту відгуку за допомогою моделі TF-IDF і використовуючи алгоритм RF для реалізації класифікації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Finding Meaning in Contradiction on Employee Review Sites – Effects of Discrepant Online Reviews on Job Application Intentions [Text] / Raoul Könsgen, Mario Schaarschmidt, Stefan Ivens, Andreas Munzel // Journal of Interactive Marketing. – 2018. – Vol. 43 (1). – Pp. 165-177.
2. Work.ua – сайт пошуку роботи №1 в Україні [Електрон. ресурс]. – Режим доступу : <https://www.work.ua/>.
3. Glassdoor Job Search. You deserve a job that loves you back [Electronic resource]. – Access mode : <https://www.glassdoor.com/index.htm>.
4. Anderson Construction Reviews & Ratings by 7 Employees. CareerBliss [Electronic resource]. – Access mode : <https://www.careerbliss.com/anderson-construction/reviews/>.
5. Пошук вакансій Indeed [Електрон. ресурс]. – Режим доступу : <https://ua.indeed.com/?from=gnav-homepage>.
6. A guide to Natural Language Processing (NLP) in 2022 [Electronic resource] : What it is, how it works, benefits, and limitations. – Access mode : <https://www.ai.nl/knowledge-base/natural-language-processing/>.
7. What is natural language processing (NLP) [Electronic resource] : Techniques and use cases. – Access mode : <https://www.superannotate.com/blog/what-is-natural-language-processing>.
8. Bag-of-Words(BOW) – Towards AI [Electronic resource]. – Access mode : <https://towardsai.net/p/l/bag-of-wordsbow>.
9. Roepke, B. A Quick Introduction to Bag of Words and TF-IDF [Electronic resource] / B. Roepke. – Access mode : <https://www.dataknowsall.com/bowtfidf.html>.
10. Chouinard, J.-C. TF-IDF [Electronic resource] : Term frequency-inverse document frequency / J.-C. Chouinard. – Access mode : <https://www.jcchouinard.com/tf-idf/>.
11. Dutta, B. What is Random Forest Classifiers in Machine Learning?

[Electronic resource] / B. Dutta. – Access mode : <https://www.analyticssteps.com/blogs/what-random-forest-classifiers-machine-learning>.

12. Talari, S. Random Forest vs Decision Tree [Electronic resource] : Key Differences / S. Talari. – Access mode : <https://www.kdnuggets.com/2022/02/random-forest-decision-tree-key-differences.html>.

13. What is a Random Forest? [Electronic resource]. – Access mode : <https://databasecamp.de/en/ml/random-forests>.

14. Donges, N. Random Forest Classifier [Electronic resource] : A Complete Guide to How It Works in Machine Learning / N. Donges. – Access mode : <https://builtin.com/data-science/random-forest-algorithm>.

15. A novel improved random forest for text classification using feature ranking and optimal number of trees [Text] / Nasir Jalal, Arif Mehmood, Gyu Sang Choi, Imran Ashraf // Journal of King Saud University – Computer and Information Sciences. – 2022. – Volume 34, Issue 6, Part A. – Pp. 2733-2742.

16. SafeCam – CCTV Website Template [Electronic resource]. – Access mode : <https://htmlcodex.com/cctv-website-template/>.

17. Fake Company Reviews [Electronic resource]. – Access mode : https://www.kaggle.com/datasets/sachinsk/fake-company-reviews?select=Test_FakeReviewsClassifier.csv

ДОДАТОК А
Технічне завдання

Вступ

Ринок праці є двостороннім майданчиком, на якому працедавці бажають знайти потрібних їм працівників, а працівники – знайти такого працедавця, який би повністю задовольнив їх вимоги. Враховуючи останнє, людина, працюючи в поточний момент, може знаходитися в пошуку роботи своєї мрії. І з цього боку працівники потребують інформації про те, наскільки робота в компанії є достатньо цікавою, приносить відповідне задоволення, з точки зору реальних працівників цієї компанії. Таку програму і необхідно створити в цій роботі, але беручи до уваги, що представлена інформація може спотворити уявлення потенційних працівників без перевірки її достовірності.

А.1 Підстави для розробки

Розробка кваліфікаційної роботи магістра виконується на основі завдання за темою «Дослідження та програмна реалізація методів визначення достовірності відгуків про працедавців», затвердженою наказом № 363 від 4 листопада 2022 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма повинна надавати можливість працедавцям створювати оголошення про роботу, працівникам – інформацію про свої оцінки компаній, на які вони працюють, а користувачам, які знаходяться в пошуку роботи або вивчають ринок праці, – отримувати інформацію про потенційних працедавців і знаходити роботу, яка може відповідати їх вимогам з більшою вірогідністю.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечити користувачів наданням таких функціональних можливостей:

- публікація відгуків про працедавців;
- пошук працедавців;
- пошук оголошень про роботу;
- демонстрування та змінення інформації про компанію-працедавця;
- визначення менеджерів для представлення компаній-працедавців;
- демонстрування та змінення інформації про підрозділ компанії-працедавця;
- демонстрування відгуків про працедавця (загалом, у розрізі посади або у розрізі підрозділу);
- демонстрування останніх оголошень про роботу за містом, за тегами, за сферою;
- формування рейтингу компаній-працедавців за професією, за сферою;
- визначення і змінення посади компанії-працедавця;
- демонстрування всіх посад компанії-працедавця;
- демонстрування всіх вакансій компанії-працедавця;
- демонстрування підрозділів компанії-працедавця;
- створення списку збережених компаній-працедавців або оголошень про роботу;
- додавання або вилучення компаній-працедавців або оголошень зі збереженого списку;
- демонстрування збереженого списку;
- демонстрування збережених списків користувачем;
- демонстрування оголошень про роботу компаній-працедавців зі збереженого списку;

– демонстрування відгуків працівника і демонстрування його власних відгуків;

- підтвердження особистості працівника;
- підтвердження відгуку працівником або компанією-працедавцем;
- подання запиту на роботу за оголошенням;
- підтвердження або скасування запиту на роботу;
- визначення достовірності відгуку про працедавця;
- публікація оголошення про роботу;
- публікація оголошення про роботу на основі шаблону;
- змінення оголошення про роботу;
- архівування оголошення про роботу;
- визначення шаблону оголошення про роботу;
- змінення шаблону оголошення про роботу;
- демонстрування власних оголошень про роботу компанії-працедавця;
- подовження дії оголошення про роботу;
- відкликання оголошення про роботу;
- демонстрування власних запитів на роботу;
- демонстрування запитів на оголошення про роботу;
- авторизація і реєстрація записів користувачів;
- офіційне підтвердження особистості працівника.

A.3.2 Вимоги до складу та параметрів технічних засобів

Основні вимоги до технічних засобів формують вимоги до серверної частини.

Сервер повинен забезпечувати і роботу вебдодатка, і роботу інтелектуальної частини, що реалізує запропонований метод. Це потребує відповідної обчислювальної потужності: на сервері має бути забезпечено зонайменше процесор, який має не менше ніж 4 ядра, частоту від 2 ГГц,

оперативна пам'ять має бути не меншою від 4 Гб, на жорсткому диску має бути доступно не менше ніж 2 Гб.

A.3.3 Умови експлуатації

Експлуатація програми передбачає розгортання вебсервера, в ролі якого для вебдодатка може використовуватися uWSGI, Apache зокрема. Для керування БД має застосовуватися MySQL.

A.3.4 Вимоги до програмної документації

Основна програмна документація включає безпосередньо це технічне завдання, текст програми, що є додатками до пояснювальної записки.

A.4 Порядок контролю та приймання

Порядок контролю та приймання роботи визначається календарним планом, який входить до складу завдання на кваліфікаційну роботу магістра.

ДОДАТОК Б
Текст програми

Б.1 Текст файла workofferviews.py

```

from empviewer.forms import FormWorkoffer, FormFindOffers,
FormWorkofferTemplate, FormWorkask
from empviewer.models import Workoffer, Employermanager, Workposition,
WorkofferTemplate, City, Profession, Workask, Savedlist, SavedlistEmployer
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView
from django.shortcuts import render, redirect, get_object_or_404
from django.http import Http404
from django.urls import reverse_lazy

class WorkofferPublication(LoginRequiredMixin, View):
    template_name = 'empviewer/offer_work.html'

    def get(self, request, pk = None):
        wo_form = FormWorkoffer()
        manager_user = self.request.user
        emp_manager = Employermanager.objects.get(user = manager_user)
        department = emp_manager.department
        manager_employer = department.employer
        templates = WorkofferTemplate.objects.filter(employer =
manager_employer)
        workpositions = Workposition.objects.filter(employer =
manager_employer)
        ready_data = {'publication_form': wo_form,
                      'templates': templates,
                      'workpositions': workpositions }
        return render(request, self.template_name, ready_data)

    def post(self, request, pk = None):
        wo_form = FormWorkoffer(request.POST, None)
        manager_user = self.request.user
        emp_manager = Employermanager.objects.get(user = manager_user)
        department = emp_manager.department
        if not wo_form.is_valid():
            manager_employer = department.employer
            templates = WorkofferTemplate.objects.filter(employer =
manager_employer)

```

```

        workpositions = Workposition.objects.filter(employer =
manager_employer)
        ready_data = {'publication_form': wo_form,
            'templates': templates,
            'workpositions': workpositions }
        return render(request, self.template_name, ready_data)
workoffer = FormWorkoffer.save(commit = False)
workoffer.department = department
workoffer.state = 0
workoffer.save()
workoffer.save_m2m()
return redirect(reverse('empviewer:wodetail', args=[workoffer.id]))

```

```

class WorkofferModification(LoginRequiredMixin, View):
    template_name = 'empviewer/offer_work.html'

    def get(self, request, pk):
        workoffer = get_object_or_404(Workoffer, id = pk)
        wo_form = FormWorkoffer (instance = workoffer)
        manager_user = self.request.user
        emp_manager = Employermanager.objects.get(user = manager_user)
        department = emp_manager.department
        if department != workoffer.department:
            raise Http404
        manager_employer = department.employer
        templates = WorkofferTemplate.objects.filter(employer =
manager_employer)
        workpositions = Workposition.objects.filter(employer =
manager_employer)
        ready_data = {'publication_form': wo_form,
            'templates': templates,
            'workpositions': workpositions }
        return render(request, self.template_name, ready_data)

    def post(self, request, pk = None):
        workoffer = get_object_or_404(Workoffer, id = pk)
        if workoffer.state == 0 or workoffer.state == 1 or workoffer.state == 3 or
workoffer.state == 4:
            wo_form = FormWorkoffer(request.POST, None, instance = workoffer)
            oldtill = workoffer.till
        else:

```

```

        raise Http404
    manager_user = self.request.user
    emp_manager = Employermanager.objects.get(user = manager_user)
    department = emp_manager.department
    if not wo_form.is_valid():
        manager_employer = department.employer
        templates = WorkofferTemplate.objects.filter(employer =
manager_employer)
        workpositions = Workposition.objects.filter(employer =
manager_employer)
        ready_data = {'publication_form': wo_form,
                      'templates': templates,
                      'workpositions': workpositions }
        return render(request, self.template_name, ready_data)
    workoffer2 = FormWorkoffer.save(commit = False)
    if workoffer.state == 1 and workoffer.till > oldtill:
        workoffer2.state = 3
    elif workoffer.state == 4 and workoffer.till > oldtill:
        workoffer2.state = 3
    elif workoffer.state == 0 and workoffer2.state == 1:
        workoffer2.state = 1
    elif workoffer.state == 1 and workoffer.till <= oldtill:
        workoffer2.state = 1
    elif workoffer.state == 2:
        workoffer2.state = 2
    elif workoffer.state == 3 and workoffer.till > oldtill:
        workoffer2.state = 3
    elif workoffer.state == 3 and workoffer2.state == 4:
        workoffer2.state = 4
    elif workoffer.state == 2 and workoffer.till > oldtill:
        workoffer2.state = 3
    elif workoffer.state == 2 and workoffer2.state == 1:
        workoffer2.state = 1
    else:
        workoffer2.state = 0
    workoffer2.save()
    workoffer2.save_m2m()
    return redirect(reverse('empviewer:wodetail', args=[workoffer.id]))

```

```

class WorkofferDetail(DetailView):
    model = Workoffer

```

```
template_name = 'empviewer/workoffer.html'
```

```
def get(self, request, pk):
    workoffer = Workoffer.objects.prefetch_related('tags').get(id = pk)
    ready_data = {'workoffer': workoffer}
    return render(request, self.template_name, ready_data)
```

```
class ManagerWorkoffers(LoginRequiredMixin, ListView):
    model = Workoffer
    template_name = 'empviewer/own_department_offers.html'
```

```
def get_queryset(self, *args, **kwargs):
    manager_user = self.request.user
    emp_manager = Employermanager.objects.get(user = manager_user)
    emp_department = emp_manager.department
    workoffers = Workoffer.objects.filter(department = emp_department)
    return workoffers.order_by('-offered')
```

```
class DepartmentWorkoffers(ListView):
    model = Workoffer
    template_name = 'empviewer/department_offers.html'
```

```
def get_queryset(self, *args, **kwargs):
    department = int(self.request.GET.get('dep', '-1'))
    if department > -1:
        workoffers = Workoffer.objects.filter(department__id = department,
state__in = [1, 3])
        return workoffers.order_by('-offered')
    else:
        raise Http404
```

```
class EmployerWorkoffers(ListView):
    model = Workoffer
    template_name = 'empviewer/employer_offers.html'
```

```
def get_queryset(self, *args, **kwargs):
    employer_id = int(self.request.GET.get('emp', '-1'))
    if employer_id > -1:
```

```

        workoffers = Workoffer.objects.filter(department__employer__id =
employer_id, state__in = [1, 3])
        return workoffers
    else:
        raise Http404
class EmployerDepWorkoffers (View):
    template_name = 'empviewer/dep_employer_offers.html'

    def get_queryset(self, *args, **kwargs):
        employer_id = int(self.request.GET.get('emp', '-1'))
        if employer_id > -1:
            workoffers = Workoffer.objects.filter(department__employer__id =
employer_id, state__in = [1, 3])
            group_offers = []
            workpositions = []
            for workoffer in workoffers:
                if workoffer.workposition not in workpositions:
                    pos = workoffer.workposition
                    pos.workoffers = workoffers.filter(workposition = pos)
                    workpositions.append(pos)
            ready_data = {'workpositions': workpositions }
            return render(request, self.template_name, ready_data)
        else:
            raise Http404

class WorkofferTemplateCreate(LoginRequiredMixin, View):
    template_name = 'empviewer/template_create.html'

    def get(self, request, pk = None):
        to_form = FormWorkofferTemplate()
        manager_user = self.request.user
        emp_manager = Employermanager.objects.get(user = manager_user)
        if emp_manager:
            ready_data = {'template_form': to_form }
            return render(request, self.template_name, ready_data)
        else:
            raise Http404

    def post(self, request, pk = None):
        to_form = FormWorkofferTemplate(request.POST, None)
        manager_user = self.request.user

```

```

emp_manager = Employermanager.objects.get(user = manager_user)
emp_department = emp_manager.department
emp_employer = emp_department.employer
if not to_form.is_valid():
    if emp_manager:
        ready_data = {'template_form': to_form }
        return render(request, self.template_name, ready_data)
    else:
        raise Http404
template = FormWorkofferTemplate.save(commit = False)
template.employer = emp_employer
template.save()
template.save_m2m()
return redirect(reverse('empviewer:tdetail', args=[template.id]))

```

```

class EmployerWorkofferTemplates(LoginRequiredMixin, ListView):
    model = WorkofferTemplate
    template_name = 'empviewer/employer_templates.html'
    def get_queryset(self, *args, **kwargs):
        manager_user = self.request.user
        emp_manager = Employermanager.objects.get(user = manager_user)
        emp_department = emp_manager.department
        emp_employer = emp_department.employer
        templates = WorkofferTemplate.objects.filter(employer = emp_employer)
        return templates.order_by('titled_as')

```

```

class FindOffers(ListView):
    model = Workoffer
    template_name = 'empviewer/find_offers.html'

    def get_queryset(self, *args, **kwargs):
        workoffers = super().get_queryset(*args, **kwargs)
        prof = int(self.request.GET.get('prof', '-1'))
        if prof:
            prof = get_object_or_404(Profession, id = prof)
            workoffers = workoffers.filter(workposition__profession = prof)
        salary = int(self.request.GET.get('minsalary', '0'))
        if salary:
            workoffers = workoffers.filter(minsalary_gte = salary)
        minasks = int(self.request.GET.get('minasks', '0'))

```

```

maxasks = int(self.request.GET.get('maxasks', '-1'))
workoffer_ids = []
if minasks or maxasks:
    for offer in workoffers:
        asks = Workask.objects.filter(workoffer = offer)
        asks_num = asks.count()
        add = True
        if not (minasks and asks_num >= minasks):
            add = False
        if not (maxasks > -1 and asks_num <= maxasks):
            add = False
        if add:
            workoffer_ids.append(offer.id)
    workoffers = workoffers.filter(id__in = workoffer_ids)
minscore = float(self.request.GET.get('minscore', '0'))
if minscore > 0:
    workoffers = workoffers.filter(workposition__score__gte = minscore)
miniscore = float(self.request.GET.get('miniscore', '0'))
if miniscore > 0:
    workoffers = workoffers.filter(workposition__interestscore__gte =
miniscore)
minfscore = float(self.request.GET.get('minfscore', '0'))
if minfscore > 0:
    workoffers = workoffers.filter(workposition__financescore__gte =
minfscore)
minsscore = float(self.request.GET.get('minsscore', '0'))
if minsscore > 0:
    workoffers = workoffers.filter(workposition__strategyscore__gte =
minsscore)
minhscore = float(self.request.GET.get('minhscore', '0'))
if minhscore > 0:
    workoffers = workoffers.filter(workposition__humanscore__gte =
minhscore)
city = int(self.request.GET.get('city', '-1'))
if city:
    city = get_object_or_404(City, id = city)
    workoffers = workoffers.filter(department__city = city)
tags = str(self.request.GET.get('tags', ''))
if tags:
    tag_list = tags.split(',')
    workoffers = workoffers.filter(tags__contains = tag_list)
spheres = str(self.request.GET.get('spheres', ''))

```

```

    if spheres:
        spheres_list = spheres.split(',')
        workoffers =
workoffers.filter(department__employer__spheres__contains = spheres_list)
        workoffers = workoffers.order_by('-workposition_score')
    return workoffers

```

```

def get_context_data(self, **kwargs):
    finders = super(FindOffers, self).get_context_data(**kwargs)
    finders['set_params'] = FormFindOffers (initial={
        'prof': self.request.GET.get('prof', '-1'),
        'spheres': self.request.GET.get('spheres', ''),
        'minsalary': self.request.GET.get('minsalary', '0'),
        'tags': self.request.GET.get('tags', ''),
        'minasks': self.request.GET.get('minasks', '0'),
        'maxasks': self.request.GET.get('maxasks', '-1'),
        'minscore': self.request.GET.get('minscore', '0'),
        'miniscore': self.request.GET.get('miniscore', '0'),
        'minfscore': self.request.GET.get('minfscore', '0'),
        'minsscore': self.request.GET.get('minsscore', '0'),
        'minhscore': self.request.GET.get('minhscore', '0'),
        'city': self.request.GET.get('city', '-1')
    })
    return finders

```

```

class WorkofferTemplateModification(LoginRequiredMixin, View):
    template_name = 'empviewer/template_create.html'

```

```

    def get(self, request, pk):
        wotemplate = get_object_or_404(WorkofferTemplate, id = pk)
        to_form = FormWorkofferTemplate (instance = wotemplate)
        manager_user = self.request.user
        emp_manager = get_object_or_404(Employermanager, id =
manager_user.id)
        emp_department = emp_manager.department
        emp_employer = emp_department.employer
        if not emp_manager or emp_employer != wotemplate.employer:
            raise Http404
        ready_data = {'template_form': to_form }
        return render(request, self.template_name, ready_data)

```

```

def post(self, request, pk = None):
    template = get_object_or_404(WorkofferTemplate, id = pk)
    manager_user = self.request.user
    emp_manager = get_object_or_404(Employermanager, id =
manager_user.id)
    emp_department = emp_manager.department
    emp_employer = emp_department.employer
    if not emp_manager or emp_employer != wotemplate.employer:
        raise Http404
    to_form = FormWorkofferTemplate(request.POST, None, instance =
template)
    if not to_form.is_valid():
        ready_data = {'template_form': to_form }
        return render(request, self.template_name, ready_data)
    template = FormWorkofferTemplate.save(commit = False)
    template.employer = emp_employer
    template.save()
    template.save_m2m()
    return redirect(reverse('empviewer:tdetail', args=[template.id]))

```

```

class WorkofferCancel(LoginRequiredMixin, View):

```

```

def post(self, request, pk = None):
    workoffer = get_object_or_404(Workoffer, id = pk)
    manager_user = self.request.user
    emp_manager = Employermanager.objects.get(user = manager_user)
    department = emp_manager.department
    if workoffer.state == 1 and workoffer.department == department:
        workoffer.state = 5
        return redirect(reverse_lazy('empviewer:depwoffers'))
    else:
        raise Http404

```

```

class WorkaskCreate(LoginRequiredMixin, View):

```

```

    template_name = 'empviewer/workask_create.html'

```

```

def get(self, request, pk = None):
    wa_form = FormWorkask()
    employee_user = self.request.user
    employee = Employee.objects.get(user = employee_user)

```

```

if employee:
    ready_data = {'wa_form': wa_form }
    return render(request, self.template_name, ready_data)
else:
    raise Http404

```

```

def post(self, request, pk = None):
    wa_form = FormWorkask(request.POST, None)
    employee_user = self.request.user
    employee = Employee.objects.get(user = employee_user)
    if not wa_form.is_valid():
        if employee:
            ready_data = {'wa_form': wa_form }
            return render(request, self.template_name, ready_data)
        else:
            raise Http404
    workask = FormWorkask.save(commit = False)
    workask.employee = employee
    workask.save()
    return redirect(reverse('empviewer:wadetail', args=[workask.id]))

```

```

class WorkaskConfirm(LoginRequiredMixin, View):

```

```

def post(self, request, pk = None):
    workask = get_object_or_404(Workask, id = pk)
    manager_user = self.request.user
    emp_manager = Employermanager.objects.get(user = manager_user)
    department = emp_manager.department
    if workask.workoffer.department == department:
        workask.state = 1
        return redirect(reverse_lazy('empviewer:offasks'))
    else:
        raise Http404

```

```

class WorkaskCancel(LoginRequiredMixin, View):

```

```

def post(self, request, pk = None):
    workask = get_object_or_404(Workask, id = pk)
    manager_user = self.request.user
    emp_manager = Employermanager.objects.get(user = manager_user)

```

```

department = emp_manager.department
if workask.workoffer.department == department:
    workask.state = 2
    return redirect(reverse_lazy('empviewer:offasks'))
else:
    raise Http404

```

```

class OwnWorkaskCancel(LoginRequiredMixin, View):

```

```

    def post(self, request, pk = None):
        workask = get_object_or_404(Workask, id = pk)
        employee_user = self.request.user
        employee = Employee.objects.get(user = employee_user)
        if workask.employee == employee:
            workask.state = 2
            return redirect(reverse_lazy('empviewer:owasks'))
        else:
            raise Http404

```

```

class WorkofferAsks(LoginRequiredMixin, ListView):

```

```

    model = Workask
    template_name = 'empviewer/offer_asks.html'

    def get_queryset(self, *args, **kwargs):
        workoffer = int(self.request.GET.get('off', '-1'))
        if workoffer > -1:
            workoffer = get_object_or_404(Workoffer, id = workoffer)
        else:
            raise Http404
        manager_user = self.request.user
        emp_manager = Employermanager.objects.get(user = manager_user)
        emp_department = emp_manager.department
        if workoffer.department == emp_department:
            asks = Workask.objects.filter(workoffer = workoffer)
            return asks.order_by('-asked')
        else:
            raise Http404

```

```

class OwnWorkofferAsks(LoginRequiredMixin, ListView):
    model = Workask
    template_name = 'empviewer/ownoffer_asks.html'

    def get_queryset(self, *args, **kwargs):
        employee_user = self.request.user
        employee = Employee.objects.get(user = employee_user)
        if employee:
            asks = Workask.objects.filter(employee = employee)
            return asks.order_by('-asked')
        else:
            raise Http404

class SavedlistEmployerWorkoffers(ListView):
    model = Workoffer
    template_name = 'empviewer/savedlist_offers.html'

    def get_queryset(self, *args, **kwargs):
        savedlist_id = int(self.request.GET.get('slist', '-1'))
        employee_user = self.request.user
        employee = Employee.objects.get(user = employee_user)
        if savedlist_id > -1:
            slist = Savedlist.objects.get(id = savedlist_id)
            if slist.employee != employee:
                raise Http404
            slist_employers = SavedlistEmployer.objects.filter(savedlist = slist)
            woffers = []
            for emp in slist_employers:
                employer = emp.employer
                workoffers = Workoffer.objects.filter(department__employer =
employer, state__in = [1, 3])
                for woff in workoffers:
                    if woff not in woffers:
                        woffers.append(woff)
            return woffers
        else:
            raise Http404

class LastCityWorkoffers(ListView):
    model = Workoffer

```

```

template_name = 'empviewer/last_city_offers.html'

def get_queryset(self, *args, **kwargs):
    city_id = int(self.request.GET.get('city', '-1'))
    if city_id > -1:
        city = City.objects.get(id = city_id)
        workoffers = Workoffer.objects.filter(department__city = city, state__in
= [1, 3])
        workoffers = workoffers.order_by('-offered')[:1000]
        return workoffers
    else:
        raise Http404

class LastTagWorkoffers(ListView):
    model = Workoffer
    template_name = 'empviewer/last_tag_offers.html'

    def get_queryset(self, *args, **kwargs):
        tag = int(self.request.GET.get('tag', ''))
        if tag:
            tag = Tag.objects.get(name = tag)
            workoffers = Workoffer.objects.filter(tags__contains = tag, state__in =
[1, 3])
            workoffers = workoffers.order_by('-offered')[:1000]
            return workoffers
        else:
            raise Http404

class LastSphereWorkoffers(ListView):
    model = Workoffer
    template_name = 'empviewer/last_sphere_offers.html'

    def get_queryset(self, *args, **kwargs):
        sphere = int(self.request.GET.get('sphere', '-1'))
        if sphere:
            sphere = Sphere.objects.get(id = sphere)
            workoffers =
Workoffer.objects.filter(workposition__profession__spheres__contains = sphere,
state__in = [1, 3])
            workoffers = workoffers.order_by('-offered')[:1000]

```

```

        return workoffers
    else:
        raise Http404

```

Б.2 Текст файла `employerviews.py`

```

from empviewer.forms import FormEmployer, FormFindEmployers,
FormWorkposition, FormSavedlist, FormSavedlistEmployer, FormSavedlistWorkoffer
from empviewer.models import Employer, Sphere, Employermanager, City,
Department, Workposition, Profession, Savedlist, SavedlistEmployer,
SavedlistWorkoffer
from django.views.generic import DetailView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.shortcuts import render, redirect, get_object_or_404
from django.http import Http404

```

```

class EmployerModification(LoginRequiredMixin, View):
    template_name = 'empviewer/employer_modify.html'

```

```

    def get(self, request, pk):
        employer = get_object_or_404(Employer, id = pk)
        emp_form = FormEmployer (instance = employer)
        manager_user = self.request.user
        emp_manager = Employermanager.objects.get(user = manager_user)
        department = emp_manager.department
        emp_employer = department.employer
        if employer != emp_employer:
            raise Http404
        spheres = Sphere.objects.all()
        ready_data = {'employer_form': emp_form,
                     'spheres': spheres }
        return render(request, self.template_name, ready_data)

```

```

    def post(self, request, pk = None):
        employer = get_object_or_404(Employer, id = pk)
        manager_user = self.request.user
        emp_manager = Employermanager.objects.get(user = manager_user)
        department = emp_manager.department
        emp_employer = department.employer

```

```

    if employer != emp_employer:
        raise Http404
    emp_form = FormEmployer (request.POST, request.FILES or None,
instance = employer)
    if not emp_form.is_valid():
        spheres = Sphere.objects.all()
        ready_data = {'employer_form': emp_form,
            'spheres': spheres }
        return render(request, self.template_name, ready_data)
    employer = FormEmployer.save(commit = False)
    employer.save()
    employer.save_m2m()
    return redirect(reverse('empviewer:empdetail', args=[employer.id]))

```

```

class FindEmployers(ListView):
    model = Employer
    template_name = 'empviewer/find_employers.html'

    def get_queryset(self, *args, **kwargs):
        employers = super().get_queryset(*args, **kwargs)
        title = str(self.request.GET.get('title', ''))
        if title:
            employers = employers.filter(titled_as__icontains = title)
        minscore = int(self.request.GET.get('minscore', '0'))
        if minscore > 0:
            employers = employers.filter(score__gte = minscore)
        minsscore = int(self.request.GET.get('minsscore', '0'))
        if minsscore > 0:
            employers = employers.filter(strategyscore__gte = minsscore)
        spheres = str(self.request.GET.get('spheres', ''))
        if spheres:
            spheres_list = spheres.split(',')
            employers = employers.filter(spheres__contains = spheres_list)
        employers = employers.order_by('-score')
        city = int(self.request.GET.get('city', '-1'))
        employer_list = []
        if city:
            city = get_object_or_404(City, id = prof)
            for emp in employers:
                department = Department.objects.filter(city = city, employer = emp)
                if department:

```

```

        employer_list.append(emp)
    return employer_list

```

```

def get_context_data(self, **kwargs):
    finders = super(FindEmployers, self).get_context_data(**kwargs)
    finders['set_params'] = FormFindEmployers (initial={
        'title': self.request.GET.get('title', ''),
        'spheres': self.request.GET.get('spheres', ''),
        'minscore': self.request.GET.get('minscore', '0'),
        'minsscore': self.request.GET.get('minsscore', '0'),
        'city': self.request.GET.get('city', '-1')
    })
    return finders

```

```

class EmployerDetail(DetailView):
    model = Employer
    template_name = 'empviewer/employer.html'

    def get(self, request, pk):
        employer = Employer.objects.prefetch_related('spheres').get(id = pk)
        departments = Department.objects.filter(employer = employer)[:5]
        workpositions = Workposition.objects.filter(employer =
employer).order_by('-score')[:5]
        workoffers = Workoffer.objects.filter(department_employer = employer,
state__in = [1, 3]).order_by('till')[:5]
        manager_user = self.request.user
        emp_manager = Employermanager.objects.get(user = manager_user)
        if emp_manager:
            managers = Employermanager.objects.filter(department__employer =
employer).order_by('surname')[:5]
            ready_data = {'employer': employer,
                'departments': departments,
                'managers': managers,
                'workpositions': workpositions,
                'workoffers': workoffers }
        else:
            ready_data = {'employer': employer,
                'departments': departments,
                'workpositions': workpositions,
                'workoffers': workoffers }
        return render(request, self.template_name, ready_data)

```

```

class EmployerDepartments(ListView):
    model = Department
    template_name = 'empviewer/employer_departments.html'
    def get_queryset(self, *args, **kwargs):
        employer_id = int(self.request.GET.get('emp', '-1'))
        if employer_id > -1:
            emp = Employer.objects.get(id = employer_id)
            departments = Department.objects.filter(employer = emp)
            departments = departments.order_by('titled_as')
            return departments
        else:
            raise Http404

```

```

class EmployerWorkpositions(ListView):
    model = Workposition
    template_name = 'empviewer/employer_workpositions.html'

    def get_queryset(self, *args, **kwargs):
        employer_id = int(self.request.GET.get('emp', '-1'))
        if employer_id > -1:
            emp = Employer.objects.get(id = employer_id)
            workpositions = Workposition.objects.filter(employer = emp)
            workpositions = workpositions.order_by('-score')
            return workpositions
        else:
            raise Http404

```

```

class WorkpositionCreate(LoginRequiredMixin, View):
    template_name = 'empviewer/workposition_create.html'

    def get(self, request, pk = None):
        wp_form = FormWorkposition()
        employer_user = self.request.user
        employer_manager = Employermanager.objects.get(user =
employer_user)
        if employer_manager:
            ready_data = {'wp_form': wp_form }
            return render(request, self.template_name, ready_data)
        else:
            raise Http404

```

```

def post(self, request, pk = None):
    wp_form = FormWorkposition (request.POST, None)
    employer_user = self.request.user
    employer_manager = Employermanager.objects.get(user =
employer_user)
    department = employer_manager.department
    employer = department.employer
    if not wp_form.is_valid():
        if employer_manager:
            ready_data = {'wp_form': wp_form }
            return render(request, self.template_name, ready_data)
        else:
            raise Http404
    workposition = FormWorkposition.save(commit = False)
    workposition.employer = employer
    workposition.save()
    return redirect(reverse('empviewer:wpdetail', args=[workposition.id]))

```

```

class WorkpositionModification(LoginRequiredMixin, View):
    template_name = 'empviewer/workposition_create.html'

```

```

def get(self, request, pk):
    workposition = get_object_or_404(Workposition, id = pk)
    wp_form = FormWorkposition (instance = workposition)
    manager_user = self.request.user
    emp_manager = Employermanager.objects.get(user = manager_user)
    department = employer_manager.department
    employer = department.employer
    if employer != workposition.employer:
        raise Http404
    professions = Profession.objects.all()
    ready_data = {'wp_form': wp_form,
                  'professions': professions }
    return render(request, self.template_name, ready_data)

```

```

def post(self, request, pk = None):
    workposition = get_object_or_404(Workposition, id = pk)
    manager_user = self.request.user
    emp_manager = Employermanager.objects.get(user = manager_user)
    department = employer_manager.department
    employer = department.employer

```

```

    if employer != workposition.employer:
        raise Http404
    wp_form = FormWorkposition (request.POST, None, instance =
workposition)
    if not wp_form.is_valid():
        professions = Profession.objects.all()
        ready_data = {'wp_form': wp_form,
            'professions': professions }
        return render(request, self.template_name, ready_data)
    workposition = FormWorkposition.save(commit = False)
    workposition.employer = employer
    workposition.save()
    return redirect(reverse('empviewer:wpdetail', args=[workposition.id]))

```

```

class EmployerProfessionRating(ListView):
    model = Employer
    template_name = 'empviewer/employer_prof_rating.html'

    def get_queryset(self, *args, **kwargs):
        profession_id = int(self.request.GET.get('prof', '-1'))
        strateg = int(self.request.GET.get('strateg', '0'))
        if profession_id > -1:
            prof = Profession.objects.get(id = profession_id)
            workpositions = Workposition(profession = prof)
            employers = []
            for workposition in workpositions:
                if workposition.employer not in employers:
                    employers.append(workposition.employer)
            if strateg:
                employers = employers.order_by('-strategyscore')[:100]
            else:
                employers = employers.order_by('-score')[:100]
            return employers
        else:
            raise Http404

```

```

class EmployerSphereRating(ListView):
    model = Employer
    template_name = 'empviewer/employer_sphere_rating.html'

```

```

def get_queryset(self, *args, **kwargs):
    sphere_id = int(self.request.GET.get('sphere', '-1'))
    strateg = int(self.request.GET.get('strateg', '0'))
    if sphere_id > -1:
        sphere = Sphere.objects.get(id = sphere_id)
        employers = Employer.objects.filter(spheres__contains = sphere)
        if strateg:
            employers = employers.order_by('-strategyscore')[:100]
        else:
            employers = employers.order_by('-score')[:100]
        return employers
    else:
        raise Http404

```

```

class SavedlistCreate(LoginRequiredMixin, View):
    template_name = 'empviewer/savedlist_create.html'

```

```

def get(self, request, pk = None):
    sl_form = FormSavedlist()
    employee_user = self.request.user
    employee = Employee.objects.get(user = employee_user)
    if employee:
        ready_data = {'savlist_form': sl_form }
        return render(request, self.template_name, ready_data)
    else:
        raise Http404

```

```

def post(self, request, pk = None):
    sl_form = FormSavedlist (request.POST, None)
    employee_user = self.request.user
    employee = Employee.objects.get(user = employee_user)
    if not sl_form.is_valid():
        if employee:
            ready_data = {'savlist_form': sl_form }
            return render(request, self.template_name, ready_data)
        else:
            raise Http404
    savlist = FormSavedlist.save(commit = False)
    savlist.employee = employee
    savlist.save()
    return redirect(reverse('empviewer:savlists', args=[employee.id]))

```

```

class EmployeeSavedlists(ListView):
    model = Savedlist
    template_name = 'empviewer/savedlists.html'
    def get_queryset(self, *args, **kwargs):
        employee_user = self.request.user
        employee = Employee.objects.get(user = employee_user)
        savedlists = Savedlist.objects.filter(employee = employee)
        savedlists = savedlists.order_by('named_as')
        return savedlists

class SavedlistEmployerDelete(DeleteView):
    model = SavedlistEmployer

class SavedlistWorkofferDelete(DeleteView):
    model = SavedlistWorkoffer

class SavedlistEmployerAdd(LoginRequiredMixin, View):
    template_name = 'empviewer/add_emp_to_savedlist.html'

    def get(self, request, pk = None):
        sl_form = FormSavedlistEmployer()
        employee_user = self.request.user
        employee = Employee.objects.get(user = employee_user)
        savedlists = Savedlist.objects.filter(employee = employee)
        ready_data = {'savedlist_form': sl_form,
                      'savedlists': savedlists }
        return render(request, self.template_name, ready_data)

    def post(self, request, pk = None):
        sl_form = FormSavedlistEmployer (request.POST, None)
        employee_user = self.request.user
        employee = Employee.objects.get(user = employee_user)
        if not sl_form.is_valid():
            savedlists = Savedlist.objects.filter(employee = employee)
            ready_data = {'savedlist_form': sl_form,
                          'savedlists': savedlists }
            return render(request, self.template_name, ready_data)
        slemp = FormSavedlistEmployer.save(commit = False)
        slemp.save()

```

```
return redirect(reverse('empviewer:slists'))
```

```
class SavedlistWorkofferAdd(LoginRequiredMixin, View):
```

```
    template_name = 'empviewer/add_woff_to_savedlist.html'
```

```
    def get(self, request, pk):
```

```
        sl_form = FormSavedlistWorkoffer()
```

```
        employee_user = self.request.user
```

```
        employee = Employee.objects.get(user = employee_user)
```

```
        savedlists = Savedlist.objects.filter(employee = employee)
```

```
        ready_data = {'savedlist_form': sl_form,
```

```
                      'savedlists': savedlists }
```

```
        return render(request, self.template_name, ready_data)
```

```
    def post(self, request, pk = None):
```

```
        sl_form = FormSavedlistWorkoffer (request.POST, None)
```

```
        employee_user = self.request.user
```

```
        employee = Employee.objects.get(user = employee_user)
```

```
        if not sl_form.is_valid():
```

```
            savedlists = Savedlist.objects.filter(employee = employee)
```

```
            ready_data = {'savedlist_form': sl_form,
```

```
                          'savedlists': savedlists }
```

```
            return render(request, self.template_name, ready_data)
```

```
        slemp = FormSavedlistWorkoffer.save(commit = False)
```

```
        slemp.save()
```

```
        return redirect(reverse('empviewer:slists'))
```

```
class SavedlistDetail(DetailView):
```

```
    model = Savedlist
```

```
    template_name = 'empviewer/savedlist.html'
```

```
    def get(self, request, pk):
```

```
        savedlist = Savedlist.objects.get(id = pk)
```

```
        if savedlist.offered:
```

```
            workoffers = SavedlistWorkoffer.objects.filter(savedlist = savedlist)
```

```
            workoffers = workoffers.order_by('-saved')
```

```
            ready_data = {'savedlist': savedlist,
```

```
                          'workoffers': workoffers }
```

```
        else:
```

```
            employers = SavedlistEmployer.objects.filter(savedlist = savedlist)
```

```
employers = employers.order_by('-saved')
ready_data = {'savedlist': savedlist,
              'employers': employers }
return render(request, self.template_name, ready_data)
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота магістра

**ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ВИЗНАЧЕННЯ ДОСТОВІРНОСТІ ВІДГУКІВ ПРО
ПРАЦЕДАВЦІВ**

**RESEARCH AND SOFTWARE IMPLEMENTATION OF
METHODS FOR VERIFYING THE REVIEWS ABOUT
EMPLOYERS**

Виконав
студент групи КНТ-211м
Керівник
к.т.н., доцент

Д.О. Полумієнко

В.М. Льовкін

2022

Рисунок В.1 – Титульний слайд

2

Об'єкт дослідження – процес пошуку роботи.

Предмет дослідження – методи визначення
достовірності відгуків про працедавців.

Мета роботи – розробити програмне
забезпечення надання доступу до відгуків про
працедавців для підтримки шукачів роботи під
час вибору працедавців.

Рисунок В.2 – Об'єкт, предмет, мета роботи

Завдання

- ✓ проаналізувати програмні засоби надання доступу до відгуків про працедавців;
- ✓ виконати дослідження методів визначення достовірності відгуків про працедавців;
- ✓ виконати проектування програмного забезпечення надання доступу до відгуків про працедавців;
- ✓ реалізувати програмне забезпечення надання доступу до відгуків про працедавців.

Рисунок В.3 – Завдання роботи

Порівняння програмних аналогів

Критерій	Glassdoor	Indeed	Empvieuwer
Пошук вакансій	–	+	+
Подання запитів на роботу	–	+	+
Перегляд відгуків про працедавців	+	+	+
Додавання власних відгуків про працедавців	+	+	+
Визначення достовірності відгуків	–	–	+
Створення збережених списків	+	+	+

Рисунок В.4 – Порівняння програмних аналогів

Метод визначення достовірності відгуків про працедавців

Збір даних про відгуки про працедавців, щодо яких вже встановлено достовірність, і формування з них вибірки даних для навчання.

Попередня обробка тексту відгуку. Під час даного процесу необхідно вилучити всі зайві символи, символи, які не входять до базового алфавіту.

Після попередньої обробки має бути виконано векторизацію текстів відгуків за допомогою використання TF-IDF.

Застосування алгоритму RF для навчання моделі за допомогою вибірки даних, яку було створено на попередніх етапах.

Застосування створеної моделі для даних, які ще не вивчались. Для кожного такого відгуку також має реалізовуватися зведення до форми, представленої вище, без останньої ознаки, а також попереднє оброблення тексту відгуку і векторизація за допомогою TF-IDF. Результат роботи етапу – клас достовірності відгуку про працедавця.

Рисунок В.5 – Метод визначення достовірності відгуків про працедавців

Порівняння фреймворків для розробки вебдодатків

Показник / фреймворк	Django	CherryPy
Доступність механізму об'єктно-реляційного перетворення	Так	Ні
Адміністративна панель	Так	Ні
Засоби підтримки безпеки за замовчуванням	Так, постійно розширюються	Ні
Аутентифікація користувачів	Так, надаються широкі засоби керування записами користувачів	Так
Підтримувані рушії шаблонів	За замовчуванням, можна змінити	За замовчуванням немає, можна підключити

Рисунок В.6 – Порівняння фреймворків для розробки вебдодатків

Діаграма варіантів використання

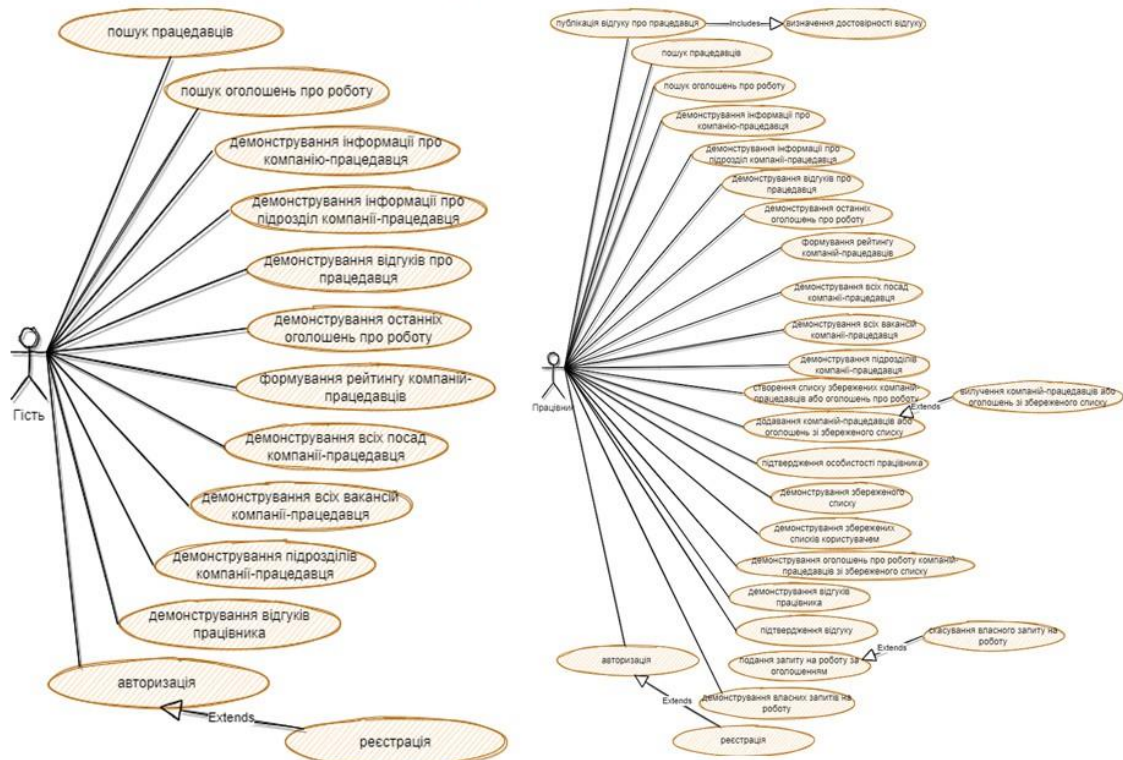


Рисунок В.7 – Діаграма варіантів використання

Діаграма варіантів використання

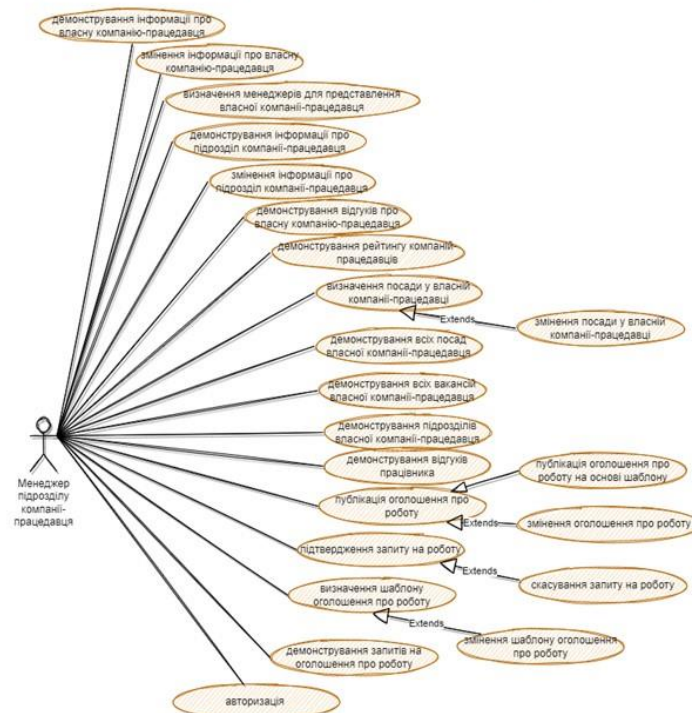


Рисунок В.8 – Діаграма варіантів використання

Схема бази даних

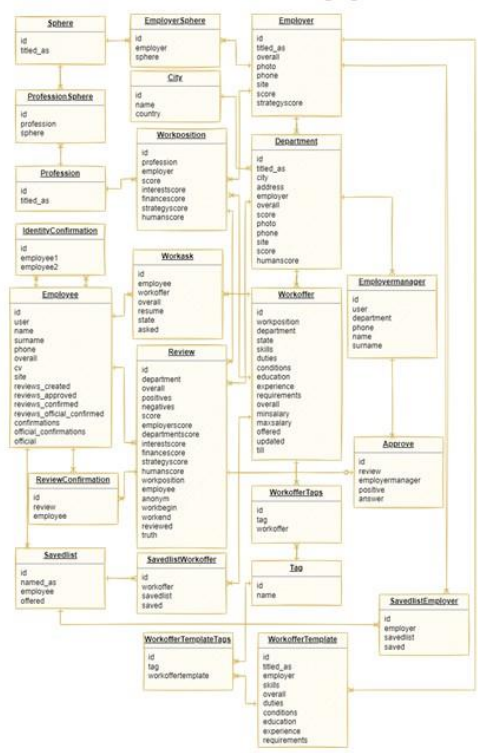


Рисунок В.9 – Схема БД

Діаграма станів оголошення про роботу

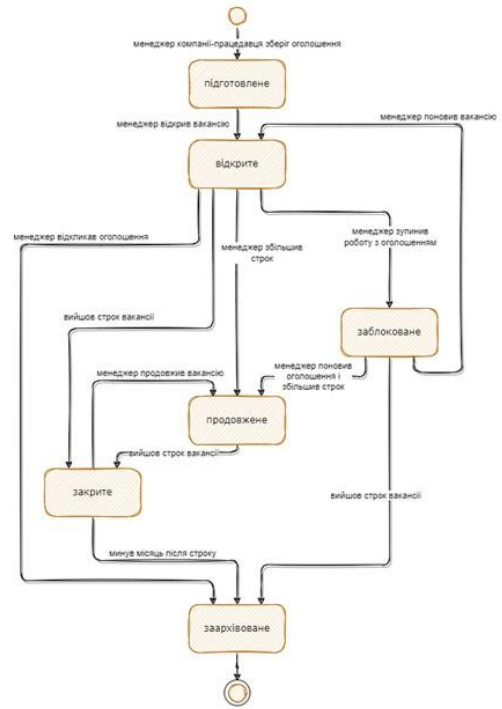


Рисунок В.10 – Діаграма станів оголошення про роботу

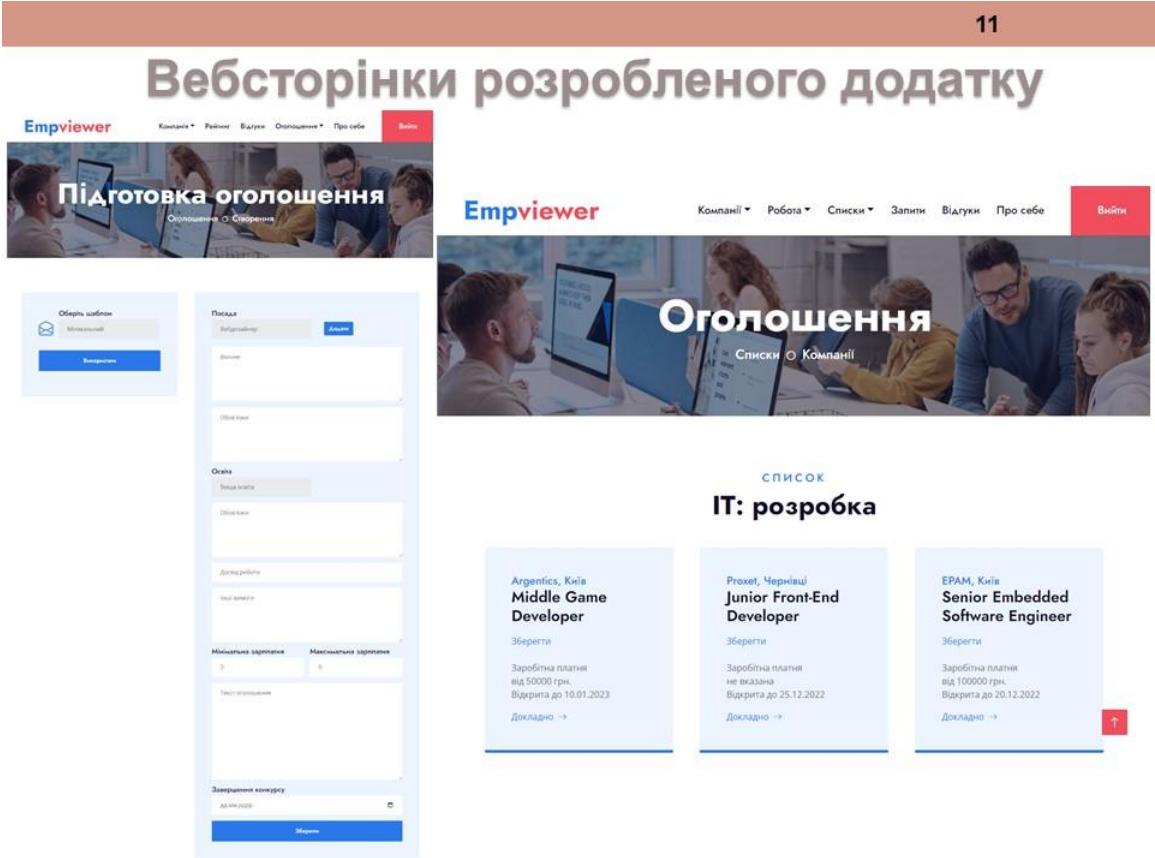


Рисунок В.11 – Вебсторінки розробленого додатку

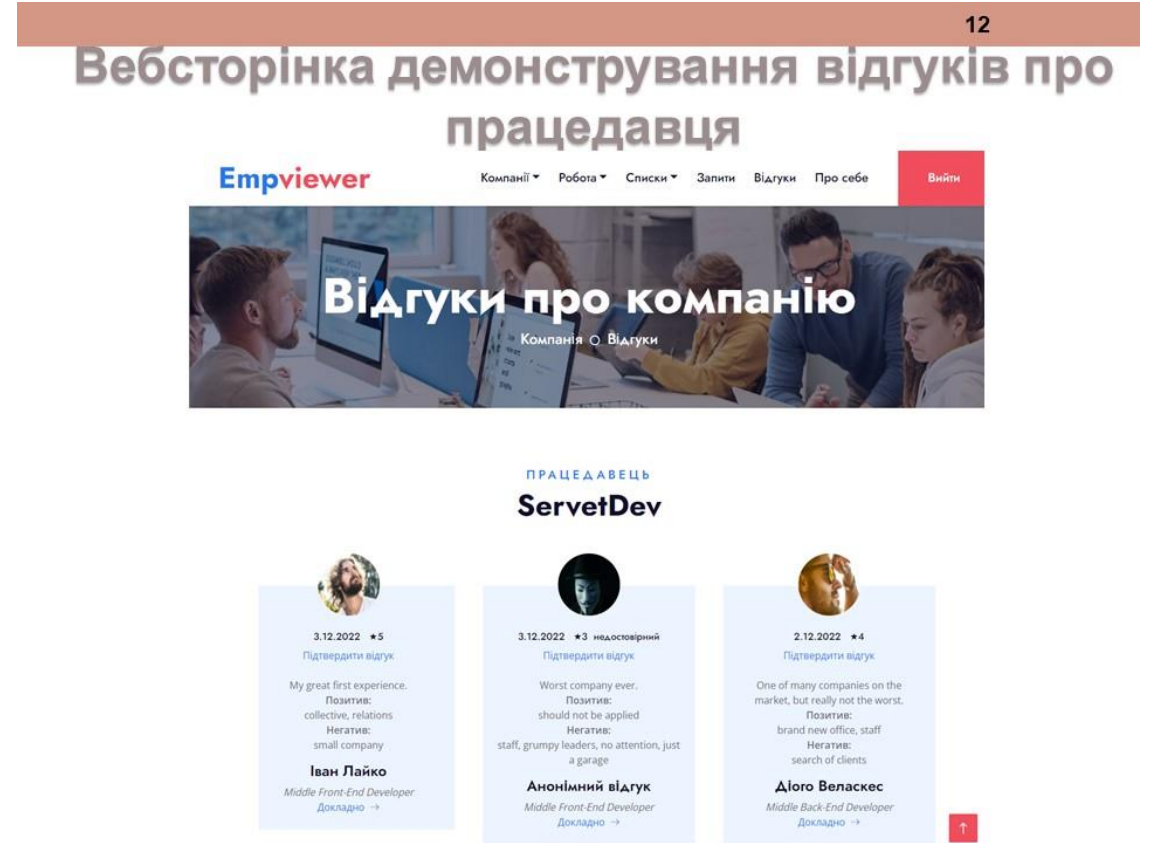


Рисунок В.12 – Вебсторінка демонстрування відгуків про працедавця

Результати експериментального дослідження

Метод класифікації	Точність, %
Наївний байєсів класифікатор	51,61
Метод визначення достовірності відгуків про працедавців	82,87

Рисунок В.13 Результати експериментального дослідження

Висновки

Результати роботи охоплюють програмне забезпечення, що підтримує як пошук роботи через доступ до оголошень про вакансії, подання запитів на роботу, керування цими запитами, так і отримання інформації про працедавців через відгуки, а зокрема і перехід від оголошення до відгуків за цією посадою. Однією з відмінностей роботи є визначення достовірності відгуків, що дозволяє шукачу роботи розуміти, на скільки варто довіряти відомостям з відгуку. При цьому забезпечується це різними інструментами: програмно результатами використання інтелектуального методу, підтвердженням відгуків іншими користувачами, для яких передбачено також можливість офіційного підтвердження особи, підтвердженням або реагуванням на відгуки зі сторони самих працедавців.

Науковою новизною роботи є метод визначення достовірності відгуків про працедавців, який дозволяє класифікувати відгуки про працедавців щодо їх достовірності або недостовірності, виконуючи векторизацію тексту відгуку за допомогою моделі TF-IDF і використовуючи алгоритм RF для реалізації класифікації.

Рисунок В.14 – Висновки