

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПІДТРИМКИ ФОРМУВАННЯ ОГолошень ПРО ПРАЦЕВЛАШТУВАННЯ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS TO
SUPPORT CREATING JOB ADVERTISEMENTS

Виконав: студент(ка) 2 курсу, групи КНТ-210м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Сердюк М.С.
(прізвище та ініціали)

Керівник Зайко Т.А.
(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 ІРЕ, 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

“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Сердюка Максима Сергійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів підтримки формування оголошень про працевлаштування

Research and Software Implementation of Methods to Support Creating Job Advertisements

керівник проєкту (роботи) Зайко Тетяна Анатоліївна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “5” листопада 2021 року №435

2. Строк подання студентом проєкту (роботи) грудень 2021 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	Зайко Т.А., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “9” вересня 2021 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) Сердюк М.С.
(прізвище та ініціали)

Керівник проєкту (роботи)

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

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
92 с., 23 рис., 3 табл., 3 дод., 17 джерел.

ОГОЛОШЕННЯ ПРО ПРАЦЕВЛАШТУВАННЯ, ВАКАНСІЇ,
РЕЗЮМЕ, ПРОГНОЗУВАННЯ, ДЕРЕВА РІШЕНЬ, ВЕБЗАСТОСУНОК,
DJANGO.

Об'єкт дослідження – процес формування оголошень про
працевлаштування.

Предмет дослідження – методи прогнозування періоду доступності
оголошення про працевлаштування.

Мета роботи – розробити програмне забезпечення формування і
пошуку оголошень про працевлаштування для надання інформації для
прийняття рішень про вибір конкретного формулювання вакансії на основі
прогнозування періоду її доступності.

Матеріали, методи та технічні засоби: мова програмування Python,
система керування базами даних MySQL, вебфреймворк Django.

Результати. Наукова новизна роботи полягає в методі прогнозування
періоду доступності вакансії на працевлаштування, який відзначається
застосуванням показника tf-idf та методу опорних векторів, визначаючи
таким чином часову категорію, що дозволяє приймати в подальшому рішення
щодо вибору конкретного формулювання вакансії.

Висновки. Розроблена програма призначена для формування
оголошень про працевлаштування, пошуку вакансій на ринку, виставлення
власних резюме, пошуку серед наявних резюме. Програма може
використовуватися як кандидатами, так і представниками компаній.

Галузь використання – створення ринку вакансій для
працевлаштування.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 92 pages, 23 figures, 3 tables, 3 appendixes, 17 sources.

ANNOUNCEMENTS OF EMPLOYMENT, VACANCIES, RESUMES, FORECASTING, DECISION TREES, WEB APPLICATION, DJANGO.

The object of research is the process of generating job advertisements.

The subject of research is methods of forecasting the period of availability of employment advertisements.

The purpose of the work is to develop software for the formation and search for job advertisements to provide information for decision-making on the choice of a particular vacancy wording based on forecasting the period of its availability.

Materials, methods and technical means: Python programming language, MySQL database management system, Django web framework.

Results. The scientific novelty of the work lies in the method of forecasting the period of vacancy availability, which is marked by the use of tf-idf and the method of reference vectors, thus determining the time category that enables further decisions on the choice of specific vacancy wording.

Conclusions. The developed program is designed to generate job advertisements, search for vacancies in the market, display user own resumes, search among available resumes. The program can be used by both candidates and company representatives.

Field of use is a creation of job market for employment.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Огляд проблем пошуку вакансій про працевлаштування.....	11
1.2 Огляд програмних аналогів.....	12
1.3 Висновки за розділом 1	15
2 Матеріали і методи.....	17
2.1 Методи оброблення тексту	17
2.2 Метод опорних векторів.....	19
2.3 Метод прогнозування періоду доступності вакансії на працевлаштування	20
2.4 Висновки за розділом 2	22
3 Проєктування програмного забезпечення	23
3.1 Вибір засобів проєктування і розробки	23
3.2 Моделювання роботи з програмою	25
3.3 Проєктування бази даних	27
3.4 Структура програмного забезпечення	34
3.5 Висновки за розділом 3	35
4 Основні рішення щодо реалізації компонентів системи.....	37
4.1 Алгоритм функціонування системи.....	37
4.2 Опис розроблених класів програми	38
4.3 Висновки за розділом 4	47
5 Експлуатація, тестування та експериментальне дослідження програми	48
5.1 Призначення програми	48
5.2 Умови виконання програми	48
5.3 Результати роботи програми.....	48
5.4 Експериментальне дослідження	51
5.5 Висновки за розділом 5	52

Висновки	53
Перелік джерел посилання	55
Додаток А Технічне завдання	57
Додаток Б Текст програми	61
Додаток В Слайди презентації.....	85

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

SQL – structured query language;

БД – база даних.

ВСТУП

Кількість вакансій в різних країнах залежить від стану економіки, але зараз в період активної пандемії загалом збільшується. У Сполучених Штатах Америки працює понад 161 мільйон людей і відкрито 15,7 мільйона робочих місць, що означає, що приблизно одне з 10 місць у кожному офісі порожнє, і реалізується конкуренція з усіма іншими за персонал, до того ж активно створюються нові робочі місця: за останній місяць у Сполучених Штатах Америки було створено понад 530 000 нових робочих місць, а рівень безробіття зараз знизився до 4,6% і знижується далі, за прогнозом наступного року буде понад 20 мільйонів вакансій, в результаті чого відсоток незаповнених місць буде становити одне з восьми [1].

Перманентною частиною життя кожної людини є працевлаштування. Цей процес потребує від його учасників, як представників компаній, так і кандидатів на вакантні місця, особливих вмінь. Здебільшого ці вміння стосуються правильної організації процесу, вміння знаходити і обирати потрібні вакансії з одного боку, а з іншого боку – формувати такі вакансії, які будуть привертати увагу значної аудиторії.

Оцінюючи вакансію, представникам компанії важливо розуміти, скільки часу займе процес пошуку працівників, а кандидатам важливо розуміти, на скільки вакансія є популярною, розраховуючи з цього власні сили, щоб не витратити час дарма у випадку недостатньо високих конкурентних позицій.

Програмне забезпечення, що має підтримувати даний процес, повинно надавати компаніям можливість проаналізувати сформовану вакансію і спрогнозувати для неї час, який вона прогнозовано буде доступною на ринку (активною). Тобто акцент в даній роботі зроблено саме на вакансії – пропозиції компаній, а не резюме як пропозиції самих кандидатів на працевлаштування.

Тому метою роботи було визначено розробити програмне забезпечення формування і пошуку оголошень про працевлаштування для надання інформації для прийняття рішень про вибір конкретного формулювання вакансії на основі прогнозування періоду її доступності.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд проблем пошуку вакансій про працевлаштування

Перед тим, як виконувати огляд програмних аналогів виконуваної розробки, потрібно спочатку проаналізувати те, які фактори є важливими у процесі пошуку вакансій кандидатом на працевлаштування. Саме ці характеристики потрібно також врахувати і за результатами наступного огляду, формуючи потім перелік функціональних вимог до програми.

Тож проблемами пошуку вакансій, комунікації в даному пошуку є наступні:

- впевненість людей, які шукають роботу, часто знижується: замість того, щоб концентруватися на невдалих спробах, потрібно знайти якомога більше вакансій, адже це не тільки прискорить пошук роботи, але й дозволить легше підтримувати свій моральний дух;

- відсутність відповідей на резюме: потрібно кількісно оцінити своє резюме, якщо не отримано жодної відповіді на надіслане резюме, потрібно змінити резюме;

- телефонні інтерв'ю завершуються невдачею: телефонні співбесіди – це процес перевірки від роботодавців, таким чином вони визначають, чи варто зустрітися з кандидатом віч-на-віч, тому потрібно підготуватися до того, щоб телефонне інтерв'ю завершилося успіхом;

- проблеми зі співбесідою: існує багато способів зазнати невдачі на співбесіді, так до співбесіди потрібно обов'язково дослідити компанію, за можливості інтерв'юєра, краща підготовка до співбесіди призводить до кращих результатів, потрібно отримати список поширених запитань на співбесіду, підготувати на них хороші відповіді та попрактикувати свої відповіді, практика важлива, тому що вона допоможе здаватися впевненішими та здібнішими;

- невміння продати себе: продавати себе означає відповідати своїм навичкам, здібностям, досвіду та рисам роботи та вказувати на це

роботодавцю, тому потрібно подивитися уважно на те, що потрібно цьому менеджеру з найму, а потім показати, як це можна забезпечити, не потрібно розраховувати на те, що рекрутери побачать це самі.

– кандидат добре проходить співбесіду, але не отримує пропозицій: коли люди, які шукають роботу, впевнені, кваліфіковані, виразні та характерні, але все ще не отримують пропозиції, вони зазвичай займають друге або третє місце в перегонах кандидатів, їм потрібен спосіб вийти на перше місце, і це не завжди великий поштовх, перший спосіб вийти з цієї ситуації – створити план на 30-60-90 днів, це письмовий опис кроків і пріоритетів у перші 3 місяці роботи, потім потрібно разом із інтерв'юером розглянути цей план і отримати зворотній зв'язок [2]-[4].

1.2 Огляд програмних аналогів

Основною своєю функціональністю існуючі вебсайти для пошуку вакансій і відповідно підтримки створення цих вакансій достатньо схожі.

Вебсайт jobs.ua (рис. 1.1) є одним із засобів пошуку вакансій в Україні [5].

Вебсайт надає можливість пошуку вакансій, перегляду гарячих вакансій, перегляду всіх наявних вакансій, вакансій за містами та вакансій для роботи за кордоном. Це основні його можливості для кандидатів на працевлаштування. Окрім того користувач може завантажити за вакансією власне резюме або відгукнутися на вакансію без резюме, написавши відповідне повідомлення. Також можна переглянути перелік вакансій за компанією, можна створити власне резюме просто на сайті, ввівши дані про себе за різними параметрами.

Окремий режим передбачено для роботи роботодавців з програмою. Спосіб групування резюме подібний до того, як це реалізовано для вакансій. При цьому варто відзначити, що кожна вакансія і резюме мають свої ярлики

(рівень резюме, терміновість можуть бути встановлені), але відобразити всі резюме або вакансій з такими ярликами неможливо.

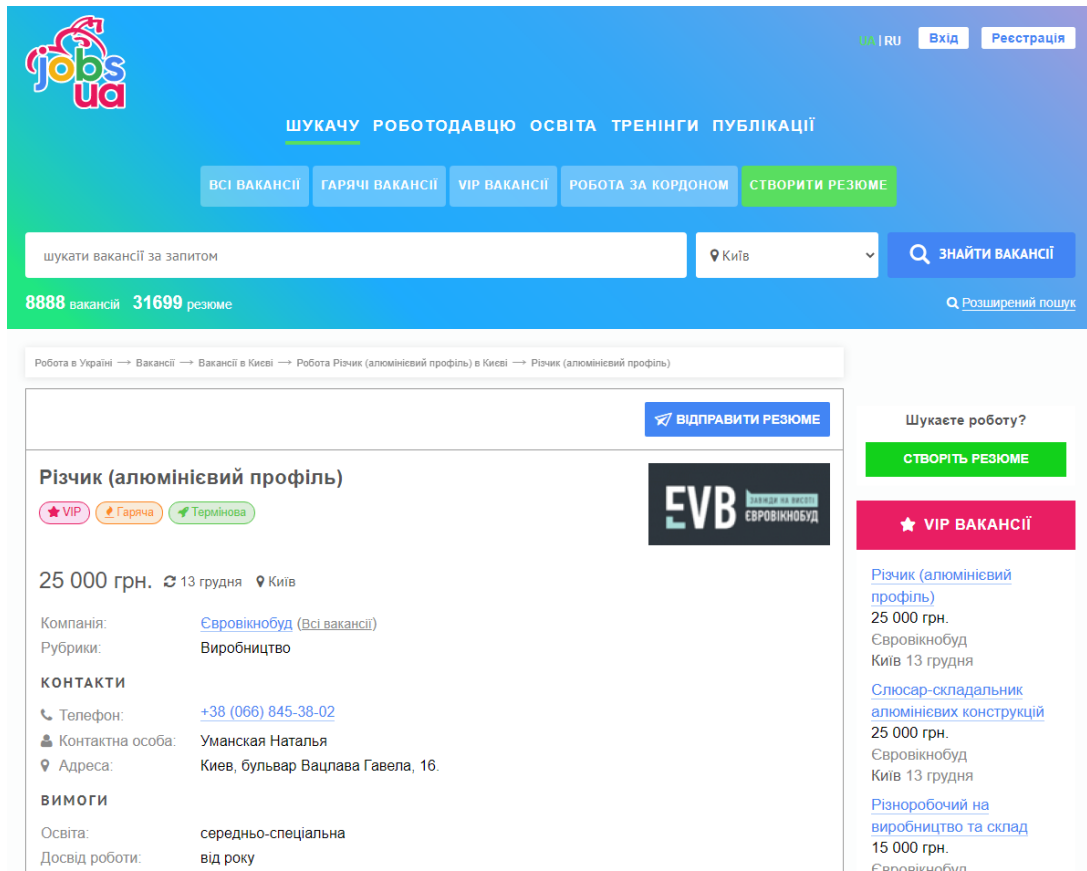


Рисунок 1.1 – Вебсайт jobs.ua

Вебсайт jooble.org (рис. 1.2) призначений для пошуку вакансій на території Сполучених Штатів Америки [6].

Даний програмний застосунок надає можливість гнучкого пошуку вакансій, включаючи такі параметри як період публікації вакансії, рівень заробітної платні, місто роботи, тип зайнятості. За результатами пошуку за кожною вакансією представляється назва вакансії, короткий опис, місто та період з дати публікації оголошення. При переході до оголошення надається його детальний опис і надається можливість подати власне резюме. При цьому на відміну від попереднього рішення програма не надає можливість створити своє резюме для пошуку його роботодавцем, тобто пошук реалізований одностороннім чином.

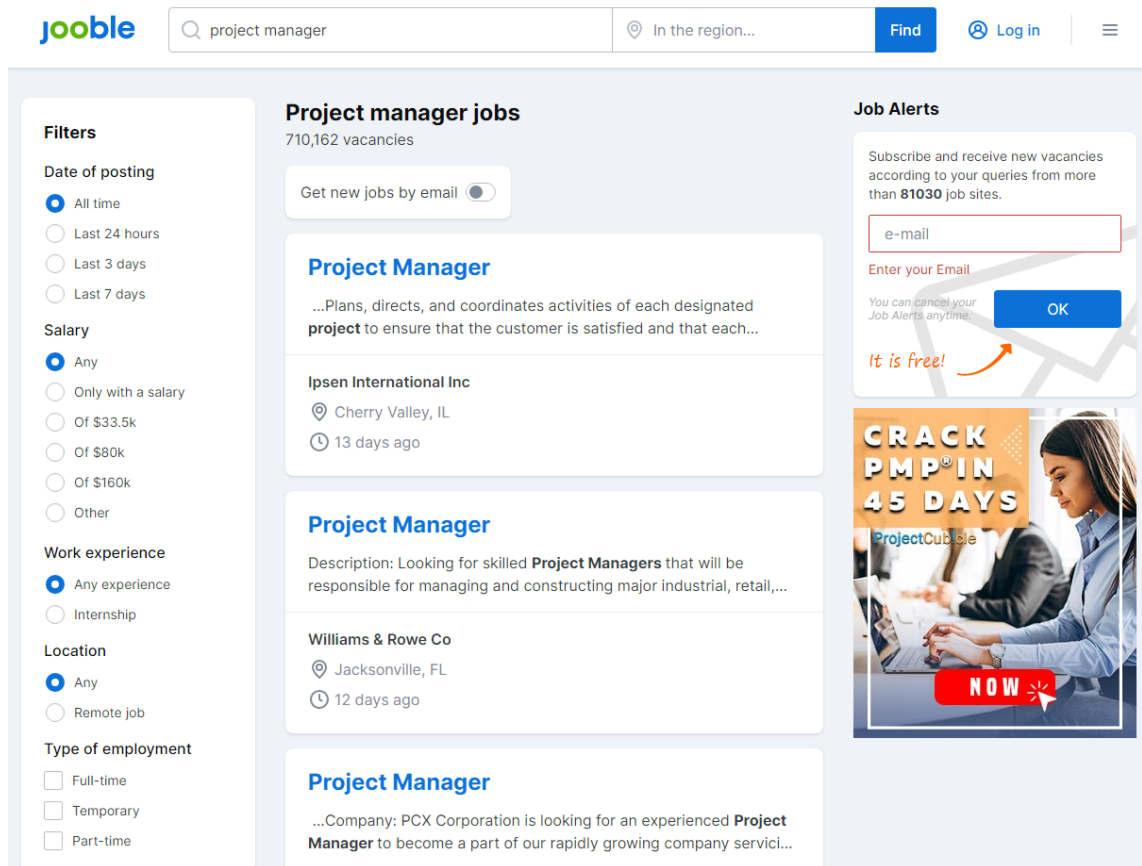


Рисунок 1.2 – Вебсайт jooble.org

За результати проведеного огляду було виконано порівняння аналогів. Окрім даних двох аналогів аналізувалися і інші, але вони були обрані як найбільш показові.

Дані два аналоги відрізняються своїми підходами до роботи в першу чергу: якщо jobs.ua дозволяє шукати як вакансії, так і резюме, то jooble.org працює тільки з вакансіями. Це різні підходи до організації комунікації двох сторін процесу пошуку роботи.

Варто відзначити, що жоден з цих засобів не надає можливість сортування вакансій або резюме потрібним чином. Тобто вплинути на стрічку вакансій можна, обираючи параметри пошуку, але далі відсортувати неможливо.

До того ж створення вакансій відбувається заповненням відповідних форм і інтелектуальних засобів не надається. Роботодавець не може отримати

прогнозу оцінку часу доступності його вакансії. Як раз ці недоліки було вирішено врахувати в розробці, яка виконується.

У табл. 1.1 приведено результати порівняння програмних аналогів.

Таблиця 1.1 – Порівняння програмних аналогів

Характеристика	jobs.ua	jooble.org	Власна розробка
Пошук вакансій	+	+	+
Пошук резюме	+	—	+
Сортування результатів пошуку вказаним користувачем чином	—	—	+
Наявність ярликів оголошень і пошук за ярликами оголошень	+—	—	++
Прогнозування часу доступності вакансій	—	—	+

Таким чином, отримані результати проведеного аналізу програмних аналогів вказують на те, що розробка, виконувана в роботі, є актуальною, адже дозволяє підвищити зручність роботи з вакансіями порівняно з існуючими програмними засобами.

1.3 Висновки за розділом 1

Виконано огляд проблем, які виникають під час пошуку вакансій, та проведено аналіз програмних аналогів рішення, що розробляється. В результаті було виявлено, що вони не мають засобів прогнозування періоду доступності вакансії на працевлаштування, що стало особливістю програми,

яка розроблялась. До того ж програми не дозволяють сортувати результати пошуку зручним для користувачів чином.

У дипломній роботі необхідно розв'язати такі завдання:

- провести аналіз предметної області, що було виконано в даному розділі;
- провести дослідження методів прогнозування періоду доступності вакансії на працевлаштування;
- реалізувати проектування програми для формування і пошуку оголошень про працевлаштування;
- реалізувати програму для формування і пошуку оголошень про працевлаштування.

2 МАТЕРІАЛИ І МЕТОДИ

Вакансії для працевлаштування представлені основним чином своїм описом. Тому під час вирішення проблеми прогнозування часу доступності вакансій потрібно вирішити проблему перетворення текстових даних на числовий формат, який здатні використовувати моделі класифікації.

2.1 Методи оброблення тексту

Одним з активно використовуваних методів на даний момент в цій галузі є метод word2Vec.

Метод word2Vec створює вектори слів, які є розподіленими числовими представленнями властивостей слів, – ці характеристики слова можуть складатися зі слів, які представляють контекст окремих слів, присутніх у словнику, вбудовування слів зрештою допомагає встановити асоціацію слова з іншим подібним словом за допомогою створених векторів:

- схожі за значенням слова розташовані ближче в просторі, що вказує на їх семантичну подібність;

- дві різні архітектурні моделі можуть використовуватися word2vec для вбудовування слів, – це модель безперервного пакета слів і модель skip-gram;

- незважаючи на те, що word2vec є моделлю навчання без учителя, де можна надати корпус без будь-якої інформації про мітки, а модель може створювати щільні вбудовування слів, word2vec внутрішньо використовує контрольовану модель класифікації, щоб отримати ці вбудовування з корпусу, архітектура безперервного пакета слів містить модель класифікації глибокого навчання, в якій беруться контекстні слова як вхідні дані і намагаються передбачити цільове слово;

- модель безперервного пакета слів працює наступним чином: контекстні слова спочатку передаються як вхідні дані до шару вбудовування

(ініціалізованого деякими випадковими ваговими показниками), потім слова передаються на лямбда-шар, де вирівнюються усереднення слів, потім передаються ці вбудовування в щільний шар softmax, який передбачає цільове слово, отриманий результат порівнюється з цільовим словом і обчислюється втрата, а потім виконується зворотне поширення з кожною епохою, щоб оновити шар вбудовування в процесі;

– у моделі skip-gram, якщо дане цільове слово, то прогножуються контекстні слова, загалом модель працює на основі наступної процедури: і цільові, і контекстні пари слів передаються на окремі шари вбудовування, з яких отримуються щільні вбудовування слів для кожного з цих двох слів, потім використовується шар злиття, щоб обчислити точковий добуток цих двох вкладень, це значення точкового добутку потім надсилається до щільного сигмовидного шару, який виводить 0 або 1, вихідні дані порівнюються з фактичною міткою, і втрата обчислюється з подальшим зворотним поширенням з кожною епохою для оновлення шару вбудовування в процесі [7]-[8].

Даний метод є достатньо практично значимим, але він характеризується використанням глибинних мереж, що пов'язано з великим часом навчання, великими вимогами до даних (щодо обсягів цих даних), до того ж пояснити рішення, які приймаються такою моделлю складно. Тому було вирішено використовувати більш простий підхід до перетворення текстових даних в числову форму. Такий підхід ґрунтується на оцінці TF-IDF.

TF-IDF розшифровується як обчислення того, на скільки слово в серії або корпусі є релевантним тексту, значення збільшується пропорційно кількості зустрічей слова в тексті, але компенсується частотою слів у корпусі (наборі даних), тож стосовно даної оцінки можна виділити наступне:

– частота термінів: в документі частота представляє кількість екземплярів даного слова, для кожного конкретного терміна в тексті цієї

дипломної роботи є запис із значенням частоти терміна, вага терміна, який зустрічається в документі, просто пропорційна частоті терміну;

- частота документів перевіряє значення тексту, який дуже схожий на частоту термінів, у всій колекції корпусу, єдина відмінність полягає в тому, що в документі показник tf – це лічильник частоти для терміна t , а df – кількість входжень терміна t у наборі документів n , іншими словами, кількість документів, у яких присутнє слово, дорівнює df ;

- зворотна частота документів: в основному перевіряє, наскільки релевантне слово, основна мета пошуку – знайти відповідні записи, які відповідають вимогам, оскільки tf розглядає всі терміни однаково значущими, то можна використовувати частоту термінів для вимірювання ваги терміна в документі;

- частота термінів – це кількість екземплярів терміна лише в одному документі, хоча частота документа – це кількість окремих документів, у яких зустрічається термін, вона залежить від усього корпусу;

- idf слова – це кількість документів у корпусі, розділених на частоту тексту;

- оцінка $tf-idf$ є одним із найкращих показників для визначення важливості терміна для тексту в серії або корпусі, $tf-idf$ – це система зважування, яка призначає вагу кожному слову в документі на основі його частоти термінів і частоти документа, більш значущими вважаються слова з більшою вагою, при цьому загальна вага зазвичай складається з двох складових: нормованої частоти термінів та зворотної частоти документів [9]-[11].

2.2 Метод опорних векторів

Метод опорних векторів – це потужний алгоритм, який найкраще працює на невеликих наборах даних, але на складних, може

використовуватися як для завдань регресії, так і для завдань класифікації, але загалом метод найкраще працює у задачах класифікації [12].

Відносно даного метода можна виділити наступне:

- це метод машинного навчання з учителем, де намагаються знайти гіперплощину, яка найкраще розділяє два класи;
- на відміну від логістичної регресії, яка представляє ймовірнісний підхід, метод опорних векторів заснований на статистичних підходах;
- задача методу – обрати таку гіперплощину серед всієї нескінченної їх кількості, що проходять через одну точку, що ідеально виділяє два класи;
- метод знаходить таку гіперплощину, що має максимальні відстані між двома класами;
- тільки тоді, коли дані ідеально лінійно розділені, можна використовувати лінійний метод опорних векторів: ідеальна лінійна роздільність означає, що точки даних можна класифікувати на 2 класи за допомогою однієї прямої лінії;
- нелінійний метод опорних векторів може використовуватися тоді, коли дані не є лінійно розділеними, що означає, що коли точки даних не можна розділити на 2 класи за допомогою прямої лінії, так у більшості реальних додатків не знаходяться лінійно розділені точки даних;
- опорні вектори – це точки, найближчі до гіперплощини: за допомогою цих точок даних буде визначено розділову лінію;
- розділення – це відстань між гіперплощиною та спостереженнями, найближчими до гіперплощини (опорними векторами): у методі опорних векторів велика відстань вважається хорошим розділенням [12]-[13].

2.3 Метод прогнозування періоду доступності вакансії на працевлаштування

Для прогнозування періоду доступності вакансії використовувалась постановка задачі, яка є задачею класифікації. Класифікація відбувається з

метою визначення часової категорії, до якої належить текстове повідомлення, яке представляє опис вакансії для працевлаштування.

Було виділено наступні часові категорії:

- період до 1 доби: у такому випадку час, який минув з дати публікації вакансії і до її закриття, не перевищує 1 доби;
- період до 5 діб: період, протягом якого оголошення було активним є більше ніж 1 доба, але не перевищує 5 діб;
- період до 10 діб;
- період до 1 місяця;
- тривалі: робота над пошуком працівника продовжувалась більше 1 місяця (можливо не була завершена успішно).

Відповідно запропонований метод і мав визначити за текстовим описом вакансії, до якої з цих 5 часових категорій в результаті буде належати вакансія. Маючи таку інформацію, працедавець може приймати рішення про переформулювання оголошення для того, щоб змінити необхідним йому чином час очікування. Таким чином, отримана інформація є основою для прийняття рішень.

На першому етапі визначається набір даних, що складається з пар «текстовий опис вакансії – часова категорія».

На другому етапі виконується передоброблення тестових даних шляхом виділення з них слів, вилучення знаків пунктуації, спеціальних символів

На третьому етапі необхідно провести дослідження текстових описів усіх вакансій та виконати попередні обчислення частот слів на корпусі текстів, отриманого на першому етапі, з виділенням слоника слів.

На четвертому етапі потрібно провести навчання моделі. Для цього кожний отриманий опис з вибірки даних потрібно обробити, перетворюючи його у векторну форму, що складається з частот слів, а далі отриману оцінку подати на вхід метода опорних векторів.

На п'ятому етапі реалізується прогнозування часової категорії періоду доступності вакансії для працевлаштування на основі реальних даних.

2.4 Висновки за розділом 2

У даному розділі розглянуто методи, які можуть використовуватися для оброблення тексту з метою перетворення його з текстової форми на форму, придатну для моделей класифікації. В якості моделі класифікації розглянуто метод опорних векторів. Наукова новизна роботи полягає в методі прогнозування періоду доступності вакансії на працевлаштування, який відзначається застосуванням показника tf-idf та методу опорних векторів, визначаючи таким чином часову категорію, що дозволяє приймати в подальшому рішення щодо вибору конкретного формулювання вакансії.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів проєктування і розробки

При виборі мови програмування для реалізації програмного проєкту було враховано наступні фактори:

- проєкт має бути розроблений у вигляді вебзастосунка, щоб дозволити широкому колу користувачів працювати з програмою на різних пристроях, при чому бажано, щоб це було кросплатформне рішення, що дозволить розширити в майбутньому платформи, на яких працює програма;
- проєкт має реалізовувати метод, який було представлено в розділі 2, тобто використовувати конкретні інтелектуальні технології.

Мовою програмування, яка з максимальною зручністю поєднує в собі обидві можливості, на даний момент є мова програмування Python. Вона надає велику кількість засобів для роботи з інтелектуальними технологіями в програмі, при цьому використання цих засобів максимально просте, адже зазвичай включає виклик відповідної функції з передачею в неї необхідних параметрів, до того ж надається велика кількість бібліотек, які підтримують обробку даних, їх організацію зручним чином. До того ж мова Python дозволяє створювати кросплатформні рішення і є однією з відомих мов для організації сучасних вебзастосунків. Тому саме на її основі вирішено було будувати програмний проєкт.

Але мова Python зручна не тільки своїми внутрішніми засобами, але і підтримкою вебфреймворків розробки. Найвідомішими з них є два – Flask і Django [14]-[15]. Тому саме їх було порівняно під час вибору засобів проєктування і розробки.

У результаті було обрано вебфреймворк Django, враховуючи наступні характеристики:

- об'єктно-реляційне відображення: наявність об'єктно-реляційного відображення дуже важлива для роботи з даними, адже у випадку його відсутності доводиться створювати багато механізованого коду для роботи з

базою даних (БД) і витрачати значний час на цей процес, відповідно за цим критерієм слід вказати на те, що об'єктно-реляційне відображення надається Django одразу після встановлення, а у Flask для цього потрібно його окремо встановити;

- уникнення structured query language (SQL)-ін'єкцій: SQL-ін'єкція є важливою прогалиною в системі захисту вебзастосунків, які працюють з БД, у такому випадку користувач може ввести SQL-запит в полі для введення даних, відповідно вебзастосунок його прочитає і виконає, якщо ніяких механізмів захисту не передбачено, тобто злоумисник може впливати на дані, з якими працює програма, що і характерно для вебфреймворку Flask, а для того щоб вирішити цю проблему, потрібно налаштовувати додаткові засоби захисту, які потребують витрат на часу на їх пошук, налаштування у складі системи, а через відсутність налаштованого і відлагодженого з'єднання таке рішення може бути ненадійним в певних випадках, при цьому вебфреймворк Django такі засоби має за замовчуванням, тому перемагає за цим критерієм;

- підтримка: наявність великої кількості статей, відповідей на запитання дозволяє ефективніше працювати, створюючи проекти на основі такого рішення, при цьому за даним критерієм перемагає Django, адже має більшу спільноту розробників;

- наявність вбудованої адміністративної панелі: у даному випадку заплановано розробити проект з підтримкою окремо роботи адміністратора, створивши власний засіб, але наявність адміністративної панелі фреймворка Django дозволяє адміністратору керувати правами інших користувачів, до того ж на початкових етапах робота з даними може забезпечуватися одразу, без необхідності очікування повного завершення розроблення цієї сторони;

- масштаб проектів: за результатами дипломної кваліфікаційної роботи звичайно не забезпечується одразу підтримка великої кількості користувачів одразу, але використані рішення мають бути направлені на таку підтримку, на розширення кількості функціональних можливостей, а за цим критерієм кращим рішенням є фреймворк Django.

Таким чином за сукупністю критерії вибір було зроблено на користь Django.

Результати порівняння представлені в табл. 3.1.

Таблиця 3.1 – Порівняння фреймворків веброзробки для мови програмування Python

Характеристика	Django	Flask
Об'єктно-реляційне відображення	Так	Немає за замовчуванням, але можна встановити як розширення
Уникнення SQL-ін'єкцій	Так	Ні
Підтримка	Краща	Гірша
Наявність вбудованої адміністративної панелі	Так	Ні
Масштаб проєктів	Великі, середні	Малі

3.2 Моделювання роботи з програмою

Повний перелік функціональних можливостей програми, який повинен забезпечуватися і який був виділений за аналізом предметної області, включає:

- перегляд вакансії на працевлаштування;
- пошук вакансій на працевлаштування за назвою;
- розширений пошук вакансій;
- перегляд вакансій за посадою;
- перегляд вакансій за групою;
- пошук резюме;
- перегляд резюме за посадою;
- перегляд вакансій за посадою;

- перегляд користувачем своїх резюме;
- перегляд пропозицій на вакансію;
- перегляд власних пропозицій про працевлаштування;
- перегляд пропозицій за ярликом;
- перегляд наявних ярликів;
- перегляд наявних груп для упорядкування вакансій та резюме;
- перегляд переліку посад;
- перегляд компаній;
- перегляд вакансій за компанією;
- пошук компаній;
- перегляд компаній за містом;
- перегляд вакансій за країною;
- перегляд вакансій за містом;
- додавання і редагування за необхідності даних компаній;
- створення і редагування (в разі помилок) ярликів;
- створення і редагування пропозицій про працевлаштування;
- створення і редагування резюме;
- створення і редагування посад;
- створення і редагування вакансій;
- перегляд повних даних резюме;
- перегляд повних даних пропозицій;
- авторизація з наданням окремих облікових записів кандидатів на працевлаштування, представників компаній та адміністраторів.

Програму може використовувати звичайний користувач, який є кандидатом на займання вакансії щодо працевлаштування (рис. 3.1). Він може працювати над пошуком вакансій активно або пасивно. За активного варіанту він сам знаходить вакансії і надсилає пропозиції на зайняття відповідної посади. За пасивного варіанту користувач створює власне резюме і пропонує його для перегляду представникам компаній, які можуть звернутися до нього за цим резюме.

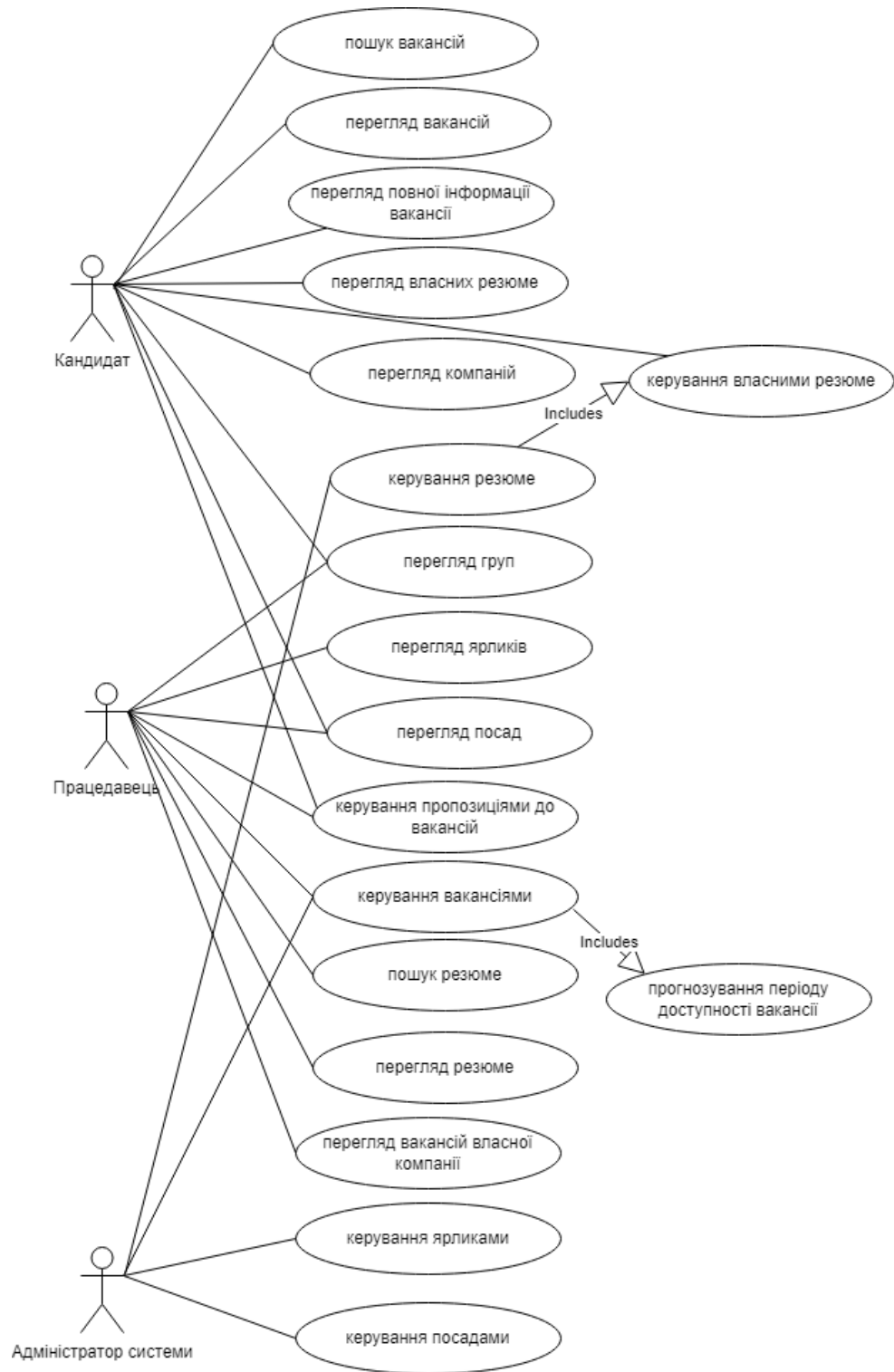


Рисунок 3.1 – Діаграма прецедентів

3.3 Проектування бази даних

Проектування БД реалізовано через визначення сутностей, які мають забезпечувати підтримку промодельованих сценаріїв використання програми.

Сутність Job визначає вакансію для працевлаштування на посаду в компанії. До складу полів таблиці входять:

- id – номер вакансії;
- title – назва вакансії;
- description – опис вакансії;
- education – освіта, яка вимагається за вакансією;
- skills – навички, які вимагаються за вакансією;
- published – дата публікації вакансії;
- company – компанія, якій належить вакансія;
- city – місто, в якому реалізується вакансія;
- experience – досвід, який вимагається для роботи за цією посадою;
- address – адреса місця роботи;
- phone – контактний номер телефону для зв'язку;
- employmenttype – тип працевлаштування, якому відповідає вакансія;
- minsalary – мінімальна заробітна платня за вакансією;
- maxsalary – максимальна заробітна платня за вакансією;
- activity – активність вакансії, що необхідно, щоб розуміти, чи можна подавати пропозиції за цією вакансією;
- ended – дата зняття вакансії;
- predictedtime – прогнозований час активності вакансії;
- updated – дата останнього оновлення вакансії;
- group – група, до якої належить вакансія;
- position – посада, до якої належить вакансія.

Таблиця Job з'єднана зв'язками з різними іншими таблицями, зокрема:

- таблиця Company визначає через відповідне поле в таблиці Job зв'язок типу один до багатьох, адже кожна компанія може мати цілий набір вакансій на сайті, при тому кожна вакансія може бути створена тільки однією компанією;

– таблиця City визначає через відповідне поле в таблиці Job зв'язок типу один до багатьох, адже у кожному місті може бути багато вакансій, але кожна вакансія може бути реалізована тільки в одному конкретному місті;

– таблиця Position визначає через відповідне поле в таблиці Job зв'язок типу один до багатьох, адже кожна вакансія може мати загальноприйнятну посаду, а кожна посада має багато вакансій;

– таблиця EmploymentType визначає через відповідне поле в таблиці Job зв'язок типу один до багатьох, адже у кожній вакансії може бути тільки один тип працевлаштування, але кожен тип може бути використаний в різних вакансіях;

– таблиця Group визначає через відповідне поле в таблиці Job зв'язок типу один до багатьох, адже у кожній групі представлені різні вакансії, але кожна вакансія належить тільки до однієї групи.

Сутність CompanyAccount представляє обліковий запис компанії. Він необхідний для роботи з резюме користувачів при активному пошуку, для роботи з пропозиціями при неактивному пошуку. До складу полів таблиці входять:

- id – номер облікового запису;
- login – логін користувача;
- password – пароль;
- company – компанія, якій належить обліковий запис.

Таблиця Company визначає через відповідне поле в таблиці CompanyAccount зв'язок типу один до багатьох, адже кожна компанія може мати цілий набір облікових записів для своїх представників для роботи з програмою, але кожен такий обліковий запис може бути відповідальний за роботу тільки з однією компанією.

Сутність Company представляє компанію, яка бажає знайти працівників.

До складу полів таблиці входять:

- id – номер компанії;

- title – назва компанії;
- phones – номери телефонів компанії;
- address – адреса, за якою розташована компанія;
- description – опис компанії;
- city – місто, в якому знаходиться компанія;
- workers – кількість працівників компанії;
- image – зображення з логотипом або будь-яким графічним представленням компанії.

Таблиця City визначає через відповідне поле в таблиці Company зв'язок типу один до багатьох, адже кожне місто може бути представлене в програмі різними компаніями, при цьому одна компанія представляється в програмі тільки одним містом, де розташовано її головний офіс. При цьому міста зазначаються для кожної вакансії, що дозволяє за наявності різних відділень у компанії надати їм можливість визначати власні вакансії.

Сутність City представляє місто, в якому знаходяться компанії, проживають працівники.

До складу полів таблиці входять:

- id – номер міста;
- title – назва міста.

Сутність Country представляє країну, в якій знаходиться місто.

До складу полів таблиці входять:

- id – номер країни;
- title – назва країни.

Сутність Proposition представляє пропозицію, висунуту працівником щодо працевлаштування за вакансією.

До складу полів таблиці входять:

- id – номер пропозиції;
- job – вакансія, щодо якої висунуто пропозицію;
- useraccount – обліковий запис користувача, який висунув пропозицію;

- published – дата публікації пропозиції;
- activity – стан, в якому знаходиться пропозиція;
- companytext – текстове повідомлення від компанії щодо даної пропозиції, яке може бути пропозицією щодо проходження інтерв'ю;
- usertext – текстове повідомлення від користувача, який висунув пропозицію;
- interview – дата інтерв'ювання здобувача посади;
- decision – текстове повідомлення про рішення щодо даної пропозиції.

Таблиця Job визначає через відповідне поле в таблиці Proposition зв'язок типу один до багатьох, адже кожна вакансія може мати багато пропозицій від кандидатів, при цьому кожна пропозиція стосується тільки однієї конкретної вакансії.

Таблиця UserAccount визначає через відповідне поле в таблиці Proposition зв'язок типу один до багатьох, адже кожен користувач може висувати різні пропозиції щодо різних вакансій, але кожна пропозиція повинна мати одного свого автора.

Сутність UserAccount представляє обліковий запис користувача. До складу полів таблиці входять:

- id – номер облікового запису;
- login – логін користувача;
- password – пароль;
- bday – дата народження користувача;
- education – освіта користувача: необхідна для того, щоб використовувати її в резюме, проте в кожному резюме вона буде однаковою, змінювати її під конкретне резюме не можна, оскільки користувач має вказати повну інформацію щодо себе;
- image – зображення з фотографією користувача.

Сутність Resume представляє резюме користувача. До складу полів таблиці входять:

- id – номер резюме;
- useraccount – обліковий запис користувача, який створив дане резюме;
- email – електронна пошта користувача;
- phone – номер телефону користувача;
- title – назва резюме;
- experience – досвід користувача: досвід у кожному резюме користувач може вказуватися по-різному, щоб акцентувати увагу на досвід щодо конкретної посади;
- published – дата публікації резюме;
- activity – стан, в якому знаходиться резюме;
- city – місто, в якому бажає працювати користувач: виділено окремо в резюме, а не розташовується в обліковому запису, щоб дозволити користувачу вказувати на можливість переїзду в певне місто для виконання такої роботи;
- group – група, до якої належить резюме;
- position – посада, на яку претендує автор резюме.

Таблиця UserAccount визначає через відповідне поле в таблиці Resume зв'язок типу один до багатьох, адже кожен користувач може створити різні резюме під різні посади, при цьому автором резюме може бути тільки один користувач.

Таблиця Job визначає через відповідне поле в таблиці Resume зв'язок типу один до багатьох, адже кожна вакансія може мати багато пропозицій про працевлаштування, проте кожна пропозиція стосується тільки однієї вакансії.

Результуюча схема БД, що визначає її структуру у відповідності з описаними таблицями та зв'язками між цими таблицями, представлена на рис. 3.2.

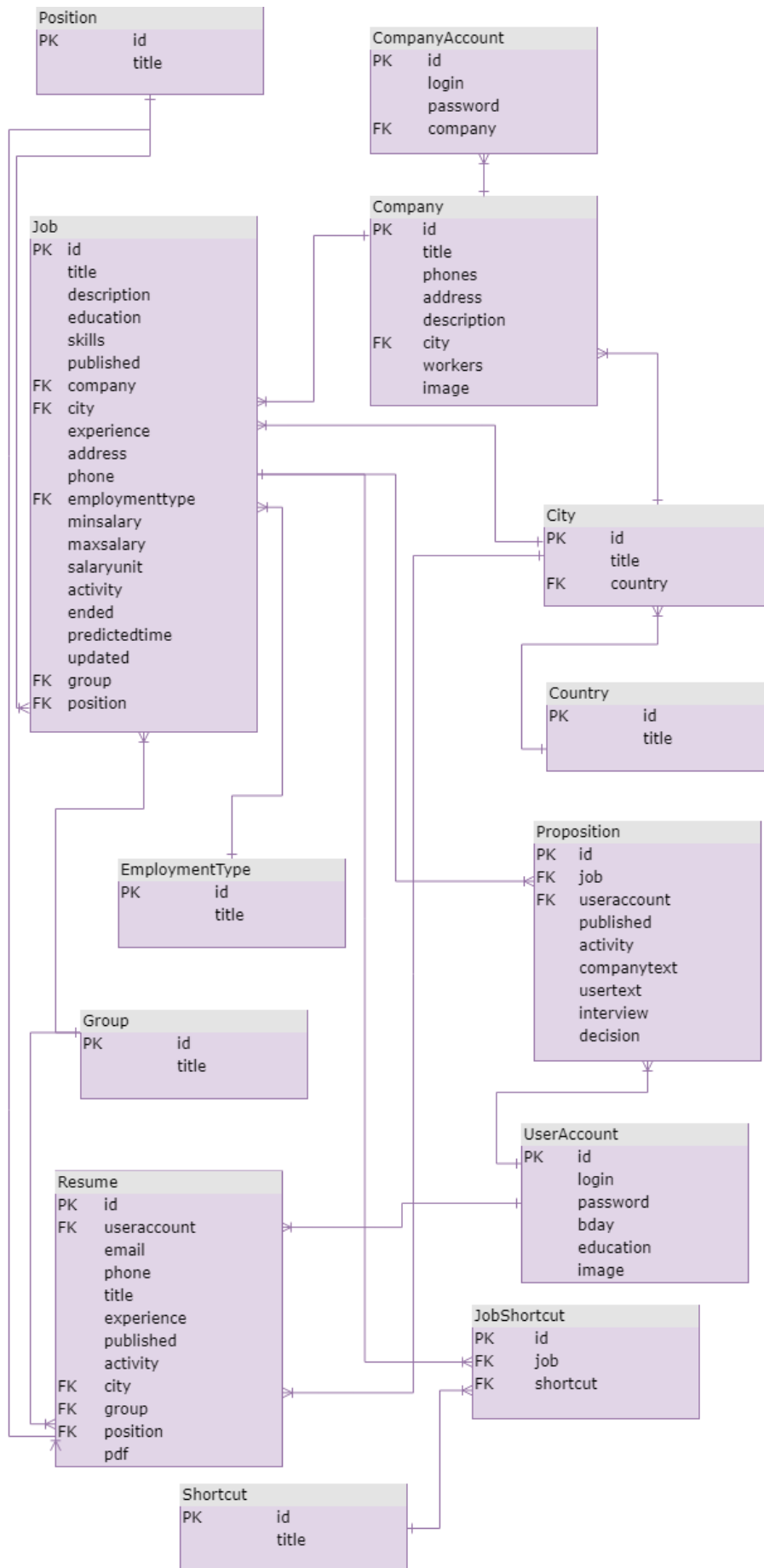


Рисунок 3.2 – Схема БД

Сутність EmploymentType представляє тип працевлаштування.

До складу полів таблиці входять:

- id – номер типу працевлаштування;
- title – назва типу працевлаштування.

Таблиця Position визначає через відповідне поле в таблиці Resume зв'язок типу один до багатьох, адже кожне резюме може мати загальноприйнятну посаду, а кожна посада має багато резюме.

Сутність Shortcut представляє ярлики, які можуть встановлюватися вакансії для виділення її серед інших вакансій. До складу полів таблиці входять:

- id – номер ярлика;
- title – назва ярлика.

Сутність Group представляє групи, в які об'єднуються вакансії та резюме для їх категоризації. До складу полів таблиці входять:

- id – номер групи;
- title – назва групи.

Таблиця JobShortcut необхідна для того, щоб реалізувати зв'язок багато-до-багатьох між таблицями JobShortcut, який виникає через те, що кожна вакансія може мати цілий набір ярликів, але і кожен ярлик може бути встановлений різним вакансіям.

Сутність Position представляє посаду. До складу полів таблиці входять:

- id – номер посади;
- title – назва посади.

3.4 Структура програмного забезпечення

Структуру програми (рис. 3.3) більшою мірою визначило використання вебфреймворку Django. Тому до складу входить стандартна частина Django у складі проєкту – djangoapp, а безпосередньо розроблені

рішення представлені в застосунку `jacapp`, де визначено структуру вебзастосунку, та бібліотеці `predictor`, де визначено засоби роботи з описаним методом у розділі 2, що включають засоби роботи з текстом у файлі `texter.py` та засоби роботи з деревами рішень в файлі `tree.py`. Шаблони вебзастосунку не було деталізовано окремими файлами, оскільки їх була розроблена велика кількість, основна частина їх описана в розділі 4 після виконання реалізації.

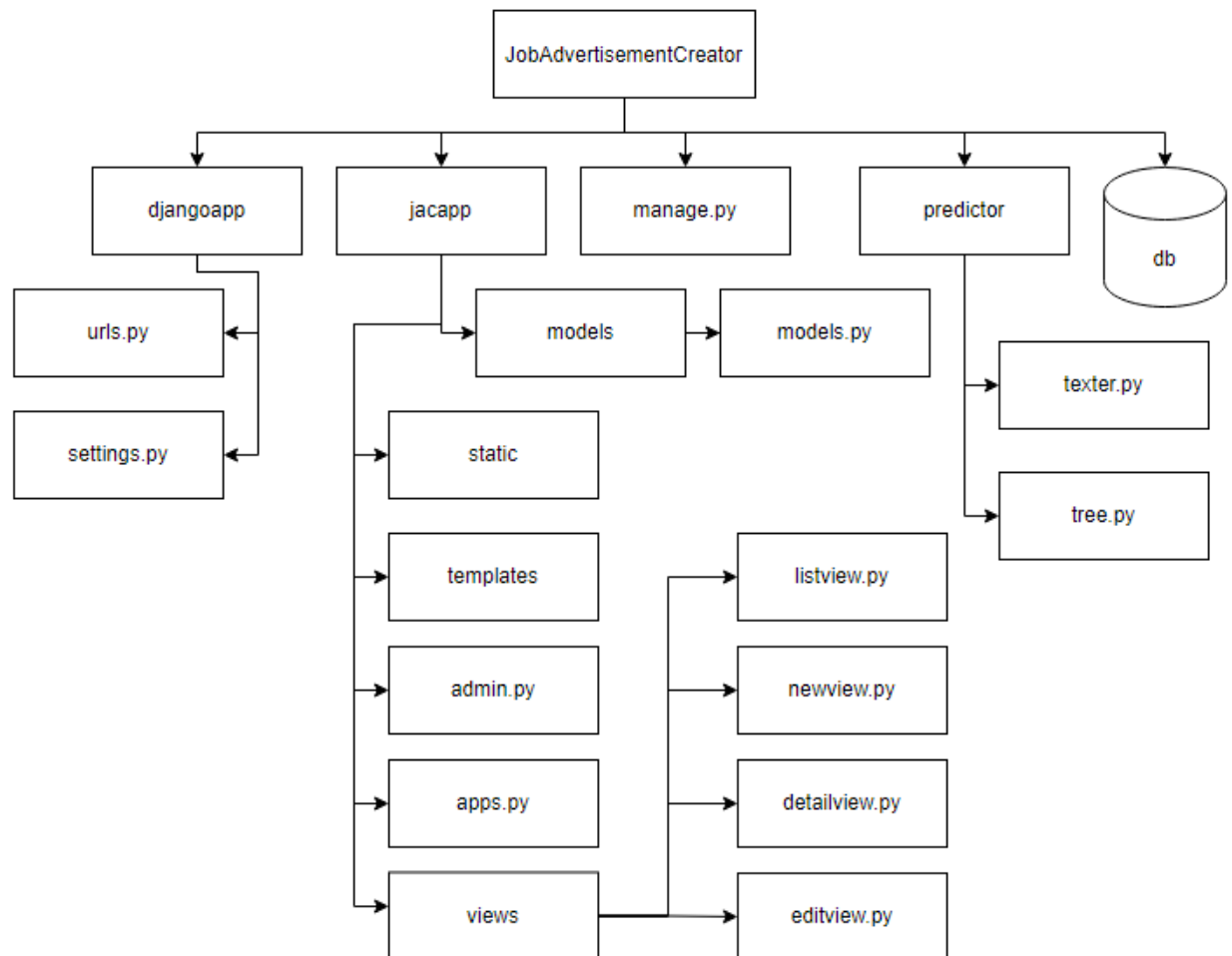


Рисунок 3.3 – Структурна схема програми

3.5 Висновки за розділом 3

Для розробки програми вирішено використовувати мову програмування Python та вебфреймворк Django, вибір якого обґрунтовано шляхом порівняння з вебфреймворком Flask.

Виконано моделювання роботи користувачів з програмою за допомогою діаграм прецедентів. Визначено структуру БД.

Визначено структуру програми за допомогою структурної схеми.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування системи

Схема функціонування програми представлена на рис. 4.1.

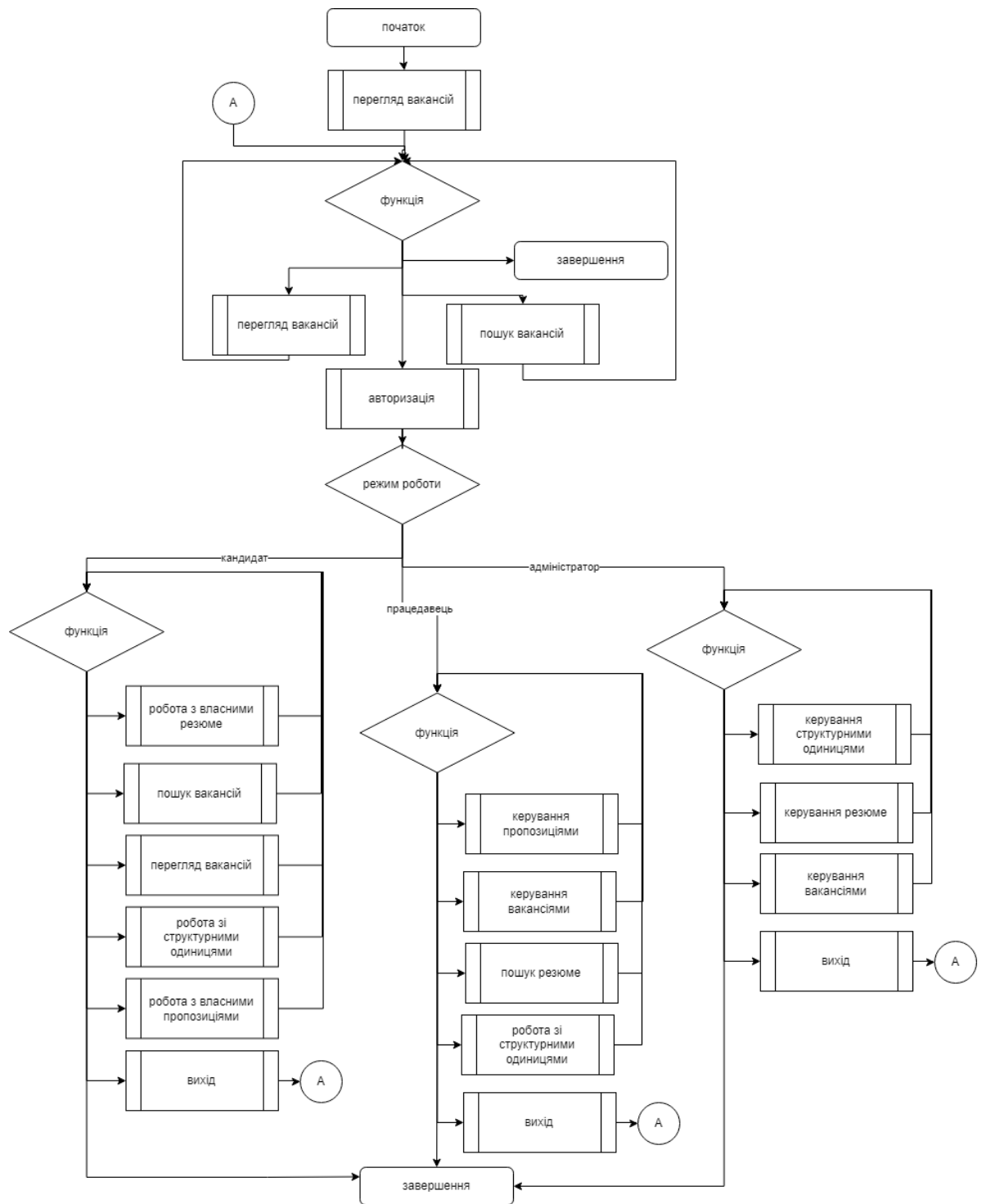


Рисунок 4.1 – Схема функціонування програми

Робота програми розпочинається з перегляду вакансій, після чого користувач може або виконати пошук вакансій, або переглянути їх в іншому режимі, або завершити роботу з програмою, або авторизуватися і отримати можливість розширеної роботи з програмою. Далі в залежності від ролі користувач працює або в режимі кандидата, обираючи з меню одну з доступних функцій, кожна з яких включає в себе інші функції, або в режимі працедавця, або в режимі адміністратора.

4.2 Опис розроблених класів програми

Використання фреймворку Django значно структурувало та багато в чому спростило процес розробки. Це виражено як на рівні організації роботи з даними, так і на рівні реалізації відповідної функціональності програми. Розглянемо докладніше рішення, які були створені на відповідних рівнях, та те, на яких засобах Django ці рішення засновані. Шаблони були реалізовані з використанням шаблону [16].

Для реалізації роботи з даними було створено класи моделей, що відповідають за підтримку об'єктно-реляційного відображення, що дозволяє створити БД в результаті на основі коду.

Класи моделей включають наступні:

- клас `Job` визначає структуру вакансії на працевлаштування, для якої визначено метод `__str__(self)`, який повертає текстове представлення об'єкта, сформоване з назви вакансії, що потрібно для зручного відображення вакансій на вебсторінках, і метод `get_absolute_url(self)`, який повертає повний шлях вебсторінки, що представляє даний об'єкт вакансії, також до складу класу включено атрибут `shortcuts`, який визначає масив ярликів, які співставлені з даною вакансією;

- клас `Proposition` визначає структуру пропозиції про працевлаштування за вакансією і містить у своїй структурі метод

`get_absolute_url(self)`, який повертає повний шлях вебсторінки, що представляє даний об'єкт пропозиції;

- клас `CompanyAccount` визначає обліковий запис користувача, який є представником компанії, шляхом наслідування класу `AbstractUser`, який підтримує створення базового облікового запису Django, що дозволяє частину атрибутів наслідувати від базового класу;

- клас `Company` визначає об'єкти компаній, для яких підтримується метод `get_absolute_url(self)`, який повертає повний шлях вебсторінки, що представляє даний об'єкт компанії, формуючи його за участю номера компанії;

- клас `City` визначає об'єкти міст, для яких підтримується метод `get_absolute_url(self)`, який повертає повний шлях вебсторінки, що представляє місто;

- клас `Country` визначає об'єкти країн, для яких підтримується метод `get_absolute_url(self)`, який повертає повний шлях вебсторінки, що представляє країну;

- клас `EmploymentType` визначає об'єкти типу працевлаштування;

- клас `UserAccount` визначає об'єкти облікових записів звичайних користувачів, які є кандидатами на працевлаштування, створений шляхом наслідування класу `AbstractUser`;

- клас `Resume` визначає об'єкти резюме, для яких підтримується метод `get_absolute_url(self)`, який повертає повний шлях вебсторінки, що представляє резюме;

- клас `Shortcut` визначає об'єкти ярликів;

- клас `Group` визначає об'єкти груп вакансій і резюме;

- клас `Position` визначає об'єкти посад.

Наступні класи реалізують види, які забезпечують визначення даних, якими має бути заповнено шаблон для реалізації однієї з функцій програми.

Клас `JobCompanyView` формує вид для відображення повної інформації про вакансію для працевлаштування для представника компанії.

Представник компанії на відміну від звичайного виду перегляду повної інформації вакансії отримає доступ до переліку пропозицій, які були висунуті кандидатами на дану вакансію. Для заповнення шаблону `adjob.html` даними використовується метод `get_context_data(self, **kwargs)`, який в контекст даних вносить повну інформацію об'єкта з номером, переданим в якості параметра роботи метода, а також додає всі пропозиції, висунуті на дану вакансію, тобто які мають в якості атрибута `job_id` той же номер, переданий через параметр методу.

Цей і описані нижче класи створені на основі класу `DetailView`, який забезпечує вид деталізації інформації про об'єкт.

Клас `JobView` формує вид для відображення інформації про вакансію для працевлаштування для кандидата на працевлаштування. Кандидату не має відображатися перелік пропозицій за вакансією, на відміну від класу `JobCompanyView`. Відображення даних реалізується через шаблон `job.html`.

Клас `ResumeView` формує вид для відображення інформації про резюме кандидата на працевлаштування. Для заповнення шаблону `resume.html` даними використовується метод `get_context_data(self, **kwargs)`, який в контекст даних вносить повну інформацію об'єкта резюме з номером, переданим в якості параметра роботи метода.

Клас `PropositionView` формує вид для відображення інформації про пропозицію на працевлаштування від кандидата на вакансію. Для заповнення шаблону `proposition.html` даними використовується метод `get_context_data(self, **kwargs)`, який в контекст даних вносить повну інформацію об'єкта пропозиції з номером, переданим в якості параметра роботи метода.

Клас `CompanyView` формує вид для відображення інформації про компанію. Для заповнення шаблону `company.html` даними використовується метод `get_context_data(self, **kwargs)`, який в контекст даних вносить повну інформацію об'єкта компанії з номером, переданим в якості параметра роботи метода.

Наступні класи організовані наслідуванням базового класу `CreateView`, що підтримує роботу з формою створення нового об'єкту на основі відповідної моделі даних. Усі наступні класи потребують авторизації.

Клас `NewJobView` формує вид з формою створення нової вакансії. Для цього визначені співвідношення між елементами форми та полями моделі. Метод `form_valid(self, form)` створений шляхом перевизначення відповідного базового методу класу `CreateView`, при цьому в якості дати створення вакансії вноситься поточна дата, в якості форми активності визначається нульове значення (тобто вакансія є активною). Підтримка роботи з даними реалізується через шаблон `newjob.html`.

Клас `NewCompanyView` формує вид з формою створення нової компанії. Для цього визначені співвідношення між елементами форми та полями моделі `Company` на основі атрибуту `fields`. Підтримка роботи з даними реалізується через шаблон `newcompany.html`.

Клас `NewPositionView` формує вид з формою створення нової посади. Для цього визначені співвідношення між елементами форми та полями моделі `Position` на основі атрибуту `fields`. Підтримка роботи з даними реалізується через шаблон `newcompany.html`.

Клас `NewPropositionView` формує вид з формою створення нової пропозиції про працевлаштування за вакансією. Для цього визначені співвідношення між елементами форми та полями моделі `Proposition`. Метод `form_valid(self, form)` створений шляхом перевизначення відповідного базового методу класу `CreateView`, при цьому в якості дати створення пропозиції вноситься поточна дата, в якості форми активності визначається нульове значення (тобто пропозиція є активною). Підтримка роботи з даними реалізується через шаблон `newproposition.html`.

Клас `NewShortcutView` формує вид з формою створення нового ярлика, що використовується для спеціального позначення вакансій про працевлаштування. Для цього визначені співвідношення між елементами

форми та полями моделі Shortcut на основі атрибуту fields. Підтримка роботи з даними реалізується через шаблон newshortcut.html.

Клас NewResumeView формує вид з формою створення нового резюме. Для цього визначені співвідношення між елементами форми та полями моделі Resume. Метод form_valid(self, form) створений шляхом перевизначення відповідного базового методу класу CreateView, при цьому в якості дати створення резюме вноситься поточна дата, в якості форми активності визначається нульове значення (тобто резюме є активною). Підтримка роботи з даними реалізується через шаблон newresume.html.

Наступні класи організовані наслідуванням базового класу UpdateView, що підтримує роботу з формою редагування існуючого об'єкту на основі відповідної моделі даних. Усі наступні класи потребують авторизації.

Клас JobEditView формує вид з формою редагування існуючої вакансії. Для цього визначені співвідношення між елементами форми та полями моделі Job на основі атрибуту fields. Метод form_valid(self, form) створений шляхом перевизначення відповідного базового методу класу UpdateView, при цьому в якості дати останнього оновлення вакансії вноситься поточна дата. Підтримка роботи з даними реалізується через шаблон jobedit.html.

Клас EditPositionView формує вид з формою редагування існуючої посади. Для цього визначені співвідношення між елементами форми та полями моделі Position на основі атрибуту fields. Підтримка роботи з даними реалізується через шаблон editposition.html.

Клас ResumeEditView формує вид з формою редагування існуючого резюме. Для цього визначені співвідношення між елементами форми та полями моделі Resume на основі атрибуту fields. Підтримка роботи з даними реалізується через шаблон resumedit.html.

Клас UserPropositionEditView формує вид з формою редагування існуючої пропозиції кандидатом на працевлаштування. Для цього визначені

співвідношення між елементами форми та полями моделі `Proposition` на основі атрибута `fields`. Підтримка роботи з даними реалізується через шаблон `proposedit.html`.

Клас `CompanyPropositionEditView` формує вид з формою редагування існуючої пропозиції представником компанії роботодавця. Для цього визначені співвідношення між елементами форми та полями моделі `Proposition` на основі атрибута `fields`. Підтримка роботи з даними реалізується через шаблон `coproposedit.html`.

Клас `EditShortcutView` формує вид з формою редагування існуючого ярлика. Для цього визначені співвідношення між елементами форми та полями моделі `Shortcut` на основі атрибута `fields`. Підтримка роботи з даними реалізується через шаблон `editshortcut.html`.

Клас `EditCompanyView` формує вид з формою редагування існуючої компанії. Для цього визначені співвідношення між елементами форми та полями моделі `Company` на основі атрибута `fields`. Підтримка роботи з даними реалізується через шаблон `editcompany.html`.

Наступні класи організовані наслідуванням базового класу `ListView`, що підтримує роботу з переліками даних.

Клас `PositionJobsView` формує вид перегляду переліку вакансій за посадою на основі шаблону `positionjobs.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі вакансії, що належать посаді з номером, який передається через параметр `position` GET-запиту. Вносяться тільки активні (не зняті, не закриті) вакансії.

Клас `CityJobsView` формує вид перегляду переліку вакансій за заданим містом на основі шаблону `cityjobs.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі вакансії, що належать місту з номером, який передається через параметр `city` GET-запиту. Вносяться тільки активні вакансії.

Клас `CompanyJobsView` формує вид перегляду переліку вакансій за заданою компанією на основі шаблону `companyjobs.html`. Метод

`get_queryset(self, *args, **kwargs)` вносить всі вакансії, що належать компанії з номером, який передається через параметр `company` GET-запиту.

Клас `GroupJobsView` формує вид перегляду переліку вакансій за заданою групою на основі шаблону `groupjobs.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі вакансії, що належать групі з номером, який передається через параметр `group` GET-запиту. Вносяться тільки активні вакансії. Сортування реалізується за датою публікації вакансії, починаючи з останньої.

Клас `CountryJobsView` формує вид перегляду переліку вакансій за заданою країною на основі шаблону `countryjobs.html`. Даний вид необхідний, щоб користувач, який згоден на переїзд, міг вивчити вакансії з різних міст. Метод `get_queryset(self, *args, **kwargs)` вносить всі вакансії, що реалізуються в країні з номером, який передається через параметр `country` GET-запиту. Вносяться тільки активні вакансії. Сортування реалізується за датою публікації вакансії, починаючи з останньої.

Клас `SearchJobsView` формує вид перегляду результатів пошуку вакансій за її назвою або частиною такої назви на основі шаблону `searchjobs.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі вакансії, що в назві містять значення, задане параметром `title` GET-запиту. Вносяться тільки активні вакансії. Сортування реалізується за датою публікації вакансії, починаючи з останньої.

Клас `AdvancedSearchJobsView` формує вид перегляду результатів розширеного пошуку вакансій на основі шаблону `advsearchjobs.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі вакансії, що відповідають параметрам, переданим через GET-запит:

- `title` – частина назви вакансії;
- `company` – компанія, якою створено вакансію;
- `city` – місто, в якому реалізується вакансія;
- `position` – посада, яка пропонується в вакансії;
- `group` – група, якій належить вакансія;

– `emptytype` – тип працевлаштування (повна зайнятість, неповний робочий день тощо);

– `minsal` – мінімальна заробітна платня, на яку розраховує кандидат;

– `maxsal` – максимальна заробітна платня, на яку розраховує кандидат.

Вносяться тільки активні вакансії. Сортування реалізується за параметром `way` GET-запиту.

Клас `CityCompaniesView` формує вид перегляду переліку компаній за містом, в якому вони розташовані, на основі шаблону `citycompanies.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі компанії, що знаходяться в місті з номером, який передається через параметр `city` GET-запиту. Вносяться тільки активні вакансії.

Клас `CompaniesView` формує вид перегляду переліку всіх наявних компаній на основі шаблону `companies.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі компанії з БД.

Клас `PositionsView` формує вид перегляду переліку всіх наявних посад на основі шаблону `positions.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі посади з БД.

Клас `GroupsView` формує вид перегляду переліку всіх наявних груп вакансій і резюме на основі шаблону `groups.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі групи з БД.

Клас `PositionResumesView` формує вид перегляду переліку резюме за заданою посадою на основі шаблону `positionresumes.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі резюме, що стосуються посади з номером, який передається через параметр `position` GET-запиту. Вносяться тільки активні резюме. Сортування реалізується за датою публікації резюме, починаючи з останньої.

Клас `ShortcutsView` формує вид перегляду переліку всіх наявних ярликів на основі шаблону `shortcuts.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі ярлики з БД.

Клас `ShortcutJobsView` формує вид перегляду переліку вакансій за заданим ярликом на основі шаблону `shortjobs.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі вакансії, що відмічені ярликом з номером, який передається через параметр `shortcut` GET-запиту. Вносяться тільки активні резюме. Сортування реалізується за датою публікації резюме, починаючи з останньої.

Клас `UserPropositionsView` формує перелік пропозицій користувача, який авторизований на даний момент. Метод `get_queryset(self, *args, **kwargs)` вносить всі пропозиції, що створені користувачем, параметри авторизації якого знаходяться в запиті. Вносяться всі пропозиції, без врахування стану. Сортування реалізується за датою публікації пропозиції, починаючи з останньої.

Клас `UserResumesView` формує перелік резюме користувача. Метод `get_queryset(self, *args, **kwargs)` вносить всі резюме, що створені користувачем, параметри авторизації якого знаходяться в запиті. Вносяться всі резюме, без врахування стану. Сортування реалізується за датою публікації резюме, починаючи з останнього.

Клас `AdvancedSearchResumesView` формує вид перегляду результатів розширеного пошуку резюме на основі шаблону `advsearchresumes.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі резюме, що відповідають параметрам, переданим через GET-запит:

- `title` – частина назви резюме;
- `city` – місто, в якому бажає працювати кандидат;
- `position` – посада, яку він готовий займати;
- `group` – група, якій належить резюме.

Вносяться тільки активні резюме. Сортування реалізується за параметром `way` GET-запиту.

Клас `CitySearchView` формує вид перегляду результатів пошуку міст за їх назвою або частиною такої назви на основі шаблону `searchcities.html`.

Метод `get_queryset(self, *args, **kwargs)` вносить всі міста, що в назві містять значення, задане параметром `title` GET-запиту.

Клас `CompanySearchView` формує вид перегляду результатів пошуку компаній за їх назвою або частиною такої назви на основі шаблону `searchcompanies.html`. Метод `get_queryset(self, *args, **kwargs)` вносить всі компанії, що в назві містять значення, задане параметром `title` GET-запиту.

4.3 Висновки за розділом 4

У розділі виконано опис алгоритму функціонування програми, що заснований на розподілі режимів використання програми різними групами користувачів, включаючи кандидатів на працевлаштування, представників роботодавців та адміністраторів. Також в розділі детально описано створені для реалізації програми класи, що забезпечують роботу з даними, в тому числі визначення їх структури та формування їх під відображення у відповідних шаблонах.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для формування оголошень про працевлаштування, пошуку вакансій на ринку, виставлення власних резюме, пошуку серед наявних резюме. Тому програма може використовуватися як кандидатами, так і представниками компаній.

5.2 Умови виконання програми

До умов виконання програми, які апаратно мають забезпечуватися на сервері, належать:

- процесор, який має 4 ядра з тактовою частотою в 2,66 ГГц;
- оперативна пам'ять обсягом у 8 Гб;
- жорсткий диск з загальним обсягом не менше 8 Гб, призначений для серверного використання.

Програма потребує також наявності встановлених інтерпретатора Python, пакета scikit-learn і всіх супутніх для його роботи пакетів та системи керування БД MySQL.

5.3 Результати роботи програми

Програму було протестовано щодо виконання функціональних вимог, які передбачались у розділі 3. За результатами було виявлено виконання заявлених функціональних можливостей.

Взаємодія користувача з програмою передбачає перехід між варіантами для кандидата і представника роботодавця. Перехід між ними відбувається за допомогою пункту меню, який відображається лівіше за кнопку входу або виходу до власного облікового запису (рис. 5.1).

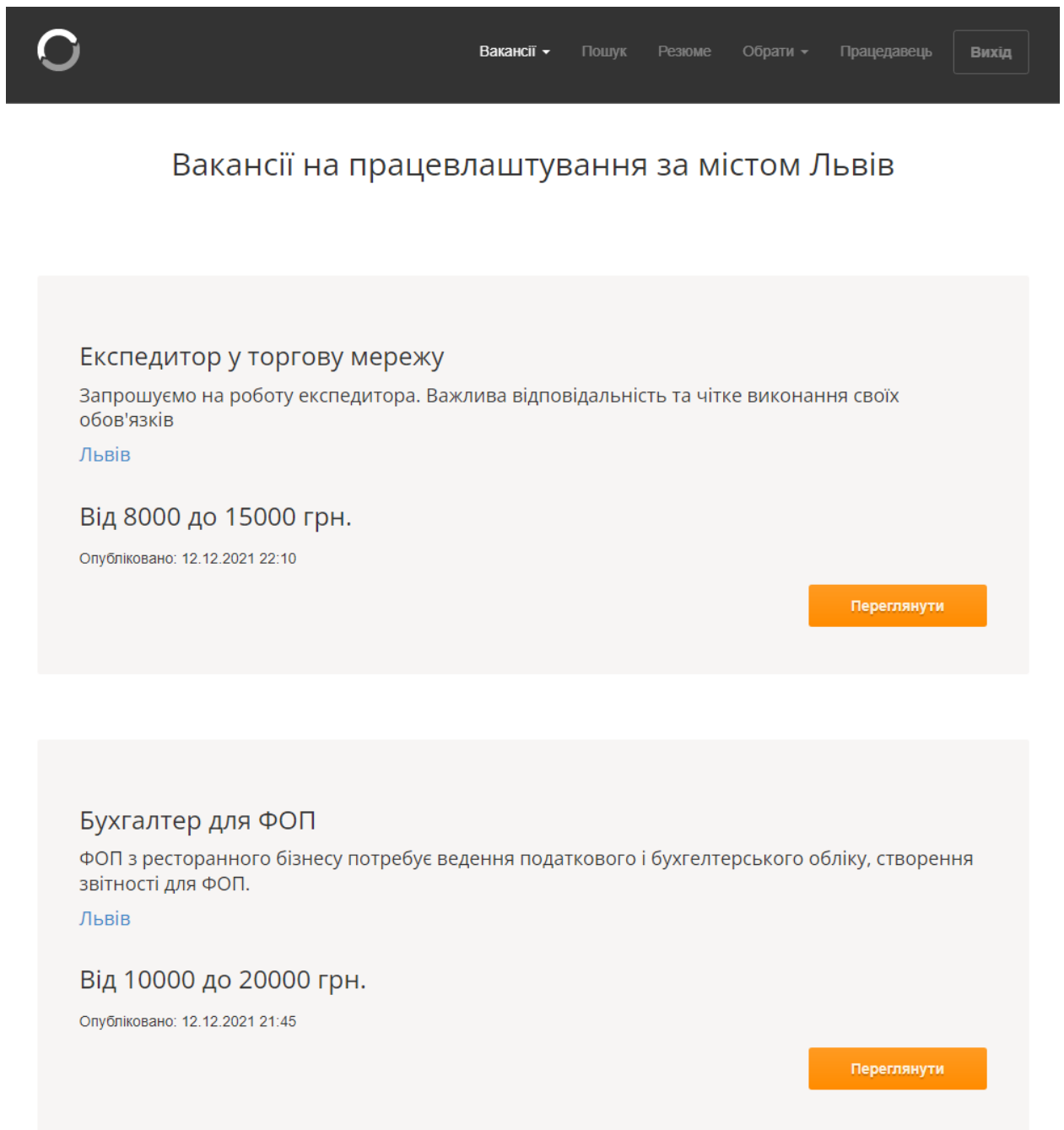
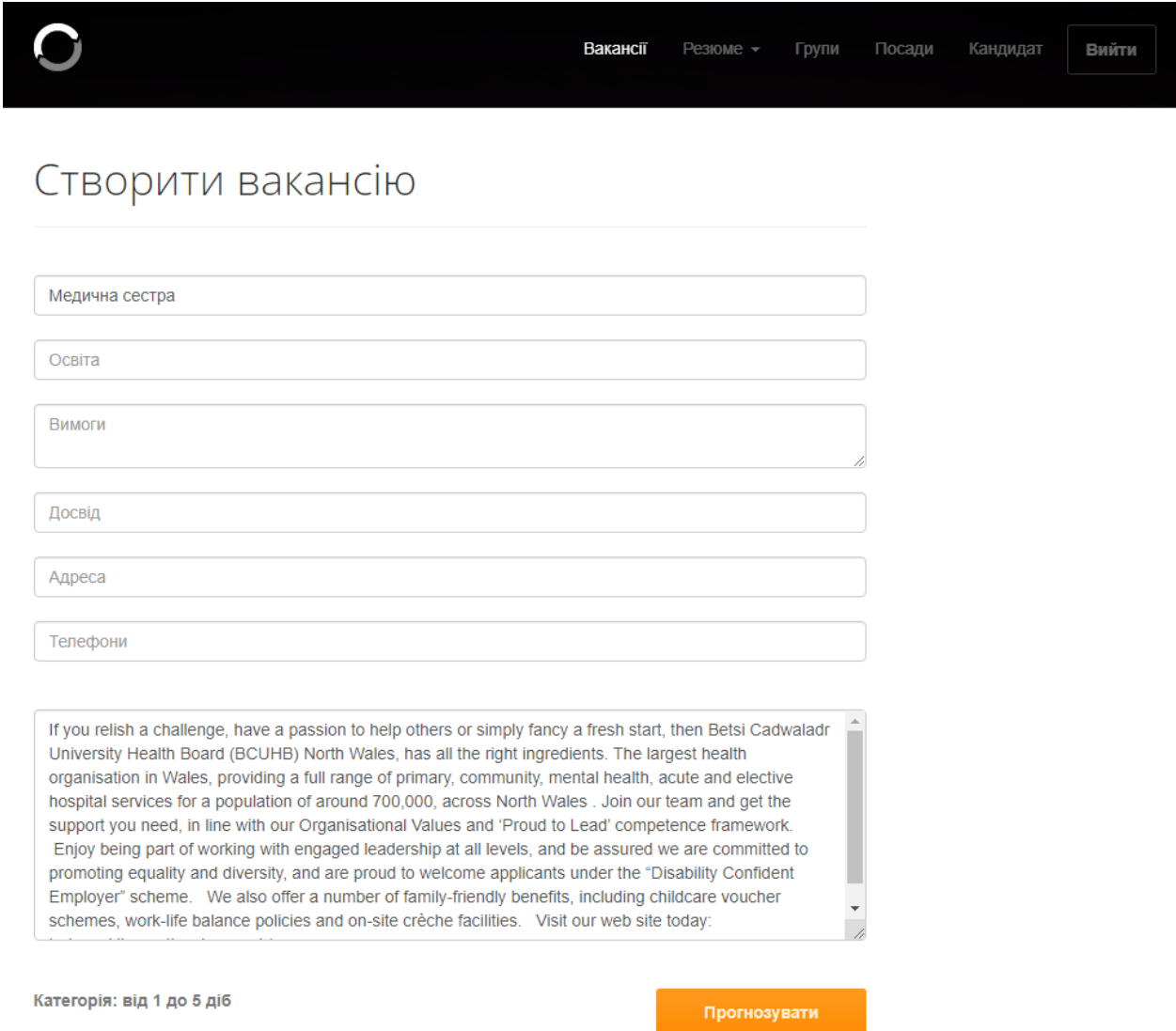


Рисунок 5.1 – Перегляд вакансій

Деякі пункти меню мають підпункти для деталізації режимів. Наприклад, пункт меню «Обрати» дозволяє обрати перелік груп, посад або компаній. На кожний такий вибір відображається сторінка з переліком всіх наявних відповідно груп, посад і компаній, наявних у програмі. Можна також використати режим пошуку. Можна шукати потрібну компанію, а можна задати широкий набір критеріїв для пошуку вакансій. Для такого вибору передбачено кнопки на відповідних сторінках.

Під час вибору конкретного значення за однією з базових одиниць: наприклад, міста вакансії, відображається перелік вакансій або резюме, відсортований за датою публікації. Під час виконання пошуку відображається перелік результатів, напрям сортування яких можна обрати натисканням кнопки. Ярлики відображаються при відкритті сторінки конкретної вакансії, за їх назвою можна перейти до вакансій, помічених цим ярликом. Під час створення вакансії працедавцем під полем опису вакансії розташована кнопка «Прогнозувати» (рис. 5.2), що дозволяє виконати прогнозування на основі запропонованого методу, а результати у вигляді категорії часу будуть розташовані лівіше цієї кнопки.



Вакансії Резюме ▾ Групи Посади Кандидат Вийти

Створити вакансію

Медична сестра

Освіта

Вимоги

Досвід

Адреса

Телефони

If you relish a challenge, have a passion to help others or simply fancy a fresh start, then Betsi Cadwaladr University Health Board (BCUHB) North Wales, has all the right ingredients. The largest health organisation in Wales, providing a full range of primary, community, mental health, acute and elective hospital services for a population of around 700,000, across North Wales. Join our team and get the support you need, in line with our Organisational Values and 'Proud to Lead' competence framework. Enjoy being part of working with engaged leadership at all levels, and be assured we are committed to promoting equality and diversity, and are proud to welcome applicants under the "Disability Confident Employer" scheme. We also offer a number of family-friendly benefits, including childcare voucher schemes, work-life balance policies and on-site crèche facilities. Visit our web site today:

Категорія: від 1 до 5 діб

Прогнозувати

Рисунок 5.2 – Створення нової вакансії

Деякі пункти меню в залежності від ролі користувача виконують різні дії. Так, пункт «Вакансії» у працедавця дозволяє керувати вакансіями (рис. 5.3), виставленими компанією, а у кандидата дозволяє переглянути вакансії за однією з одиниць.

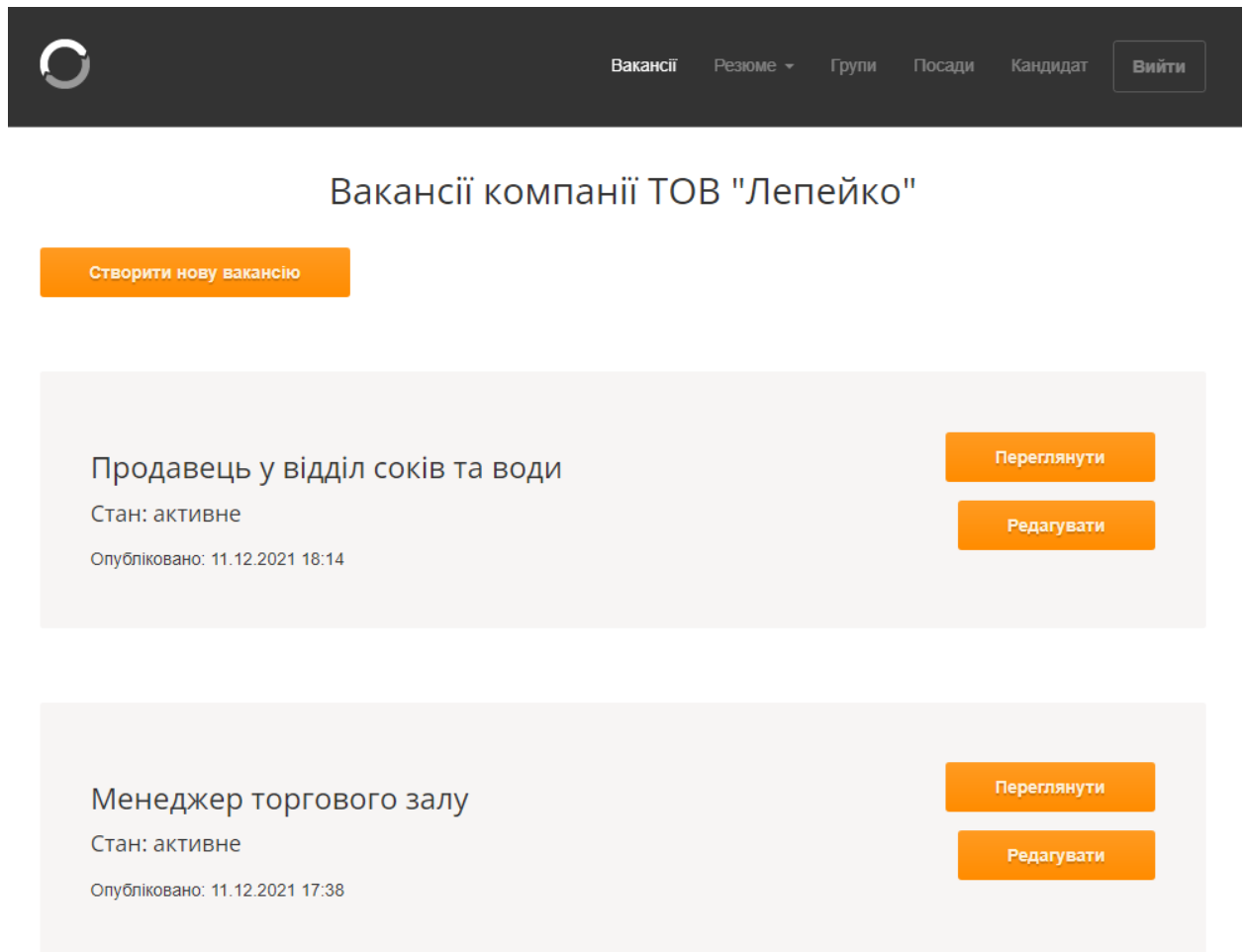


Рисунок 5.3 – Керування вакансіями компанії

5.4 Експериментальне дослідження

Експериментальне дослідження відбувалось за допомогою вибірки даних [17], що характеризувала вакансії для працевлаштування у Великобританії. Усі оголошення були розділені на окремі групи, серед них було виділено «Інженерію», де потреба виникає в інженерах, спеціалістах в сфері інформаційних технологій, та «Медицину».

Отриману у відсотках точність класифікації для кожної підвибірки було зведено до табл. 5.1.

Таблиця 5.1 – Результати експериментального дослідження

Група вакансій	Точність класифікації
Інженерія	82,5
Медицина	79,2

Точність класифікації дещо коливається на різних вибірках даних, проте дозволяє отримати достатньо точний прогноз як для першої, так і для другої підвибірки.

5.5 Висновки за розділом 5

Виконано огляд результатів роботи програми у вигляді відповідних екранних форм, які демонструють відповідність програми поставленим початково вимогам. Експериментальне дослідження, приведене в даному розділі, продемонструвало достатньо якісні результати відносно точності класифікації тестів на 5 заданих категорій.

ВИСНОВКИ

У дипломній кваліфікаційній роботі розроблено програмне забезпечення формування і пошуку оголошень про працевлаштування для надання інформації для прийняття рішень про вибір конкретного формулювання вакансії на основі прогнозування періоду її доступності. За результатами виконання основних завдань роботи було отримано наступні результати.

Виконано огляд проблем, які виникають під час пошуку вакансій, та проведено аналіз програмних аналогів рішення, що розробляється. В результаті було виявлено, що вони не мають засобів прогнозування періоду доступності вакансії на працевлаштування, що стало особливістю програми, яка розроблялась. Розглянуто способи вирішення даної проблеми і в результаті запропоновано власний метод розв'язання проблеми.

Для розробки програми вирішено використовувати мову програмування Python та вебфреймворк Django, вибір якого обґрунтовано шляхом порівняння з вебфреймворком Flask. Виконано моделювання роботи користувачів з програмою за допомогою диграм прецедентів. Визначено структуру БД. Визначено структуру програми за допомогою структурної схеми.

Виконано опис алгоритму функціонування програми, що заснований на розподілі режимів використання програми різними групами користувачів, включаючи кандидатів на працевлаштування, представників роботодавців та адміністраторів. Детально описано створені для реалізації програми класи, що забезпечують роботу з даними, в тому числі визначення їх структури та формування їх під відображення у відповідних шаблонах.

Розроблено програмне забезпечення формування і пошуку оголошень про працевлаштування, що надає кандидатам на працевлаштування, роботодавцям можливість формувати оголошення про працевлаштування з наданням можливості прогнозування періоду доступності такої вакансії,

виконувати пошук вакансій на ринку, виставляти власні резюме, виконувати пошук серед наявних резюме.

Наукова новизна роботи полягає в методі прогнозування періоду доступності вакансії на працевлаштування, який відзначається застосуванням показника tf-idf та методу опорних векторів, визначаючи таким чином часову категорію, що дозволяє приймати в подальшому рішення щодо вибору конкретного формулювання вакансії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Job Market Is Going Crazy And It's Going To Get Worse. How Do We Recruit? [Electronic resource]. – Access mode : <https://joshbersin.com/2021/11/the-job-market-is-going-crazy-and-its-going-to-get-worse-how-do-we-recruit/>.
2. 9 Common Job Search Problems and How to Fix Them [Electronic resource]. – Access mode : <https://careerconfidential.com/9-common-job-search-problems-and-how-to-fix-them/>.
3. 14 Job Hunting Tips To Get the Job You Want [Electronic resource]. – Access mode : <https://www.indeed.com/career-advice/finding-a-job/job-hunting-tips>.
4. Foss, J. 6 Job Search Tips That Are So Basic People Forget Them [Electronic resource] / J. Foss. – Access mode : <https://www.themuse.com/advice/6-job-search-tips-that-are-so-basic-people-forget-them>.
5. Сайт пошуку роботи в Україні. Пошук вакансій та резюме на Jobs.ua [Електрон. ресурс]. – Режим доступу : <https://jobs.ua/>.
6. Jobs in USA – 860.000+ current vacancies. Jooble [Електрон. ресурс]. – Режим доступу : <https://jooble.org/>.
7. Dutta, M. Word2Vec For Word Embeddings – A Beginner's Guide [Electronic resource] / M. Dutta. – Access mode : <https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/>.
8. Word2Vec [Электрон. ресурс] : как работать с векторными представлениями слов. – Режим доступа : <https://neurohive.io/ru/osnovy-data-science/word2vec-vektornye-predstavlenija-slov-dlja-mashinnogo-obuchenija/>.
9. Understanding TF-IDF (Term Frequency-Inverse Document Frequency) [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/>.

10. Understanding TF-IDF for Machine Learning [Electronic resource]. – Access mode : <https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/>.
11. Understanding TF-ID [Electronic resource] : A Simple Introduction . – Access mode : <https://monkeylearn.com/blog/what-is-tf-idf/>.
12. Saini, A. Support Vector Machine(SVM) [Electronic resource] : A Complete guide for beginners. – Access mode : <https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/>.
13. Классификация данных методом опорных векторов [Электрон. ресурс] : примеры на Python. – Режим доступа : <https://habr.com/ru/post/105220/>.
14. Web Development with Django [Text] : Learn to build modern web applications with a Python-based framework / B. Shaw, S. Badfwar, A. Bird et al. – Birmingham : Packt Publishing, 2021. – 826 p.
15. Дронов, В. А. Django 3.0. Практика создания веб-сайтов на Python [Текст] / В. А. Дронов. – СПб. : БХВ-Петербург, 2021. – 704 с.
16. Progressus Free Website Template [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page273/progressus>.
17. Jobs On CareerBuilder UK [Electronic resource]. – Access mode : <https://data.world/promptcloud/jobs-on-careerbuilder-uk>.

ДОДАТОК А
Технічне завдання

Вступ

Процес працевлаштування потребує від його учасників, як представників компаній, так і кандидатів на вакантні місця, особливих вмінь. Здебільшого ці вміння стосуються правильної організації процесу, вміння знаходити і обирати потрібні вакансії з одного боку, а з іншого боку – формувати такі вакансії, які будуть привертати увагу значної аудиторії. Оцінюючи вакансію, представникам компанії важливо розуміти, скільки часу займе процес пошуку працівників, а кандидатам важливо розуміти, на скільки вакансія є популярною, розраховуючи з цього власні сили, щоб не витрачати час дарма у випадку недостатньо високих конкурентних позицій.

Програмне забезпечення, що має підтримувати даний процес, повинно надавати компаніям можливість проаналізувати сформовану вакансію і спрогнозувати для неї час, який вона прогнозовано буде активною.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів підтримки формування оголошень про працевлаштування», яку затверджено наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Розробка призначена для формування оголошень про працевлаштування, пошуку вакансій на ринку, виставлення власних резюме, пошуку серед наявних резюме. Тому програма може використовуватися як кандидатами, так і представниками компаній.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма має надавати можливість використовувати наступні функції:

- перегляд вакансій на працевлаштування;
- пошук вакансій на працевлаштування за назвою;
- розширений пошук вакансій;
- перегляд вакансій за посадою;
- перегляд вакансій за групою;
- пошук резюме;
- перегляд резюме за посадою;
- перегляд вакансій за посадою;
- перегляд користувачем своїх резюме;
- перегляд пропозицій на вакансію;
- перегляд власних пропозицій про працевлаштування;
- перегляд пропозицій за ярликом;
- перегляд наявних ярликів;
- перегляд наявних груп для упорядкування вакансій та резюме;
- перегляд переліку посад;
- перегляд компаній;
- перегляд вакансій за компанією;
- пошук компаній;
- перегляд компаній за містом;
- перегляд вакансій за країною;
- перегляд вакансій за містом;
- додавання і редагування за необхідності даних компаній;
- створення і редагування (в разі помилок) ярликів;
- створення і редагування пропозицій про працевлаштування;
- створення і редагування резюме;

- створення і редагування посад;
- створення і редагування вакансій;
- перегляд повних даних резюме;
- перегляд повних даних пропозиції;
- авторизація з наданням окремих облікових записів кандидатів на працевлаштування, представників компаній та адміністраторів.

А.3.2 Вимоги до складу та параметрів технічних засобів

Для роботи програми має бути виділено окремий сервер, що має мінімально відповідати наступним вимогам:

- процесор, який має 4 ядра з тактовою частотою в 2,66 ГГц;
- оперативна пам'ять обсягом у 8 Гб;
- жорсткий диск з загальним обсягом не менше 8 Гб, призначений для серверного використання.

А.4 Порядок контролю та приймання

Календарний план є основою для контролю виконання роботи, у свою чергу приймання роботи здійснюється під час її захисту перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла `models.py`

```

from django.contrib.auth.models import AbstractUser
from django.db import models
from django.urls import reverse

class Job (models.Model):
    title = models.CharField(max_length=255)
    description = models.TextField()
    education = models.TextField()
    skills = models.TextField()
    published = models.DateTimeField()
    company = models.ForeignKey (Company,
on_delete=models.CASCADE)
    city = models.ForeignKey (City, on_delete=models.CASCADE)
    experience = models.TextField()
    address = models.CharField(max_length=255)
    phone = models.CharField(max_length=255)
    employmenttype = models.ForeignKey (EmploymentType,
on_delete=models.CASCADE)
    minsalary = models.IntegerField()
    maxsalary = models.IntegerField()
    activity = models.IntegerField()
    ended = models.DateTimeField()
    predictedtime = models.DateTimeField()
    updated = models.DateTimeField()
    group = models.ForeignKey (Group, on_delete=models.CASCADE)
    shortcuts = models.ManyToManyField(Shortcut)
    position = models.ForeignKey (Position, on_delete=models.CASCADE)

```

```
def __str__(self):
    return self.title
```

```
def get_absolute_url(self):
    return reverse('job', args=[self.id])
```

```
class Proposition (models.Model):
    job = models.ForeignKey (Job, on_delete=models.CASCADE)
    useraccount = models.ForeignKey (UserAccount,
on_delete=models.CASCADE)
    published = models.DateTimeField()
    activity = models.IntegerField()
    companytext = models.TextField()
    usertext = models.TextField()
    interview = models.DateTimeField()
    decision = models.TextField()
```

```
def get_absolute_url(self):
    return reverse('proposition', args=[self.id])
```

```
class CompanyAccount (AbstractUser):
    login = models.CharField(max_length=100)
    password = models.CharField(max_length=100)
    company = models.ForeignKey (Company,
on_delete=models.CASCADE)
```

```
def get_absolute_url(self):
    return reverse('companyaccount', args=[self.id])
```

```
class Company (models.Model):
    title = models.CharField(max_length=100)
    phones = models.CharField(max_length=255)
    address = models.CharField(max_length=255)
    description = models.CharField(max_length=255)
    city = models.ForeignKey (City, on_delete=models.CASCADE)
    workers = models.IntegerField()
    image = models.CharField(max_length=255)

    def get_absolute_url(self):
        return reverse('company', args=[self.id])

class City (models.Model):
    title = models.CharField(max_length=100)

    def get_absolute_url(self):
        return reverse('city', args=[self.id])

class Country (models.Model):
    title = models.CharField(max_length=100)

    def get_absolute_url(self):
        return reverse('country', args=[self.id])

class EmploymentType (models.Model):
    title = models.CharField(max_length=100)

    def get_absolute_url(self):
        return reverse('employmenttype', args=[self.id])
```

```

class UserAccount (AbstractUser):
    login = models.CharField(max_length=100)
    password = models.CharField(max_length=100)
    bday = models.DateTimeField()
    education = models.TextField()
    image = models.CharField(max_length=255)

    def get_absolute_url(self):
        return reverse('companyaccount', args=[self.id])

```

```

class Resume (models.Model):
    useraccount = models.ForeignKey (UserAccount,
on_delete=models.CASCADE)
    email = models.CharField(max_length=100)
    phone = models.CharField(max_length=255)
    title = models.CharField(max_length=255)
    experience = models.TextField()
    published = models.DateTimeField ()
    activity = models.IntegerField()
    city = models.ForeignKey (City, on_delete=models.CASCADE)
    group = models.ForeignKey (Group, on_delete=models.CASCADE)
    pdf = models.CharField(max_length=255)
    position = models.ForeignKey (Position, on_delete=models.CASCADE)

    def get_absolute_url(self):
        return reverse('resume', args=[self.id])

```

```

class Shortcut (models.Model):
    title = models.CharField(max_length=100)

```

```

class Group (models.Model):

```

```
title = models.CharField(max_length=100)
```

```
class Position (models.Model):
```

```
    title = models.CharField(max_length=100)
```

```
    def get_absolute_url(self):
```

```
        return reverse('position', args=[self.id])
```

Б.2 Текст файла detailview.py

```
from django.views.generic.detail import DetailView
```

```
from django.contrib.auth.mixins import LoginRequiredMixin
```

```
import JobAdvertisementCreator.models
```

```
class JobCompanyView (LoginRequiredMixin, DetailView):
```

```
    model = Job
```

```
    template_name = 'adjob.html'
```

```
    def get_context_data(self, **kwargs):
```

```
        jobs = super().get_context_data(**kwargs)
```

```
        job = jobs.get(id=self.kwargs['pk'])
```

```
        jobs ['jobabout'] = job
```

```
        propositions = Proposition.objects.get(job_id = self.kwargs['pk'])
```

```
        propositions.order_by('-id')
```

```
        jobs ['propositions'] = propositions
```

```
        return jobs
```

```
class JobView (DetailView):
```

```
model = Job
```

```
template_name = 'job.html'
```

```
def get_context_data(self, **kwargs):
    jobs = super().get_context_data(**kwargs)
    job = jobs.get(id=self.kwargs['pk'])
    jobs ['jobabout'] = job
    return jobs
```

```
class ResumeView (DetailView):
```

```
    model = Resume
```

```
    template_name = 'resume.html'
```

```
    def get_context_data(self, **kwargs):
        resumes = super().get_context_data(**kwargs)
        resume = resumes.get(id=self.kwargs['pk'])
        resumes ['resumeabout'] = resume
        return resumes
```

```
class PropositionView (DetailView):
```

```
    model = Proposition
```

```
    template_name = 'proposition.html'
```

```
    def get_context_data(self, **kwargs):
        propositions = super().get_context_data(**kwargs)
        proposition = propositions.get(id=self.kwargs['pk'])
        propositions ['proposabout'] = proposition
        return propositions
```

```
class CompanyView (DetailView):
```

```
    model = Company
```

```
    template_name = 'company.html'
```

```
    def get_context_data(self, **kwargs):
```

```
        companies = super().get_context_data(**kwargs)
```

```
        company = companies.get(id=self.kwargs['pk'])
```

```
        companies ['companyabout'] = company
```

```
        return companies
```

Б.3 Текст файла newview.py

```
from django.contrib.auth.mixins import LoginRequiredMixin
```

```
from django.views.generic.edit import CreateView
```

```
import datetime
```

```
import JobAdvertisementCreator.models
```

```
class NewJobView(LoginRequiredMixin, CreateView):
```

```
    template_name = 'newjob.html'
```

```
    model = Job
```

```
    fields = [
```

```
        'title',
```

```
        'description',
```

```
        'education',
```

```
        'skills',
```

```
        'company',
```

```

        'city',
        'experience',
        'address',
        'phone',
        'employmenttype',
        'minsalary',
        'maxsalary',
        'group',
        'shortcuts'
    ]

```

```

login_url = 'companyauthview'
success_url = 'jobview'

```

```

def form_valid(self, form):
    form.published = datetime.datetime.now()
    form.activity = 0
    return super().form_valid(form)

```

```

class NewCompanyView(LoginRequiredMixin, CreateView):

```

```

    template_name = 'newcompany.html'
    model = Company

```

```

    fields = [
        'title',
        'description',
        'city',
        'address',
        'phones',

```

```
        'workers'  
    ]
```

```
login_url = 'adminauthview'  
success_url = 'companyview'
```

```
class NewPositionView(LoginRequiredMixin, CreateView):
```

```
    template_name = 'newposition.html'  
    model = Position
```

```
    fields = [  
        'title'  
    ]
```

```
login_url = 'adminauthview'  
success_url = 'positionview'
```

```
classNewPropositionView(LoginRequiredMixin, CreateView):
```

```
    template_name = 'newproposition.html'  
    model = Proposition
```

```
    fields = [  
        'usertext'  
    ]
```

```
login_url = 'authview'  
success_url = 'propositionview'
```

```
def form_valid(self, form):  
    form.published = datetime.datetime.now()  
    form.activity = 0  
    form.useraccount = self.request.user  
    return super().form_valid(form)
```

```
class NewShortcutView(LoginRequiredMixin, CreateView):
```

```
    template_name = 'newshortcut.html'  
    model = Shortcut
```

```
    fields = [  
        'title'  
    ]
```

```
    login_url = 'authview'  
    success_url = 'shortcutview'
```

```
class NewResumeView(LoginRequiredMixin, CreateView):
```

```
    template_name = 'newresume.html'  
    model = Resume
```

```
    fields = [  
        email,  
        phone,  
        title,  
        experience,  
        city,  
        group,
```

```
pdf
]
```

```
login_url = 'authview'
success_url = 'propositionview'
```

```
def form_valid(self, form):
    form.published = datetime.datetime.now()
    form.activity = 0
    form.useraccount = self.request.user
    return super().form_valid(form)
```

Б.4 Текст файла editview.py

```
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import UpdateView
import datetime
import JobAdvertisementCreator.models
```

```
class JobEditView (LoginRequiredMixin, UpdateView):
```

```
    template_name = 'jobedit.html'
    model = Job
```

```
    fields = [
        'title',
        'description',
        'education',
        'skills',
        'company',
```

```

'city',
'experience',
'address',
'phone',
'employmenttype',
'minsalary',
'maxsalary',
'group',
'shortcuts'
]

```

```
login_url = 'companyauthview'
```

```
success_url = 'jobview'
```

```
def form_valid(self, form):
```

```
    form.updated = datetime.datetime.now()
```

```
    return super().form_valid(form)
```

```
class EditPositionView(LoginRequiredMixin, UpdateView):
```

```
    template_name = 'editposition.html'
```

```
    model = Position
```

```
    fields = [
```

```
        'title'
```

```
    ]
```

```
    login_url = 'adminauthview'
```

```
    success_url = 'positionview'
```

```
class ResumeEditView (LoginRequiredMixin, UpdateView):
```

```
template_name = 'resumeedit.html'  
model = Resume
```

```
fields = [  
    email,  
    phone,  
    title,  
    experience,  
    city,  
    group,  
    pdf  
]
```

```
login_url = 'authview'  
success_url = 'propositionview'
```

```
class UserPropositionEditView (LoginRequiredMixin, UpdateView):
```

```
    template_name = 'proposedit.html'  
    model = Proposition
```

```
    fields = [  
        'usertext'  
    ]
```

```
    login_url = 'authview'  
    success_url = 'propositionview'
```

```
class CompanyPropositionEditView (LoginRequiredMixin, UpdateView):
```

```
    template_name = 'coproposedit.html'
```

```
model = Proposition
```

```
fields = [  
    'activity',  
    'companytext',  
    'interview',  
    'decision'  
]
```

```
login_url = 'authview'
```

```
success_url = 'propositionview'
```

```
class EditShortcutView(LoginRequiredMixin, UpdateView):
```

```
    template_name = 'editshortcut.html'
```

```
    model = Shortcut
```

```
    fields = [  
        'title'  
    ]
```

```
class EditCompanyView(LoginRequiredMixin, UpdateView):
```

```
    template_name = 'editcompany.html'
```

```
    model = Company
```

```
    fields = [  
        'title',  
        'description',  
        'city',
```

```

        'address',
        'phones',
        'workers'
    ]

```

```

login_url = 'adminauthview'
success_url = 'companyview'

```

Б.5 Текст файла `ListView.py`

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView
import JobAdvertisementCreator.models

```

```

class PositionJobsView (ListView):

```

```

    model = Job
    template_name = 'positionjobs.html'

```

```

    def get_queryset(self, *args, **kwargs):
        jobs = super(PositionJobsView, self).get_queryset(*args, **kwargs)
        getposition = self.request.GET.get('position')
        jobs = jobs.filter(position = getposition)
        result = jobs.filter(activity = 0)
        result.order_by('-published')
        jobs ['bypos'] = result
        return jobs

```

```

class CityJobsView (ListView):

```

```
model = Job
template_name = 'cityjobs.html'

def get_queryset(self, *args, **kwargs):
    jobs = super(CityJobsView, self).get_queryset(*args, **kwargs)
    getcity = self.request.GET.get('city')
    jobs = jobs.filter(city = getcity)
    result = jobs.filter(activity = 0)
    result.order_by('-published')
    jobs ['bycity'] = result
    return jobs
```

```
class CompanyJobsView (ListView):
```

```
    model = Job
    template_name = 'companyjobs.html'

    def get_queryset(self, *args, **kwargs):
        jobs = super(CompanyJobsView, self).get_queryset(*args, **kwargs)
        getcompany = self.request.GET.get('company')
        jobs = jobs.filter(company = getcompany)
        result.order_by('-published')
        jobs ['bycompany'] = result
        return jobs
```

```
class GroupJobsView (ListView):
```

```
    model = Job
    template_name = 'groupjobs.html'

    def get_queryset(self, *args, **kwargs):
```

```

jobs = super(GroupJobsView, self).get_queryset(*args, **kwargs)
getgroup = self.request.GET.get('group')
jobs = jobs.filter(group = getgroup)
result = jobs.filter(activity = 0)
result.order_by('-published')
jobs ['bygroup'] = result
return jobs

```

```
class CountryJobsView (ListView):
```

```

    model = Job
    template_name = 'countryjobs.html'

```

```

    def get_queryset(self, *args, **kwargs):
        jobs = super(CountryJobsView, self).get_queryset(*args, **kwargs)
        getcountry = self.request.GET.get('country')
        jobs = jobs.filter(city_country = getcountry)
        result = jobs.filter(activity = 0)
        result.order_by('-published')
        jobs ['bycountry'] = result
        return jobs

```

```
class SearchJobsView (ListView):
```

```

    model = Job
    template_name = 'searchjobs.html'

```

```

    def get_queryset(self, *args, **kwargs):
        jobs = super(SearchJobsView, self).get_queryset(*args, **kwargs)
        gettitle = self.request.GET.get('title')

```

```

jobs = jobs.filter(title__contains = gettitle)
result = jobs.filter(activity = 0)
result.order_by('-published')
jobs ['searchresults'] = result
return jobs

```

```

class AdvancedSearchJobsView (ListView):

```

```

    model = Job

```

```

    template_name = 'advsearchjobs.html'

```

```

    def get_queryset(self, *args, **kwargs):

```

```

        jobs = super(AdvancedSearchJobsView, self).get_queryset(*args,
**kwargs)

```

```

        gettitle = self.request.GET.get('title')

```

```

        getcompany= self.request.GET.get('company')

```

```

        getcity = self.request.GET.get('city')

```

```

        getposition = self.request.GET.get('position')

```

```

        getgroup = self.request.GET.get('group')

```

```

        getemptytype = self.request.GET.get('emptytype')

```

```

        getminsal = self.request.GET.get('minsal')

```

```

        getmaxsal = self.request.GET.get('maxsal')

```

```

        getway = self.request.GET.get('way')

```

```

        jobs = jobs.filter(title__contains = gettitle)

```

```

        jobs = jobs.filter(company = getcompany)

```

```

        jobs = jobs.filter(city = getcity)

```

```

        jobs = jobs.filter(position = getposition)

```

```

        jobs = jobs.filter(group = getgroup)

```

```

        jobs = jobs.filter(maxsalary__lte = getmaxsal)

```

```

        jobs = jobs.filter(minsalary__gte = getminsal)

```

```

result = jobs.filter(activity = 0)
result.order_by(way)
jobs ['adsearchresults'] = result
return jobs

```

```
class CityCompaniesView (ListView):
```

```

    model = Company
    template_name = 'citycompanies.html'

```

```

    def get_queryset(self, *args, **kwargs):
        companies = super(CityCompaniesView, self).get_queryset(*args,
**kwargs)
        getcity = self.request.GET.get('city')
        companies = companies.filter(city = getcity)
        result = companies.filter(activity = 0)
        companies ['citycomp'] = result
        return companies

```

```
class CompaniesView (ListView):
```

```

    model = Company
    template_name = 'companies.html'

```

```
class PositionsView (ListView):
```

```

    model = Position
    template_name = 'positions.html'

```

```
class GroupsView (ListView):
```

```

model = Group
template_name = 'groups.html'

```

```

class PositionResumesView (ListView):

```

```

    model = Resume
    template_name = 'posioioresumes.html'

```

```

    def get_queryset(self, *args, **kwargs):
        positions = super(PositionResumesView, self).get_queryset(*args,
**kwargs)

        getposition = self.request.GET.get('position')
        positions = positions.filter(position = getposition)
        result = positions.filter(activity = 0)
        result.order_by('-published')
        positions ['bypos'] = result
        return positions

```

```

class ShortcutsView (ListView):

```

```

    model = Shortcut
    template_name = 'shortcuts.html'

```

```

class ShortcutJobsView (ListView):

```

```

    model = Job
    template_name = 'shortjobs.html'

```

```

    def get_queryset(self, *args, **kwargs):
        jobs = super(ShortcutJobsView, self).get_queryset(*args, **kwargs)

```

```

getshortcut = self.request.GET.get('shortcut')
jobs = jobs.filter(shortcuts__contains = getshortcut)
result = jobs.filter(activity = 0)
result.order_by('-published')
jobs ['byshort'] = result
return jobs

```

```
class UserPropositionsView (ListView):
```

```

    model = Proposition
    template_name = 'userpropositions.html'

```

```

    def get_queryset(self, *args, **kwargs):
        propositions = super(UserPropositionsView, self).get_queryset(*args,
**kwargs)
        getuser = self.request.user
        propositions = propositions.filter(useraccount = getuser)
        result.order_by('-published')
        propositions ['userpropos'] = result
        return propositions

```

```
class UserResumesView (ListView):
```

```

    model = Resume
    template_name = 'userresumes.html'

```

```

    def get_queryset(self, *args, **kwargs):
        resumes = super(UserResumesView, self).get_queryset(*args,
**kwargs)
        getuser = self.request.user

```

```

resumes = resumes.filter(useraccount = getuser)
result.order_by('-published')
resumes ['usresumes'] = result
return resumes

```

```
class AdvancedSearchResumesView (ListView):
```

```

    model = Resumes
    template_name = 'advsearchresumes.html'

```

```

    def get_queryset(self, *args, **kwargs):
        resumes = super(AdvancedSearchResumesView,
self).get_queryset(*args, **kwargs)
        gettitle = self.request.GET.get('title')
        getcity = self.request.GET.get('city')
        getposition = self.request.GET.get('position')
        getgroup = self.request.GET.get('group')
        getway = self.request.GET.get('way')
        resumes = resumes.filter(city = getcity)
        resumes = resumes.filter(position = getposition)
        resumes = resumes.filter(group = getgroup)
        resumes = resumes.filter(title__contains = gettitle)
        result = resumes.filter(activity = 0)
        result.order_by(getway)
        resumes ['adsearchresults'] = result
        return resumes

```

```
class CitySearchView (ListView):
```

```

    model = City

```

```
template_name = 'searchcities.html'
```

```
def get_queryset(self, *args, **kwargs):
    cities = super(CitySearchView, self).get_queryset(*args, **kwargs)
    gettitle = self.request.GET.get('title')
    cities = cities.filter(title__contains = gettitle)
    cities ['searchresults'] = result
    return cities
```

```
class CompanySearchView (ListView):
```

```
    model = Company
    template_name = 'searchcompanies.html'
```

```
    def get_queryset(self, *args, **kwargs):
        companies = super(CompanySearchView, self).get_queryset(*args,
**kwargs)
        gettitle = self.request.GET.get('title')
        companies = companies.filter(title__contains = gettitle)
        companies ['searchresults'] = result
        return companies
```

ДОДАТОК В
Слайди презентації

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ПІДТРИМКИ ФОРМУВАННЯ ОГолошень ПРО ПРАЦЕВЛАШТУВАННЯ

Виконав
студент групи КНТ-210м

М.С. Сердюк

Керівник
к.т.н., доцент

Т.А. Зайко

2021

Рисунок В.1 – Слайд 1

Об'єкт дослідження – процес формування оголошень про працевлаштування.

Предмет дослідження – методи прогнозування періоду доступності оголошення про працевлаштування.

Мета роботи – розробити програмне забезпечення формування і пошуку оголошень про працевлаштування для надання інформації для прийняття рішень про вибір конкретного формулювання вакансії на основі прогнозування періоду її доступності.

Рисунок В.2 – Слайд 2

Завдання роботи

3

- провести аналіз предметної області, що було виконано в даному розділі;
- провести дослідження методів прогнозування періоду доступності вакансії на працевлаштування;
- реалізувати проектування програми для формування і пошуку оголошень про працевлаштування;
- реалізувати програму для формування і пошуку оголошень про працевлаштування.

Рисунок В.3 – Слайд 3

Порівняння аналогів

4

Характеристика	jobs.ua	jooble.org	Власна розробка
Пошук вакансій	+	+	+
Пошук резюме	+	–	+
Сортування результатів пошуку вказаним користувачем чином	–	–	+
Наявність ярликів оголошень і пошук за ярликами оголошень	+-	–	++
Прогнозування часу доступності вакансій	–	–	+

Рисунок В.4 – Слайд 4

Метод прогнозування періоду доступності вакансії на працевлаштування

Часові категорії:

- період до 1 доби;
- період до 5 діб;
- період до 10 діб;
- період до 1 місяця;
- тривалі.

На першому етапі визначається набір даних.

На другому етапі виконується передоброблення тестових даних шляхом виділення з них слів, вилучення знаків пунктуації, спеціальних символів

На третьому етапі необхідно провести дослідження текстових описів усіх вакансій та виконати попередні обчислення частот слів на корпусі текстів, отриманого на першому етапі, з виділенням слоника слів.

На четвертому етапі потрібно провести навчання моделі. Для цього кожний отриманий опис з вибірки даних потрібно обробити, перетворюючи його у векторну форму, що складається з частот слів, а далі отриману оцінку подати на вхід метода опорних векторів.

На п'ятому етапі реалізується прогнозування часової категорії періоду доступності вакансії для працевлаштування на основі реальних даних.

5

Рисунок В.5 – Слайд 5

Порівняння фреймворків веброзробки для мови програмування Python

Характеристика	Django	Flask
Об'єктно-реляційне відображення	Так	Немає за замовчуванням, але можна встановити як розширення
Уникнення SQL-ін'єкцій	Так	Ні
Підтримка	Краща	Гірша
Наявність вбудованої адміністративної панелі	Так	Ні
Масштаб проєктів	Великі, середні	Малі

6

Рисунок В.6 – Слайд 6

Діаграма прецедентів

7

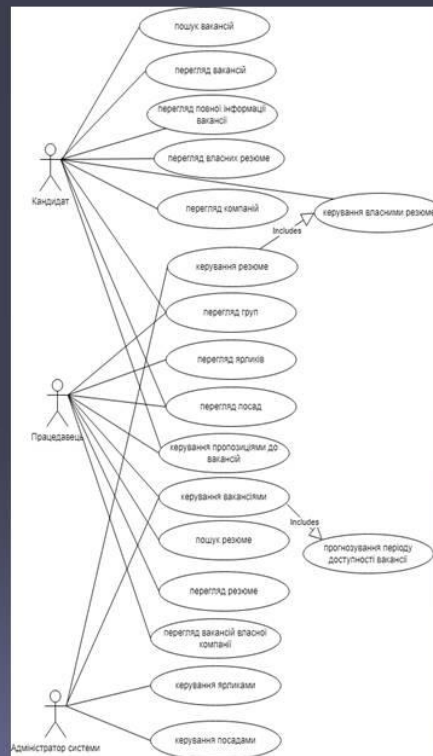


Рисунок В.7 – Слайд 7

Схема бази даних

8

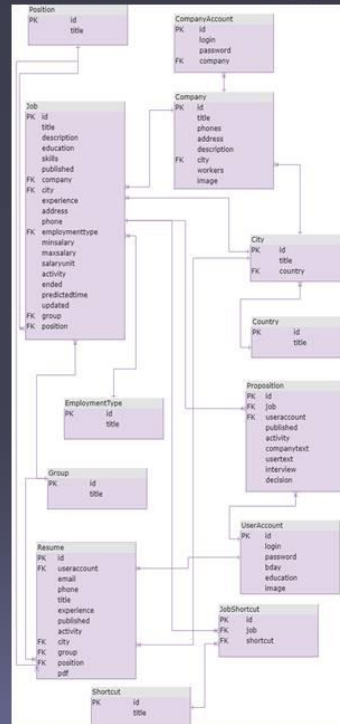
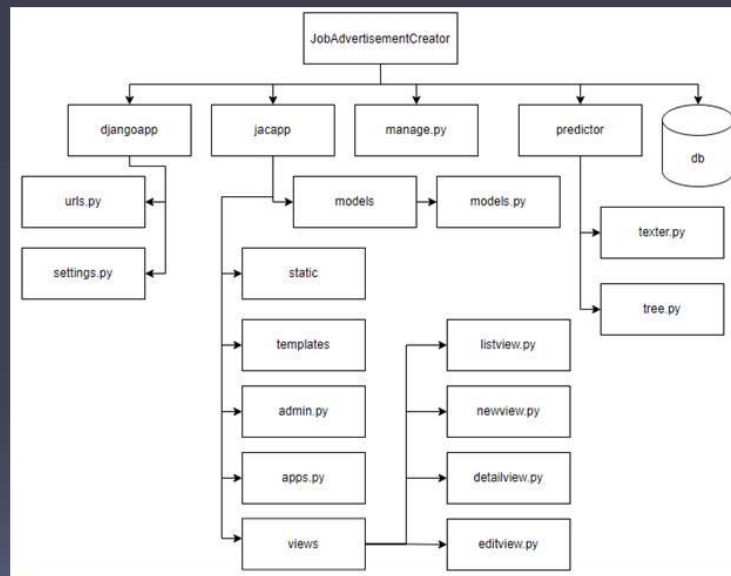


Рисунок В.8 – Слайд 8

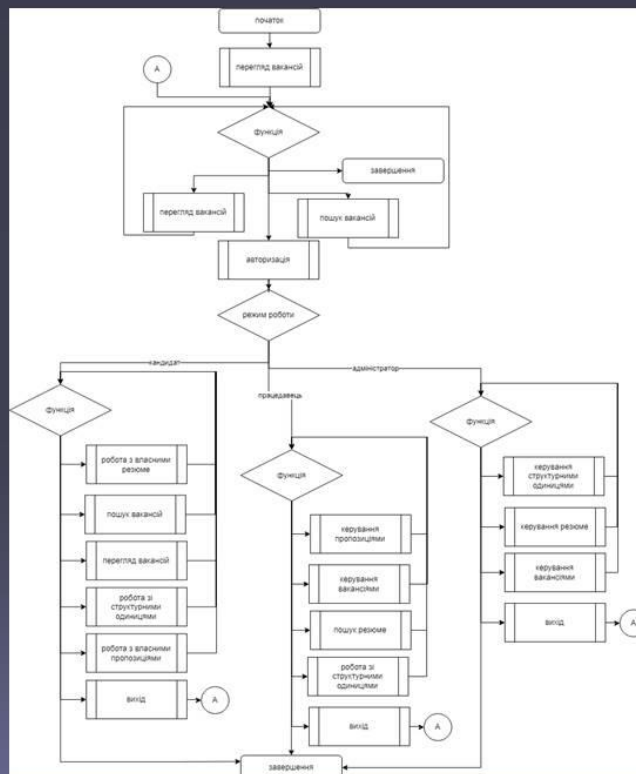
Структурна схема програми



9

Рисунок В.9 – Слайд 9

Схема функціонування програми



10

Рисунок В.10 – Слайд 10

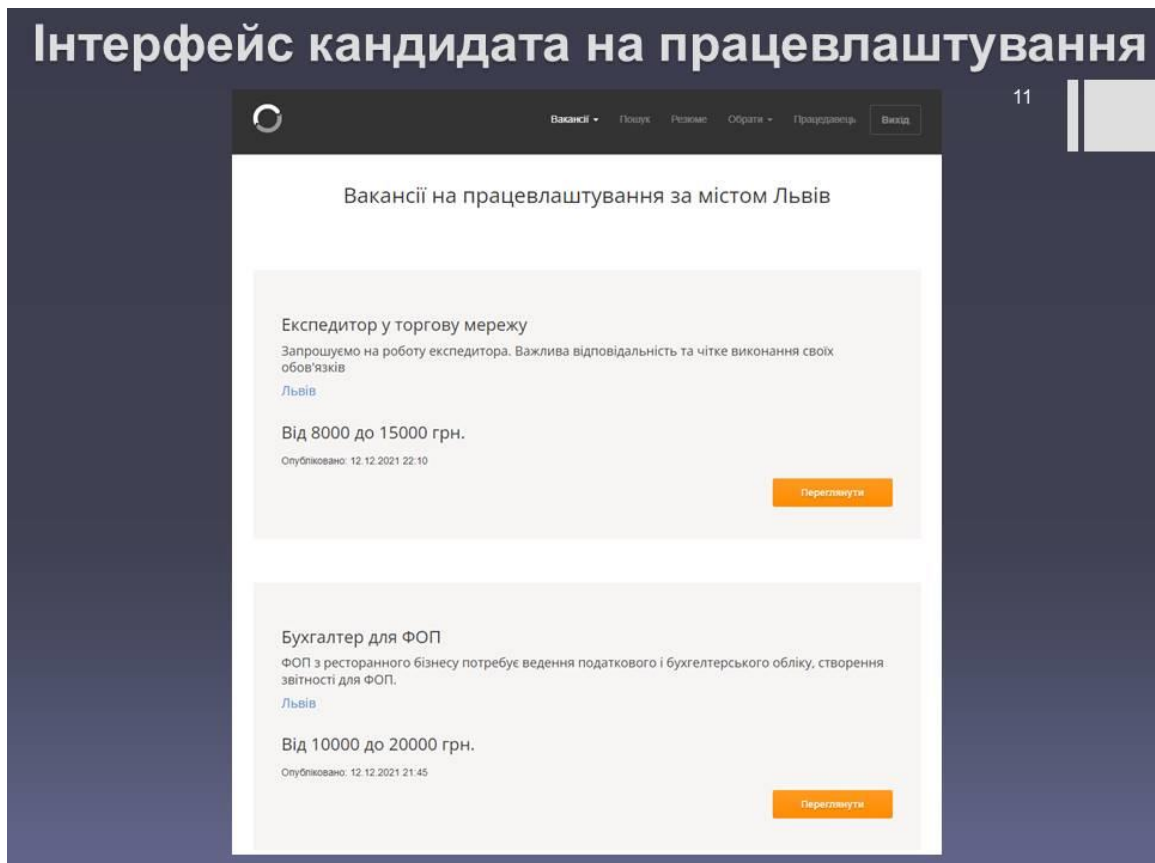


Рисунок В.11 – Слайд 11

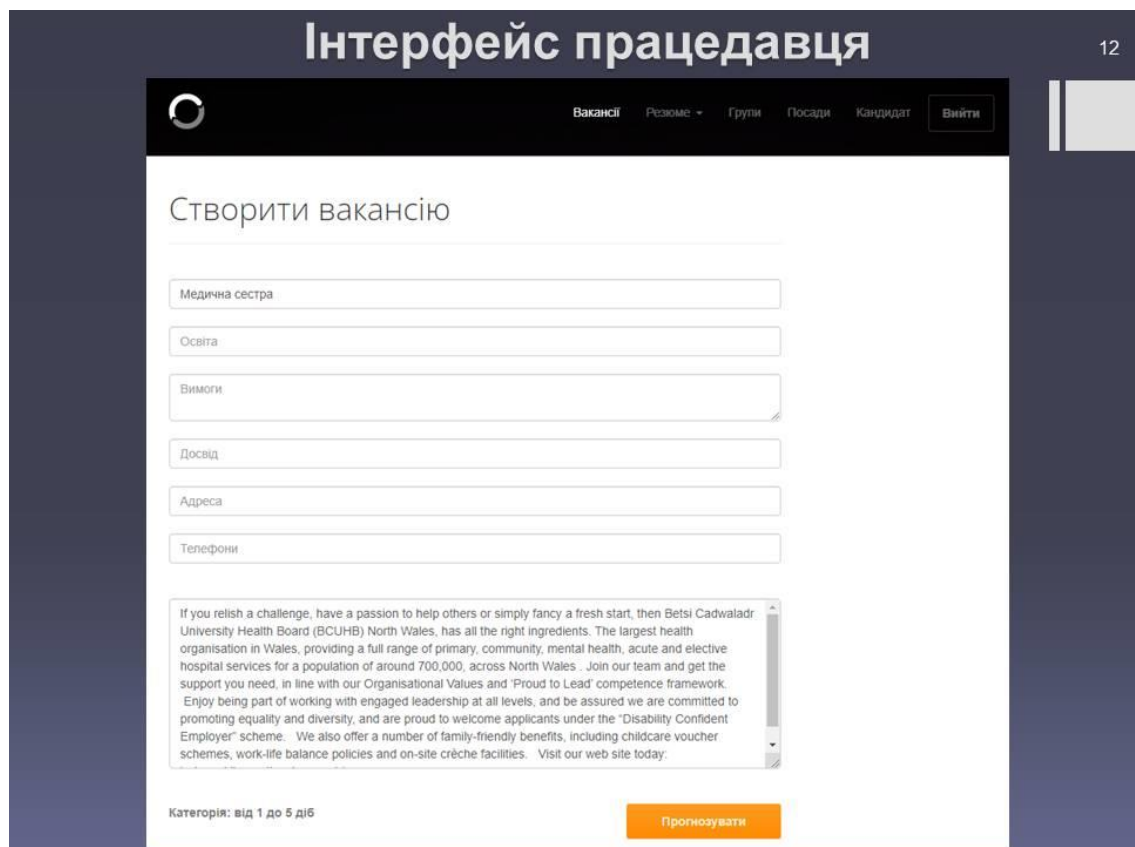


Рисунок В.12 – Слайд 12

Експериментальне дослідження

13

Група вакансій	Точність класифікації
Інженерія	82,5
Медицина	79,2

Рисунок В.13 – Слайд 13

Висновки

У дипломній кваліфікаційній роботі розроблено програмне забезпечення формування і пошуку оголошень про працевлаштування для надання інформації для прийняття рішень про вибір конкретного формулювання вакансії на основі прогнозування періоду її доступності.

Розроблено програмне забезпечення формування і пошуку оголошень про працевлаштування, що надає кандидатам на працевлаштування, роботодавцям можливість формувати оголошення про працевлаштування з наданням можливості прогнозування періоду доступності такої вакансії, виконувати пошук вакансій на ринку, виставляти власні резюме, виконувати пошук серед наявних резюме.

Наукова новизна роботи полягає в методі прогнозування періоду доступності вакансії на працевлаштування, який відзначається застосуванням показника tf-idf та методу опорних векторів, визначаючи таким чином часову категорію, що дозволяє приймати в подальшому рішення щодо вибору конкретного формулювання вакансії.

14

Рисунок В.14 – Слайд 14