НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Вебзастосунок для спрощення пошуку роботи»

Виконав:
студент IV курсу, групи КП-93
Долгов Олексій Юрійович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Олещенко Любов Михайлівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри інформаційних технологій, штучного інтелекту та кібербезпеки, к.т.н., доцент,
Мошенський Андрій Олександрович _____

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

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____ 2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Долгову Олексій Юрійовичу

1. Тема проєкту «Вебзастосунок для спрощення пошуку роботи», керівник проєкту Олещенко Любов Михайлівна, к.т.н., доцент, затверджені наказом по університету від «31» травня 2023 р. №2107-С.

2. Термін подання студентом проєкту «16» червня 2023 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
– аналіз існуючих програмних рішень;
– обґрунтування вибору засобів реалізації;
– розроблення архітектури та програмних модулів;
– аналіз розробленого програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
– схема бази даних. ERD-діаграма (креслення);
– взаємодія з додатком користувача «Адміністратор». Діаграма діяльності (креслення);
– взаємодія з додатком користувача «Кандидат». Діаграма діяльності (креслення);
– дерево проблем (плакат);
– дерево цілей розробленого ПЗ (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «30» жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	09.11.2022	
2.	Розроблення та узгодження технічного завдання	26.11.2022	
3.	Розроблення структури програми	15.12.2022	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2022	
5.	Розроблення дизайну, псевдокоду алгоритмів.	05.02.2023	
6.	Підготовка матеріалів другого розділу дипломного проєкту	18.02.2023	
7.	Програмна реалізація	10.03.2023	
8.	Тестування програми	17.03.2023	
9.	Підготовка матеріалів третього розділу проєкту	30.03.2023	
10.	Підготовка графічної частини дипломного проєкту	21.04.2023	
11.	Оформлення документації дипломного проєкту	28.05.2023	

Студент Олексій ДОЛГОВ

Керівник проєкту Любов ОЛЕЩЕНКО

АНОТАЦІЯ

Даний дипломний проєкт присвячений створенню вебзастосунку для полегшення процесу пошуку роботи.

Задача дипломного проєкту полягає у тому, щоб розробити програмне забезпечення, що дозволить спростити процес пошуку роботи кандидатами.

Проаналізовано існуючі аналоги веб застосунків, після чого сформовано вимоги до розробленого веб застосунку, з урахуванням недоліків конкурентів.

Побудований застосунок дозволяє будь-кому, хто знаходиться у процесі пошуку роботи отримувати релевантні вакансії.

У даному дипломному проєкті було розроблено: архітектуру для серверної та клієнтської частини веб застосунку, структуру SQL та NoSQL бази даних, алгоритм реєстрації та авторизації користувача, алгоритм внесення та обробки інформації про уподобання користувачів, модуль для скрапінгу і аналізу оголошень про роботу, модуль для побудови рекомендацій для користувачів.

ABSTRACT

This diploma project is dedicated to the development of a web application to facilitate the job search process. The task of the project is to create software that will simplify the job search process for candidates.

Existing analogs of web applications have been analyzed, and requirements for the developed web application have been formulated, taking into account the drawbacks of competitors.

The constructed application allows anyone who is in the process of searching for a job to receive relevant job vacancies. In this diploma project, the following have been developed: architecture for the server and client parts of the web application, the structure of SQL and NoSQL databases, user registration and authorization algorithms, algorithms for collecting and processing user preference information, a module for scraping and analyzing job advertisements, and a module for generating recommendations for users.

ДП.045440-01-90. Вебзастосунок для спрощення пошуку роботи. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Вебзастосунок для	5	
	спрощення пошуку роботи.		
	Технічне завдання		
ДП.045440-03-81	Вебзастосунок для	55	
	спрощення пошуку роботи.		
	Пояснювальна записка		
ДП.045440-04-51	Вебзастосунок для	4	
	спрощення пошуку роботи.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Вебзастосунок для	14	
	спрощення пошуку роботи.		
	Керівництво користувача		
ДП.045440-06-99	Вебзастосунок для	1	
	спрощення пошуку роботи.		
	Структура бази даних.		
	ERD-діаграма		
ДП.045440-07-99	Вебзастосунок для	1	
	спрощення пошуку роботи.		
	Взаємодія з додатком		
	користувача «Адміністратор».		
	Діаграма діяльності		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-08-98	Вебзастосунок для	1	
	спрощення пошуку роботи.		
	Взаємодія з додатком		
	користувача «Кандидат».		
	Діаграма діяльності		
ДП.045440-09-98	Вебзастосунок для	1	
	спрощення пошуку роботи.		
	Компакт-диск		

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____ 2022 р.

ВЕБЗАСТОСУНОК СПРОЩЕННЯ ПОШУКУ РОБОТИ

Технічне завдання

ДП.045430-02-91

"ПОГОДЖЕНО"
Керівник проєкту:
_____ Любов ОЛЕЩЕНКО

Нормоконтроль:
_____ Микола ОНАЙ

Виконавець:
_____ Олексій ДОЛГОВ

2022

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебзастосунок для спрощення пошуку роботи.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для створення та пошуку подій користувачів з використанням картографічних сервісів.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмне забезпечення повинно забезпечувати такі можливості:

1. Можливість реєстрації в системі.
2. Створення та оновлення уподобань.
3. Можливість переглядати релевантні вакансії згідно уподобанням.
4. Перегляд скороченої інформації про вакансію.
5. Перегляд повної інформації про вакансію.

Розробку серверної частини проєкту виконати за допомогою Python та фреймворку Django. Для клієнтської частини використати TypeScript та фреймворк Angular. Для зберігання даних використати базу даних PostgreSQL та ElasticSearch.

Додаткові вимоги:

1. Зручність та зрозумілість користувацького інтерфейсу.

2. Адаптивність користувацького інтерфейсу для роботи з ПК.
3. Безпека даних користувачів.
4. Доступність вебзастосунку у найбільш популярних веббраузерах.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1. Пояснювальна записка.
2. Програма та методика тестування.
3. Керівництво користувача.
4. Креслення:
   4.1. Схема бази даних. ERD-діаграма (креслення).
   4.2. Взаємодія з додатком користувача «Кандидат». Діаграма діяльності (креслення).
   4.3. Дерево проблем розробленого ПЗ (плакат).
   4.4. Дерево цілей розробленого ПЗ (плакат).

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи ……... 12.11.2022
Розроблення та узгодження технічного завдання ……. 27.11.2022
Розроблення структури вебдодатку …… 17.12.2022
Розроблення дизайну сторінок та графічних елементів …….. 07.02.2023
Програмна реалізація вебдодатку …….. 11.03.2023
Тестування вебдодатку …… 15.04.2023
Підготовка матеріалів текстової частини проєкту …… 28.04.2023
Підготовка матеріалів графічної частини проєкту ……... 15.05.2023
Оформлення технічної документації проєкту …….. 27.05.2023

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ВЕБЗАСТОСУНОК ДЛЯ СПРОЩЕННЯ ПОШУКУ РОБОТИ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Любов ОЛЕЩЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ДОЛГОВ

2023

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ATS – Це програмне забезпечення (ПЗ), яке дозволяє автоматизувати процес підбору, відстежувати його ефективність та обробляти інформацію відповідно до потреб найму.

GPA – Середній бал диплому у американських вищих навчальних закладах. В американський бізнес традиції є важливим параметром при прийомі на першу роботу.

ПЗ – програмне забезпечення.

СКБД – система керування базами даних.

БД – база даних.

UI (user interface) – графічний інтерфейс, який відповідає за взаємодію користувача з додатком.

ООП – об'єктно-орієнтоване програмування.

SQL-ін'єкція – вид атаки на вебсайти способом вставки SQL-запиту, який буде виконується на вебсервері.

SQL – структурована мова запитів до БД.

Фреймворк – набір програмних рішень, які полегшують розробку комплексного програмного забезпечення.

Django – фреймворк на мові Python для розробки вебзастосунків.

ORM – техніка програмування в програмній інженерії, що ставить у відповідність БД та моделі ООП.

3MVC (model-view-controller) – шаблон проєктування, який включає три пов'язані між собою частини: модель даних, інтерфейс для відображення цих даних та модуль, який керує цими даними.

MVT (model-view-template) – аналог MVC в Django, на відміну від класичного варіанту, контроль віддається фреймворку.

Діаграма діяльності – наочне представлення графу, вершинами якого є визначені дії, а після їх виконання відбувається перехід.

ВСТУП

В сучасному світі пошуку роботи шукачам доводиться стикатися зі значними труднощами. Незважаючи на те, що розміщення оголошень про роботу в Інтернеті значно полегшує процес пошуку, наявність великої кількості платформ і сайтів, що пропонують такі послуги, створює нові проблеми. Одна з них полягає в тому, що великі компанії, на відміну від менших, часто не розміщують свої оголошення на зовнішніх платформах, а використовують свої корпоративні сайти. Це ускладнює пошук роботи для шукачів, які не знають про такі можливості.

Крім того, розбіжності в класифікації вакансій можуть призвести до того, що певні оголошення не будуть відображатися в пошукових запитах, які здавалися шукачам релевантними. Це також ускладнює процес пошуку роботи та знижує ефективність використання платформ.

Метою даного дипломного проєкту є створення рішення, яке може допомогти вирішити ці проблеми. Основними ідеями цього рішення є агрегації оголошень з різних джерел та класифікація їх за єдинообразним стандартом. Це дозволить шукачам ефективно знаходити релевантні вакансії на одній платформі, не витрачаючи час на перегляд оголошень на різних сайтах. Такий підхід може значно полегшити процес пошуку роботи та зробити його більш результативним.

Отже, хоча існують певні недоліки у процесі пошуку роботи в Інтернеті, існують і шляхи, які можуть полегшити цей процес.

1. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Огляд проблеми, яка вирішується ПЗ

Якщо розглянути сучасні системи для пошуку роботи, то вони мають деякі недоліки, що можуть стати причиною втрати часу для користувачів. Основними недоліками таких систем є:

1. Обмежений обсяг вакансій: більшість систем для пошуку роботи включає лише вакансії, що розміщені на цій платформі, ігноруючи корпоративні сайти компаній та інші платформи, що може зменшити шанси на успішне знаходження роботи.
2. Обмежені можливості фільтрації. Багато систем не надають можливості настроювання пошуку за більш детальними критеріями, такими як розташування вакансії, розмір зарплати та інші.
3. Низька ефективність рекомендацій. Більшість систем для пошуку роботи надає рекомендації користувачам на основі кількох критеріїв, що не завжди відповідає потребам конкретного користувача.
4. Система не розуміє реальні потреби користувача. Більшість систем для пошуку роботи вимагають від користувача введення детальної інформації про його досвід роботи та навички, що не завжди відображає реальні потреби користувача.

Розроблювана система для пошуку роботи може вирішити ці недоліки, надаючи користувачам більш актуальну та налаштовувану інформацію про вакансії, а також пропонуючи більш персоналізовані рекомендації на основі аналізу даних про користувача. На рис. 1.1 наведено дерево проблем цієї задачі.

Рис. 1.1. Дерево проблем

1.2. Аналіз та порівняння існуючих рішень

Існує багато систем пошуку роботи, які допомагають знайти вакансії, відповідні певним критеріям. Нижче розглянемо кілька таких систем та їх переваги та недоліки.

1.2.1. *LinkedIn*

LinkedIn є однією з найпопулярніших систем пошуку роботи. Платформа дозволяє користувачам створювати свій профіль, на якому вони можуть додавати свої навички та досвід роботи. Крім того, LinkedIn надає можливість користувачам знаходити вакансії відповідно до своїх критеріїв та підписатися на повідомлення про нові вакансії [2].

Переваги:

- Широкий вибір вакансій з усього світу.
- Можливість знаходити вакансії відповідно до своїх критеріїв.
- Можливість створювати профіль та відстежувати свій розвиток.

Недоліки:

- Немає агрегації вакансій.

- Відсутність стандартизованої класифікацій вакансій і навичок (одна й та сама навичка може мати декілька назв, що погіршує результати системи рекомендацій).
- Відсутність перевірки невідповідності назв вакансій до вказаних обов'язків.

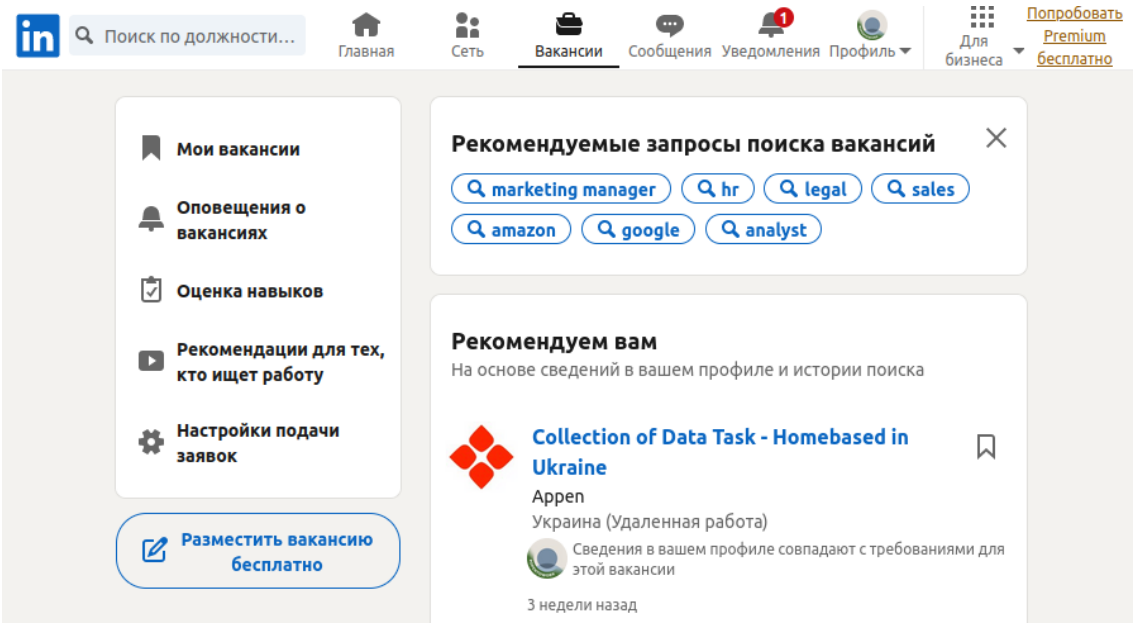


Рис. 1.2. Пошук вакансії у LinkedIn

1.2.2. *Glassdoor*

Glassdoor – це платформа, яка дозволяє користувачам шукати роботу, вивчати інформацію про компанії, огляди від працівників, а також пропонує засоби для дослідження ринку праці [3].

Переваги:

- Інформація про заробітну плату та відгуки від реальних працівників дозволяють отримати чітку картину про компанію та можливість розвитку кар'єри.
- Можливість пошуку вакансій за різними критеріями, такими як місцезнаходження, компанія, розмір компанії та інші.

- Glassdoor також пропонує функцію "Огляд компанії", яка дає користувачам можливість дізнатися більше про потенційного роботодавця, його культуру та стиль управління.

Недоліки:

- Не всі компанії публікують вакансії на Glassdoor.
- Інформація, надана на Glassdoor, може бути піддається впливу особистих переконань та досвіду працівників, тому її слід розглядати з обережністю.
- Інформація на Glassdoor може бути застарілою або неповною.

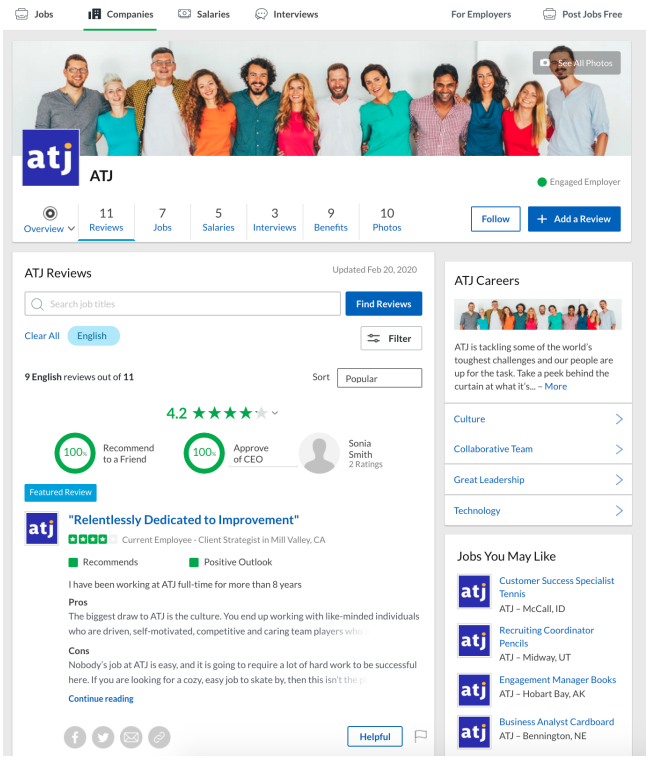


Рис. 1.3. Пошук вакансій у GlassDoor [3]

1.2.3. *Indeed*

Indeed – це популярна платформа пошуку роботи, яка дозволяє користувачам шукати вакансії за ключовими словами, розташуванням, компанією тощо. Платформа також пропонує інструменти для створення

резюме, налаштування сповіщень про нові вакансії і зв'язку з роботодавцями [31]. Ось деякі переваги та недоліки цієї платформи.

Переваги:

- Багато вакансій: Indeed має один з найбільших джерел вакансій у світі, що охоплює більше 60 країн.
- Проста у використанні: користувачі можуть легко шукати вакансії та створювати свої резюме, навіть якщо вони ніколи раніше не користувалися платформою.
- Налаштування сповіщень: Indeed пропонує різні налаштування сповіщень, такі як електронна пошта, додаток або повідомлення в соціальних мережах, щоб повідомляти користувачів про нові вакансії, що відповідають їхнім критеріям.

Недоліки:

- Багато конкуренції: через велику кількість користувачів Indeed, на деякі вакансії може бути велика конкуренція, що зробить пошук роботи більш складним.
- Відсутність персоналізованого підходу: Indeed пропонує багато різних вакансій, але не завжди може забезпечити персоналізований підхід для кожного користувача.
- Багато спаму: так як багато роботодавців можуть відправляти запити на співбесіду, користувачі можуть отримувати багато спаму від ненавмисних роботодавців або рекрутерів.

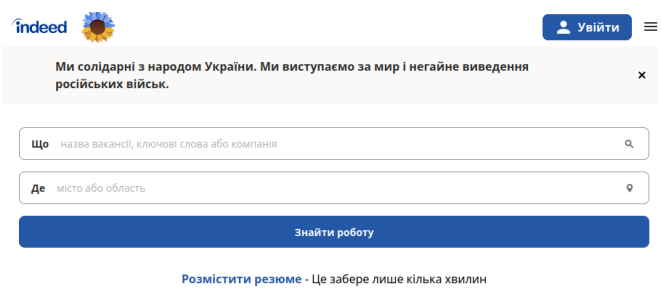


Рис. 1.4. Сторінка пошуку вакансій в Indeed [31]

1.2.4. *Djinni*

Djinni – український вебсайт для пошуку роботи, який позиціонує себе як сайт пошуку роботи для IT-спеціалістів [32].

Переваги:

- Спеціалізація на IT-індустрії: Djinni спеціалізується на пошуку роботи в сфері інформаційних технологій. Це означає, що ви можете знайти вакансії, які точно відповідають вашим навичкам та інтересам у цьому конкретному сегменті ринку праці.
- Багатий вибір вакансій: Djinni має велику базу даних вакансій в IT-сфері, що робить його привабливим для кандидатів, які шукають роботу в цій галузі. Ви маєте можливість вибирати з різноманіття вакансій залежно від вашого досвіду, рівня та спеціалізації.
- Можливості для IT-спеціалістів: Djinni надає можливість IT-спеціалістам презентувати свої навички та досвід, створюючи свій профіль на платформі. Це дозволяє роботодавцям легко знайти та зв'язатися з вами для пропозицій роботи.
- Кар'єрні поради та ресурси: Djinni також надає корисні кар'єрні поради, інформацію про ринок праці та інші ресурси, які можуть бути корисними для IT-спеціалістів, що шукають роботу. Це може допомогти вам підготуватися до співбесіди та покращити свої шанси на успішне працевлаштування.

Недоліки:

- Обмежений фокус: Хоча спеціалізація на IT-сфері є перевагою для IT-спеціалістів, це може бути обмеженням для кандидатів, які шукають роботу в інших галузях. Якщо ви не маєте досвіду в IT-сфері, то Djinni може бути менш релевантною платформою для вас.
- Конкуренція: З огляду на популярність платформи Djinni серед IT-спеціалістів, конкуренція за вакансії може бути високою. Вам

може бути складно виділитися серед інших кандидатів і отримати пропозицію роботи.

- Одностороннє представлення: Платформа Djinni переважно фокусується на представленні кандидатів роботодавцям, а не надає можливості кандидатам активно шукати роботу або контактувати з потенційними роботодавцями безпосередньо.

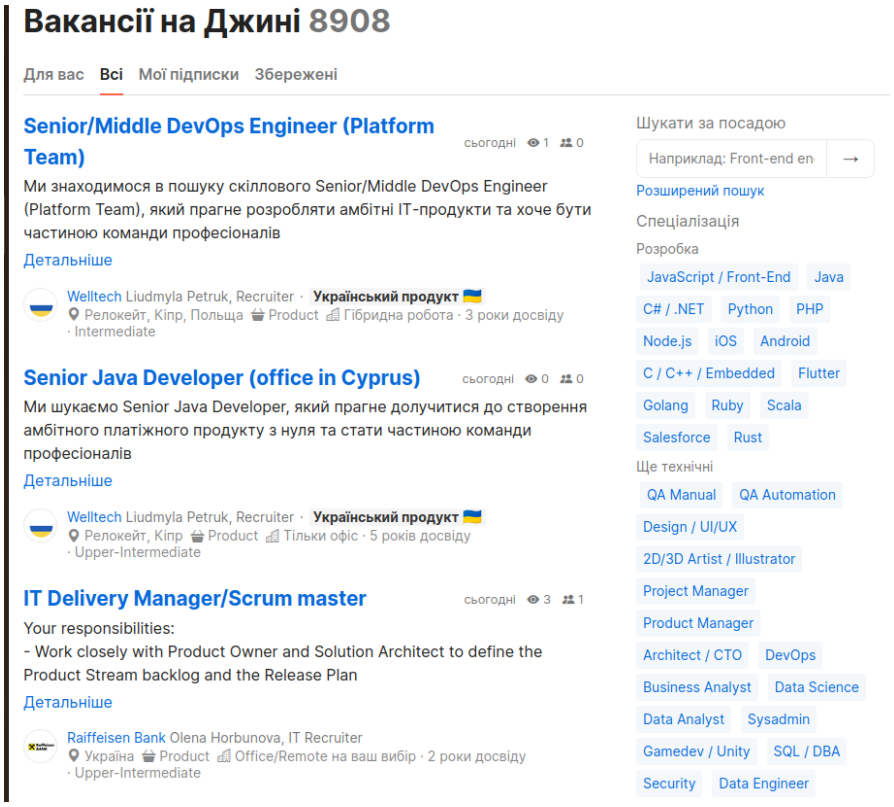


Рис. 1.5. Сторінка пошуку вакансій в Djinni [32]

У табл. 1.3 наведено порівняльну характеристику аналогів щодо їх переваг та недоліків. Виокремлено основні категорії для порівняння, такі як стандартиризована класифікація вакансій, конкуренція серед кандидатів, перевірка на спам, агрегація оголошень та персоналізовані рекомендації.

Таблиця 1.3

Порівняння існуючих рішень

Назва	LinkedIn	Glassdoor	Indeed	Djinni
Стандартизована класифікація вакансій	Ні	Частково	Частково	Частково
Велика конкуренція серед кандидатів	Так	Так	Так	Так
Перевірка на спам	Ні	Ні	Ні	Ні
Агрегація оголошень з сайтів компаній	Ні	Ні	Ні	Ні
Персоналізовані рекомендації	Частково	Частково	Частково	Частково

Проаналізувавши існуючі платформи, можна зробити висновок, що жодна з них не займається агреграцію оголошень з сайтів компаній та не робить перевірку відповідності назв вакансії до її тексту (спам). Крім цього, система рекомендації часто дає недостатньо релевантні результати для користувачів. Саме ці проблеми має розв’язати розроблена система.

1.3. Перелік функціональних вимог до системи

Після аналізу існуючих рішень та наявних проблем (рис. 1.2), було визначено цілі, які було організовано у вигляді дерева цілей.

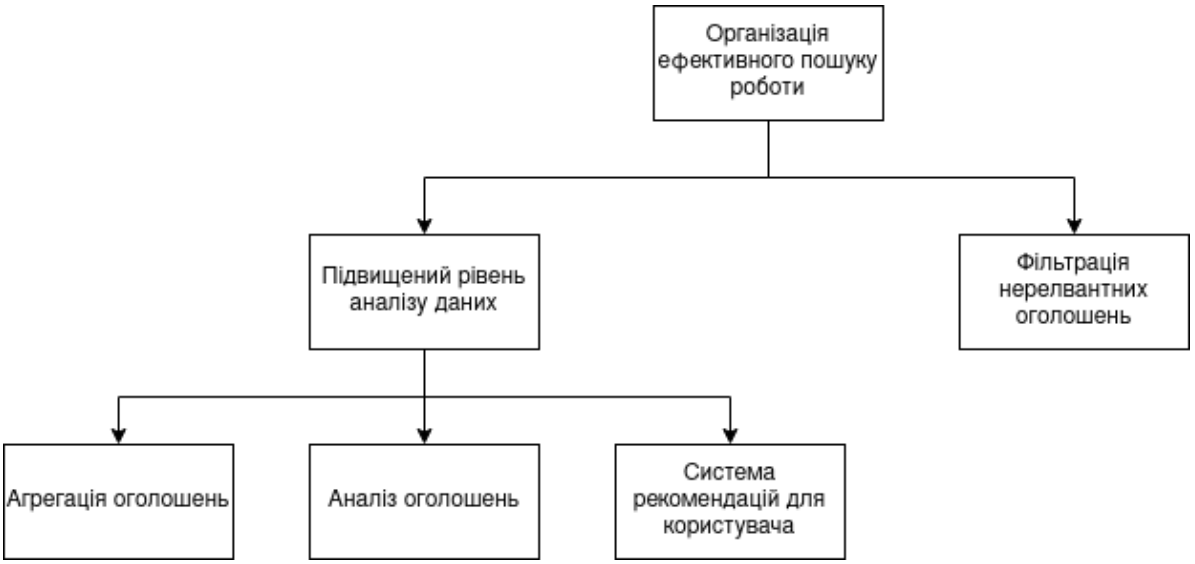


Рис. 1.2. Дерево цілей

Були сформовані наступні вимоги до програмного забезпечення:

1. Агрегація оголошень з різних джерел, зокрема з корпоративних сайтів компаній шляхом скрапінгу оголошень [35].
2. Аналіз агрегованих оголошень, перевірка відповідності назви вакансії, до її вмісту.
3. Побудова системи рекомендацій на основі уподобань користувача та проаналізованих оголошень.

Також в процесі аналізу було виділено вимоги до якості системи:

1. Система має легко масштабуватись для збільшення джерел агрегації оголошень.
2. Система має бути відмовостійкою.

Враховуючи вищезазначені пункти, стає можливим виділення наступних функціональних вимог до розроблюваної системи:

1. Система має надавати можливість реєстрації користувачів.
2. Система має надавати можливості для налаштування уподобань користувачів.
3. Система має агрегувати оголошення.
4. Система має аналізувати оголошення, та індексувати їх для движку рекомендацій.

5. Система має надавати можливості фільтрації оголошень.

З огляду на функціональні вимоги, було визначено нефункціональні вимоги до розробленого ПЗ:

1. Для забезпечення функціонування системи мають бути такі ролі користувачів: звичайний користувач та адміністратор.
2. Має бути створений окремий модуль для агрегації оголошень.
3. Має бути створений окремий модуль для аналізу даних.
4. Дані користувачів мають бути захищеними від несанкціонованого доступу.
5. Інтерфейс має бути відокремлений від серверної логіки та має слідувати сучасним нормам адаптивного дизайну.

1.4. Висновки до розділу 1

В цьому розділі було проаналізовано існуючі програмні рішення, на основі цього аналізу було сформовано дерево проблем та дерево цілей, які було уточнено переліком функціональних та нефункціональних вимог.

З головних конкурентних переваг існуючих рішень можна відзначити великий розмір самих платформ та існуючу базу шукачів та роботодавців, які звикли до означених платформ. З головних недоліків конкурентів можна виокремити відсутність агрегації оголошень, незадовільну якість движку рекомендацій та фільтрації оголошень. Саме на вирішення вказаних недоліків має зосередитися розроблюваний продукт.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування

Обрання мови програмування для проєкту або задачі може мати значний вплив на ефективність та результативність роботи. Вибір правильної мови програмування може допомогти збільшити швидкість розробки, зменшити кількість помилок, зробити код більш зрозумілим та підтримуваним. Крім того, деякі мови програмування спрощують роботу з певними типами даних або дозволяють розробляти певні види програмного забезпечення, які можуть бути неможливі або дуже складні з використанням інших мов.

2.1.1. *Python*

Python – інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою [4]. Розробники Python підтримують філософію “постачається разом з батарейками” – тобто має багату та гнучку стандартну бібліотеку, що надає Python перевагу у порівнянні з іншими мовами програмування [5]. Синтаксис Python є простим, не перевантаженим синтаксичними конструкціями, що покращує легкість підтримки коду. Окрім великої стандартної бібліотеки, Python має велику кількість прихильників, які створюють свої бібліотеки для цієї мови та розповсюджують їх, використовуючи open source ліцензії, що також сприяє простоті розробки на Python.

Оскільки Python є інтерпретованою мовою програмування, то їй також властивий загальний для більшості інтерпретованих мов програмування, компроміс, а саме пришвидшення швидкості розробки програми та полегшення її підтримки програмістом, в обмін на швидкодію та ефективність використання ресурсів комп’ютера [6].

Переваги [7]:

- Велика стандартна бібліотека.

- Велика кількість open source рішень, розроблених спільнотою.
- Зручний синтаксис.
- Простоа у підтримці програм.
- Відсутність процесу компіляції програми.

Недоліки [7]:

- Повільніший за компільовані мови програмування.
- Використовує більше оперативної пам’яті у порівнянні з іншими мовами програмування.
- Відсутність багатопоточності через Global Interpreter Lock.

Django – вебфреймворк, який позиціонує себе як інструмент для швидкого створення додатків. Одним з найпопулярніших проєктів, створених на Django є Instagram. Серед переваг Django слід зазначити вбудовану панель адміністратора та ORM, який захищає від SQL-ін’єкцій. Через велику кількість готових компонентів, та обширну екосистему Django, утворення MVP з цим фреймворком є швидким [15].

2.1.2. *Javascript*

Javascript є кросплатформною мовою програмування створеною для побудови динамічних вебсторінок [8]. Для розробки серверних частин вебсайтів існує вдосконалена версія Javascript – Node.js.

До стандартної бібліотеки Javascript входять основні структури даних, що використовуються у сучасному програмуванні, а саме рядки, динамічні списки, числа, дати. Javascript підтримує використання сторонніх бібліотек для розширення функціональних можливостей побудованих на цій мові програмування додатків.

Клієнтський JavaScript надає API для взаємодії з браузером та DOM-деревом. Саме завдяки такому використанню браузерного API Javascript надає можливість створювати динамічні вебсторінки.

Переваги:

- Краща швидкодія у порівнянні з іншими інтерпретованими мовами програмування (зокрема з Python) [9].
- Велика спільнота, що створює нові та підтримує старі бібліотеки для Javascript.
- Використання Javascript призводить до зменшення навантаження на веб сервер, оскільки він виконуються на стороні клієнта.

Недоліки:

- Браузери можуть по різному інтерпретувати код Javascript, що може призвести до неочікуваних помилок.
- Не типова об’єктна модель, яка суттєво відрізняється від її реалізації в мовах C#, Java, Python.

Опис фреймворку Angular: Angular – це відкритий фреймворк для розробки вебдодатків, створений компанією Google. Angular використовується для створення односторінкових додатків (SPA) та додатків на основі компонентів.

Переваги фреймворку Angular:

1. Кросс-платформеність: Angular дозволяє розробляти додатки, які працюють на різних платформах, включаючи веб, мобільні пристрої та настільні комп'ютери.
2. Модульність: Angular дозволяє створювати додатки за допомогою компонентів та модулів, що дозволяє ефективно організовувати код та робити додатки більш масштабованими.
3. Документація та підтримка: Angular має дуже добре написану документацію та активну спільноту, що дозволяє розробникам швидко знайти рішення для своїх проблем та питань.
4. Високий рівень безпеки: Angular має вбудовані функції безпеки, які дозволяють захищати додатки від різних видів атак, таких як XSS та CSRF.

Недоліки фреймворку Angular:

1. Велика складність: Angular має дуже велику кількість можливостей, що може зробити процес розробки досить складним.
2. Неоптимальна продуктивність: Angular може працювати дещо повільніше, ніж його прямий конкурент – React.js.
3. Вимоги до ресурсів: Angular вимагає відносно багато ресурсів, що може зробити додатки менш доступними на старіших або менш потужних пристроях.

2.1.3. *Java*

Java – це високорівнева, об'єктно-орієнтована мова програмування, яка була розроблена в 1995 році компанією Sun Microsystems (зараз – Oracle Corporation). Ось кілька переваг та недоліків мови програмування Java [10]:

Переваги:

- Переносимість: Java-програми можуть запускатися на різних операційних системах, оскільки вони компілюються в байт-код, який може бути виконаний на будь-якій машині з встановленою віртуальною машиною Java (JVM).
- Безпека: Java має вбудований механізм безпеки, що дозволяє виконувати код в ізольованому середовищі. Виконання коду може бути обмежено до певних функцій та доступу до файлової системи інших додатків, що забезпечує захист від вірусів та зловмисного ПЗ.
- Об'єктно-орієнтований підхід: Java підтримує концепції ООП, такі як інкапсуляція, наслідування та поліморфізм. Це дозволяє програмістам легко розробляти складні програми, які можуть бути розширені та модифіковані у майбутньому.

- Багатофункціональність: Java має велику кількість стандартних бібліотек та фреймворків, що дозволяє програмістам розробляти програми зі складною функціональністю, таку як робота з мережами, базами даних, графікою та іншими складними операціями.

Недоліки:

- Швидкодія: Java є повільнішою, ніж інші компільовані мови програмування, оскільки код повинен спочатку компілюватися в байт-код, а потім виконуватися на віртуальній машині.
- Великий розмір: програми на Java можуть мати більший обсяг коду і займати більше місця в пам'яті та на диску порівняно з іншими мовами програмування.
- Потребує великої кількості пам'яті: програми на Java можуть вимагати великої кількості пам'яті для запуску, що може бути проблемою для пристроїв з обмеженими ресурсами, таких як мобільні телефони або вбудовані системи.
- Складність: Java може бути складною мовою програмування, оскільки має велику кількість функціональних можливостей.

2.1.4. *C#*

C# (вимовляється як "Сі шарп") – це об'єктно-орієнтована мова програмування, створена компанією Microsoft для розробки програмного забезпечення для платформи .NET. C# була створена у 2000 році як заміна C++ і з тих пір стала однією з найпопулярніших мов програмування [11].

Переваги мови програмування C#:

- Об'єктно-орієнтована мова програмування: C# дозволяє розробляти дуже організовані та модульні програми за допомогою об'єктно-орієнтованого підходу, що дозволяє ефективно управляти великими проєктами.

- Інтегроване середовище розробки: C# інтегрується з Visual Studio – потужне інтегроване середовище розробки (IDE), яке дозволяє розробляти, тестувати та налагоджувати програми на різних платформах.
- Висока швидкість: C# компілюється в машинний код, що дозволяє розробляти дуже швидкі програми.
- Підтримка мультиплатформеності: C# програми можуть працювати на різних платформах, таких як Windows, Linux та macOS, завдяки розширенню .NET Core.
- Багата стандартна бібліотека: C# має велику кількість готових компонентів та бібліотек, що дозволяє розробляти програми швидше та ефективніше.

Недоліки [12]:

- Збільшене використання ресурсів у порівнянні з іншими компільованими мовами програмування.
- Повільніший процес розробки та ускладнена підтримка програм через компільовану природу мови.

Таблиця 2.1

Порівняння мов програмування Python, JavaScript, Java, C#

Назва	Python	JavaScript	Java	C#
Швидкість розробки	Так	Так	Частково	Частково
Простота підтримки	Так	Частково	Частково	Частково
Гнучкість використання синтаксису	Так	Частково	Ні	Ні
Підтримка ООП	Так	Частково	Ні	Ні

2.2. Вибір Системи Керування Базою Даних

2.2.1. *PostgreSQL*

PostgreSQL – потужна, open source об’єктно-реляційна система керування базами даних, що активно розробляється та підтримується більше 35 років, та має репутацію надійної, наповненої функціональними можливостями та високопродуктивної [13].

PostgreSQL має багату колекцію вбудованих типів даних, та можливість їх розширювати за допомогою додаткових бібліотек. Для PostgreSQL існують інтерфейси та ORM для усіх популярних мов програмування, а саме Python, JavaScript, Java, C# та інші.

Серед переваг PostgreSQL можна також зазначити підтримку написання сценаріїв для виконання на стороні базі даних, що дозволяє зменшувати кількість обчислень та запитів до БД на стороні розроблюваного додатку.. Основною мовою сценаріїв є PLpgSQL.

Розглянемо основні переваги та недоліки PostgreSQL.

Переваги:

- Opensource.
- Підтверджена великим часом в експлуатації надійність.
- Велика кількість вбудованих типів даних та функціональних можливостей при роботі з ними.
- Існують ORM для усіх популярних мов програмування.
- Документація, що повністю покриває усі можливості СУБД та має велику кількість прикладів.
- Активна спільнота, яка розробляє додаткові бібліотеки.
- Підтримується хмарними провайдерами (Amazon Web Services, Google Cloud, Microsoft Azure).

Недоліки:

- Потребує порівняно багато ресурсів комп’ютера для своєї роботи.

- Широкі функціональні можливості зумовлюють складність опанування цієї СУБД.

2.2.2. *MySQL*

MySQL – це реляційна база даних, яка використовує мову запитів SQL для управління даними. MySQL є однією з найпопулярніших баз даних у світі, і використовується для розробки різних типів вебдодатків та програмного забезпечення [14].

Переваги MySQL:

- Відкритий код: MySQL є відкритою базою даних, що дозволяє розробникам змінювати та розширювати її функціональність.
- Швидкість: MySQL має швидкість обробки даних та швидкий відгук на запити, що дозволяє забезпечити високу продуктивність додатків.
- Масштабованість: MySQL підтримує горизонтальне масштабування, що дозволяє збільшувати потужність бази даних з допомогою додавання нових серверів у кластер.
- Надійність: MySQL забезпечує високий рівень надійності та стійкості бази даних.
- Простота використання: MySQL має простий і зрозумілий інтерфейс користувача, що дозволяє з легкістю виконувати різні операції з даними.

Недоліки MySQL:

- Обмеження на швидкість: У випадку з великими обсягами даних, MySQL може стати повільнішим у порівнянні з PostgreSQL.
- Брак гнучкості: MySQL вимагає використання жорсткої схеми даних, що обмежує гнучкість у зберіганні даних.
- Обмеження на масштабування: У випадку з великими обсягами даних, масштабування MySQL може бути складним та обмеженим.

- Брак підтримки JSON: MySQL не має повної підтримки для зберігання даних у форматі JSON.

2.2.3. *MongoDB*

MongoDB – це документ-орієнтована NoSQL база даних, яка використовує JSON-подібні документи з динамічними схемами. MongoDB є досить популярним рішенням для зберігання даних в вебдодатках та інших проєктах, що вимагають гнучкої схеми даних та високої продуктивності [15].

Основні переваги MongoDB:

- Гнучкість: MongoDB дозволяє зберігати дані у вигляді документів, що дозволяє зберігати дані будь-якої структури.
- Масштабованість: MongoDB розроблена з урахуванням масштабування та підтримує розподілені системи.
- Швидкість: MongoDB має швидкість обробки даних через використання внутрішніх індексів та оптимізацію запитів.
- Геопросторовий пошук: MongoDB підтримує геопросторові запити, що дозволяє зберігати та шукати дані за географічними координатами.
- Відкритий код: MongoDB є відкритою базою даних, що дозволяє розробникам змінювати та розширювати її функціональність.

Деякі з недоліків MongoDB:

- Великий обсяг пам'яті: MongoDB вимагає значно більше пам'яті в порівнянні з реляційними базами даних.
- Складність в управлінні даними: MongoDB не має засобів для запобігання дублювання даних, що може стати проблемою при зберіганні даних великого обсягу.
- Брак транзакцій: MongoDB не підтримує транзакції на кшталт реляційних баз даних, що може бути проблемою в деяких сценаріях.

2.2.4. *ElasticSearch*

ElasticSearch є потужною та розширюваною пошуковою системою та аналітичним інструментом, який забезпечує збереження, пошук та аналіз великих обсягів даних у реальному часі. Він побудований на основі Apache Lucene і забезпечує можливість швидкого і гнучкого пошуку, індексації та аналізу структурованих і неструктурованих даних.

Переваги ElasticSearch:

1. Потужний пошук: ElasticSearch надає широкий спектр можливостей для розширеного пошуку, включаючи повнотекстовий пошук, пошук з підтримкою розширеного синтаксису запитів та підтримку мовних аналізаторів.
2. Гнучкість: ElasticSearch має гнучку схему, що дозволяє легко змінювати структуру даних без необхідності перезапуску системи або переіндексації даних.
3. Горизонтальне масштабування: ElasticSearch дозволяє розподіляти дані по кластеру серверів, що дозволяє обробляти великі обсяги даних та запитів швидко та ефективно.
4. Аналітика у реальному часі: ElasticSearch дозволяє проводити агрегацію даних та виконувати аналітичні запити в реальному часі, що допомагає отримувати швидкі та актуальні результати.

Недоліки ElasticSearch:

1. Складність налаштування: ElasticSearch може бути складним у налаштуванні та керуванні, особливо для менш досвідчених користувачів. Існує велика кількість налаштовуваних параметрів, які можуть впливати на продуктивність та результати пошуку.
2. Вимоги до ресурсів: Великі обсяги даних та інтенсивне використання можуть вимагати значних обчислювальних та мережевих ресурсів. ElasticSearch може вимагати потужну апаратну інфраструктуру для забезпечення ефективної роботи.

3. Синхронізація даних: ElasticSearch немає вбудованого механізму для синхронізації даних між різними кластерами. Це може стати проблемою при роботі з розподіленими системами, де вимагається синхронізація даних в реальному часі.
4. Нестабільність: Незважаючи на широке застосування та активне співтовариство, ElasticSearch може мати проблеми зі стабільністю та виявляти деякі помилки, особливо під час масштабування та обробки великих обсягів даних.

Враховуючи перелічені переваги і недоліки, ElasticSearch залишається потужним та корисним інструментом для пошуку та аналізу даних.

Таблиця 2.2

Порівняння СУБД

Назва	PostgreSQL	MySQL	MongoDB	ElasticSearch
Реляційна	Так	Так	Ні	Ні
Продуктивність	Частково	Так	Частково	Частково
Гнучкість	Частково	Обмежена гнучкість	Так	Так
Повнотекстовий пошук	Частково	Частково	Ні	Так

2.3. Вибір середовища розробки програмного забезпечення

Для розробки системи було обрано середовище PyCharm. PyCharm – це інтегроване середовище розробки (IDE) для мови програмування

Python. Воно розроблене компанією JetBrains, яка спеціалізується на розробці інструментів для розробників [17].

Переваги середовища PyCharm:

- Підтримка Python: PyCharm забезпечує повну підтримку мови програмування Python, включаючи версії 2.x та 3.x. Воно також підтримує багато додаткових бібліотек та фреймворків Python.
- Інтелектуальний редактор коду: PyCharm має потужний редактор коду, який надає інтелектуальні функції, такі як автодоповнення коду, перевірка помилок та аналіз коду.
- Відладка: PyCharm надає розширену підтримку відладки, що дозволяє знайти та виправити помилки в програмі.
- Управління проєктами: PyCharm надає широкі можливості для управління проєктами, включаючи вбудовану систему контролю версій та можливість збирання та розгортання додатків.
- Підтримка різних платформ: PyCharm працює на різних платформах, включаючи Windows, macOS та Linux.

Недоліки середовища PyCharm:

- Висока вартість: PyCharm є комерційним продуктом, і це може зробити його вартість досить високою для окремих розробників.
- Великі вимоги до ресурсів: PyCharm вимагає відносно багато ресурсів, що може зробити його менш доступним на менш потужних комп'ютерах.
- Великий обсяг: PyCharm має досить великий обсяг, що може зайняти багато місця на диску.
- Неінтуїтивний інтерфейс: Інтерфейс PyCharm може бути дещо складним для новачків.

2.4. Вибір алгоритму класифікації вакансії

2.4.1. *Multinomial Bayes Algorithm*

Multinomial Bayes Algorithm є статистичним алгоритмом машинного навчання, використовуваним для класифікації текстових документів. Він базується на Теоремі Баєса та моделі мультиноміального розподілу ймовірностей. Алгоритм припускає, що кожен документ представляється як вектор, де кожне значення відображає частоту входження терміну у текст [28].

Переваги:

1. Простота: Мультиноміальний баєсовий алгоритм є простим у реалізації та зрозумілим для розуміння. Він не вимагає складних налаштувань та тренування моделі.
2. Ефективність: Алгоритм працює швидко, навіть при великій кількості документів та великому словнику термінів. Він є ефективним у використанні ресурсів та швидкому класифікуванні.
3. Добре справляється з великими наборами даних: Multinomial Bayes Algorithm показує хороші результати при класифікації текстів навіть у випадку великої кількості документів та великому словнику.
4. Добре працює з розрідженими даними: Алгоритм може ефективно працювати з розрідженими даними, де багато термінів мають нульові значення.

Недоліки:

1. Алгоритм ґрунтується на наївному припущенні про незалежність термінів, що може бути не реалістичним для деяких текстових даних. Це може вплинути на точність класифікації.

2. Втрата контексту: Алгоритм не враховує контекст або послідовність слів у тексті, що може призвести до втрати важливої інформації при класифікації.
3. Проблема невідомих слів: Алгоритм може мати проблеми з новими словами, які не були побачені під час тренування моделі. Він не вміє навчатися на ходу та потребує оновлення моделі для врахування нових слів.
4. Не враховує семантику: Multinomial Bayes Algorithm не аналізує семантичні зв'язки між словами. Він працює на основі частоти входження слів у текст, ігноруючи їх семантичне значення.

Враховуючи перелічені переваги і недоліки, Multinomial Bayes Algorithm є простим та ефективним алгоритмом для класифікації текстів, але його результати можуть бути обмежені неврахуванням семантики та наївним припущенням про незалежність термінів.

2.4.2. *Rule Based Text Classification*

Rule-based text classification використовує набір правил або умов, щоб визначити класифікацію тексту. Цей підхід ґрунтується на заданих правилах, які визначають, який клас буде присвоєний текстовому документу на основі його властивостей або характеристик [33].

Переваги:

1. Простота і зрозумілість: Rule-based text classification є легким у розумінні та реалізації підходом. Правила можуть бути визначені та налаштовані людиною без необхідності в складних алгоритмах машинного навчання.
2. Інтерпретованість: Результати класифікації на основі правил легко інтерпретувати, оскільки вони базуються на зрозумілих умовах і властивостях тексту. Це може бути корисним, коли потрібно пояснити або виправдати результати класифікації.

3. Наявність експертних знань: Rule-based підхід дозволяє використовувати експертні знання про домен або галузь, щоб визначити правила класифікації. Це може бути корисним, коли експерти мають глибоке розуміння текстових даних та їх класифікації.

Недоліки:

1. Обмежена гнучкість: Rule-based підхід може бути обмеженим у визначенні складних залежностей або врахуванні непередбачуваних шаблонів у тексті. Він не може навчитися на основі даних або адаптуватися до нових шаблонів, як це роблять алгоритми машинного навчання.
2. Велика кількість правил: Для покриття різних випадків класифікації може знадобитися велика кількість правил, що вимагає багато ручної роботи для їх створення та підтримки.
3. Неврахування семантики: Правила в основному базуються на поверхневих характеристиках тексту і можуть не враховувати семантичні зв'язки між словами або реченнями. Це може призводити до втрати важливої інформації під час класифікації.

Хоча rule-based text classification має свої переваги в простоті та інтерпретованості, варто враховувати його обмежену гнучкість та потенційну втрату семантичної інформації. Залежно від конкретних потреб та характеристик даних, інші методи класифікації тексту можуть бути більш ефективними.

2.4.3. *Висновки до вибору алгоритму класифікації*

Зважаючи на означені переваги та недоліки наведених методів, було прийнято рішення віддати перевагу класифікатору на основі Multinomial Bayes алгоритму, оскільки він є більш простим у реалізації, ніж Rule Based класифікатор. В той же час, такий класифікатор дає хороші результати на практиці, і є простішим ніж інші методи класифікації текстів. Для

реалізації обраного алгоритму було обрано бібліотеку Scikit Learn [28], яка є де-факто стандартом для навчання текстових класифікаторів.

2.5. Висновки до розділу 2

Зважаючи на результати проведеного аналізу, було визначено, що для реалізації функціональних вимог проєкту найоптимальнішою мовою програмування для реалізації серверної частини є Python у комбінації з фреймворком Django. Такий вибір зумовлений необхідністю швидко розробити MVP продукту, для чого Python та Django має більше переваг у порівнянні з іншими мовами програмування. В даному випадку підвищене використання ресурсів комп'ютера не є проблемою, а підвищена швидкість розробки є помітною перевагою.

Для реалізації клієнтської частини було обрано мову програмування JavaScript у комбінації з фреймворком Angular. Цей вибір зумовлено де-факто монополією JavaScript при побудові інтерактивних вебсторінок. Перевагою фреймворку Angular є велика кількість готових компонентів, завдяки яким можна швидко побудувати необхідні вебсторінки.

Оскільки для реалізації серверної частини додатку було обрано Django, то вибір СУБД звужується до реляційних, оскільки Django погано підтримує роботу з NoSQL базами даних. При виборі між PostgreSQL та MySQL перевагу отримує PostgreSQL, оскільки ця СУБД є більш гнучкою ніж MySQL.

В якості додаткової бази даних для побудови системи рекомендацій було обрано ElasticSearch, оскільки її можливості повнотекстового пошуку значно кращі ніж у наведених базах даних, але для інших задач ця база даних є значно гіршою, ніж наведені.

Для побудови класифікатора вакансій було вирішено побудувати Naive Bayes текстовий класифікатор, оскільки він поєднує в собі простоту та ефективність використання на практиці. Для побудови класифікатора було обрану бібліотеку Scikit Learn.

3. РОЗРОБЛЕННЯ АРХІТЕКТУРИ ТА ПРОГРАМНИХ МОДУЛІВ

3.1. Загальний опис архітектури системи

Після аналізу функціональних вимог, сформульованих у 1 розділі, для реалізації системи було обрано клієнт-серверну архітектуру, з serverless модулем на основі AWS Lambda для утворення рекомендації користувачам. Спрощене представлення архітектури ПЗ можна побачити на рис. 3.1.

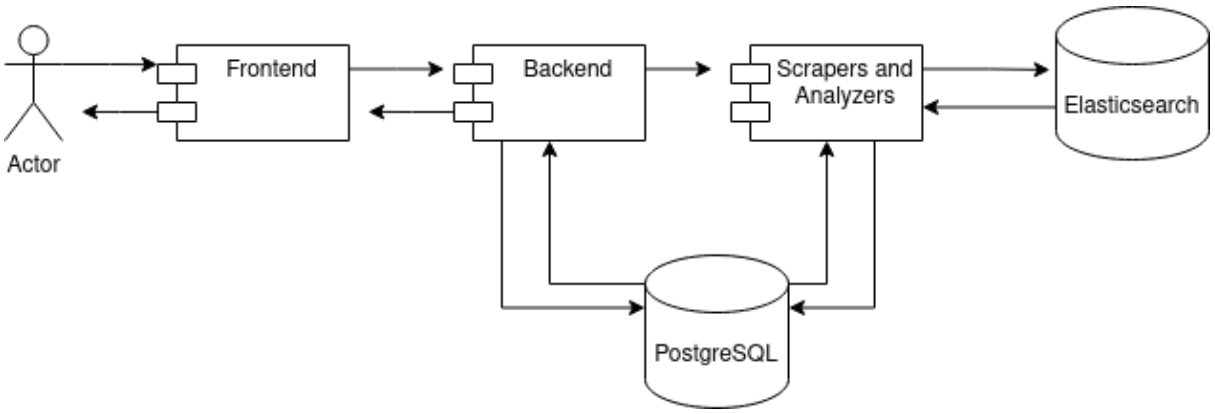


Рис. 3.1. Спрощене представлення архітектури ПЗ

Клієнт-серверна архітектура – це модель обчислення, в якій сервер забезпечує хостинг, доставку та управління більшістю ресурсів та сервісів для використання клієнтом. У цьому типі архітектури один або декілька комп'ютерів-клієнтів підключаються до центрального сервера через мережу або Інтернет-з'єднання. Ця система спільно використовує обчислювальні ресурси. Клієнт-серверна архітектура також відома як мережева обчислювальна модель або клієнт-серверна мережа, оскільки всі запити та сервіси передаються через мережу.

Існує декілька основних підходів до проєктування структури додатку для забезпечення клієнт-серверної архітектури. Це трирівнева архітектура та Model-View-Controller.

3.1.1. *Трирівнева архітектура*

Трирівнева архітектура допомагає виокремити роботу, що виконується на різних рівнях, тим самим дотримуватися принципу розподілу обов’язків та інкапсуляції. В загальному випадку, трирівнева архітектура має наступну структуру (рис. 3.2).

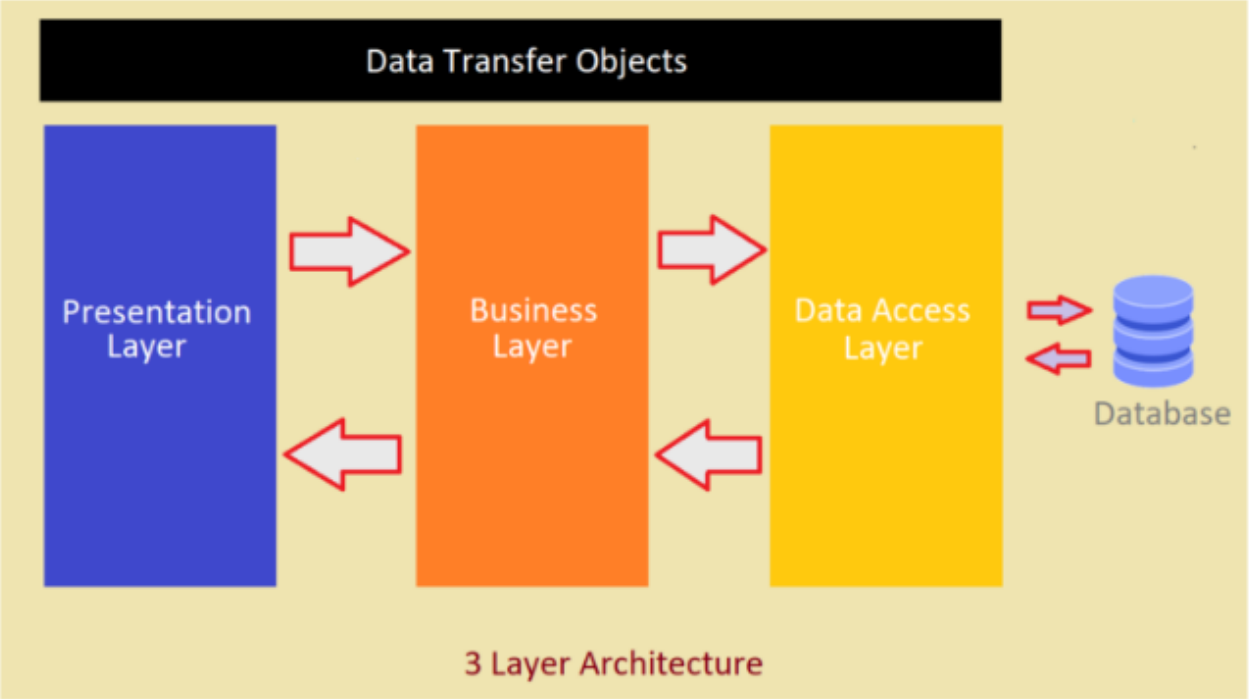


Рис. 3.2. Трирівнева архітектура

Data Access Layer

Нижчий рівень, відомий як "рівень передачі даних" або "data transfer layer", відповідає за додавання, оновлення, читання та видалення даних з бази даних. Також на цьому рівні формуються об'єкти з даними, які використовуються на вищих рівнях архітектури. Використовуються різноманітні SQL-запити або використовується ORM. У випадку використання об'єктно-реляційного відображення створюються спеціальні моделі, що повністю відповідають таблицям у базі даних та мають відповідні поля. Це дозволяє створювати об'єкти, які передають дані в додатку. Очевидно, розробникам зручніше працювати з ORM, оскільки

багато сучасних фреймворків виконують комунікацію з базою даних, уникнувши необхідності в більшості випадків використовувати мову запитів. Проте, в разі потреби отримання складної структури даних автоматично згенерований запит може бути значно менш швидким порівняно з запитом, написаним вручну людиною.

Business Logic Layer

Наступним рівнем в розглянутій архітектурі є рівень бізнес-логіки додатку, який відповідає за логіку розроблюваного вебдодатку. На цьому рівні реалізується функціональні вимоги до програмного забезпечення, проводиться валідація інформації, що надходить у запитах від клієнтів, і викликаються методи для створення об'єктів з обробленими даними та їх подальшого збереження у базі даних [19].

Наприклад, коли надходить запит на відображення певної інформації від клієнта, на рівні бізнес-логіки перевіряється, чи має користувач необхідні права доступу до запитуваних даних. У разі відсутності необхідних прав доступу клієнту відмовляють у доступі до бази даних, і рівень взаємодії з клієнтом формує відповідне повідомлення про це. Якщо користувач має необхідні права доступу, викликаються методи на рівні передачі даних, які звертаються до бази даних та повертають очікувану інформацію. Далі об'єкт-носій даних передається на рівень відображення даних, де на його основі створюється відповідь, яка передається клієнту.

Presentation Layer

Найвищим рівнем у трьохрівневій архітектурі є рівень презентації даних, де знаходяться необхідні методи взаємодії з клієнтом. На цьому рівні створюються та обробляються запити, і від нього залежить інтерфейс та доступна користувачу функціональність. Методи на цьому рівні передають дані, що надійшли від клієнта, на рівень бізнес-логіки або формують відповідь. Формат запитів та відповідей залежить від побудови програмного забезпечення. Сервер може відправляти дані у вигляді рядків у форматі JSON, які будуть інтерпретовані фронтенд-додатком, або

створювати і відправляти повноцінні HTML-сторінки, що будуть негайно показані користувачу [20].

3.1.2. *Model-View-Controller*

Шаблон Model-View-Controller (MVC) є архітектурним шаблоном, який використовується для організації структури програмного забезпечення. Він розділяє компоненти додатку на три основні частини: Модель (Model), Представлення (View) та Контролер (Controller). Кожна з цих частин виконує свою відповідну роль і має свої функції.

1. Модель (Model): модель представляє бізнес-логіку додатку і відповідає за обробку даних, взаємодію з базою даних та збереження стану додатку. Вона не залежить від інших компонентів і може існувати незалежно. Модель включає в себе структури даних, функції та методи, які дозволяють здійснювати операції над даними.
2. Представлення (View): представлення відповідає за відображення даних користувачу. Воно відображає інформацію, яка надходить від Моделі, у зручному для користувача форматі. Представлення може бути графічним інтерфейсом, вебсторінкою, консольним виводом або будь-яким іншим способом відображення даних користувачу.
3. Контролер (Controller): контролер взаємодіє з користувачем та обробляє його дії. Він приймає вхідні дані від користувача через Представлення і виконує відповідні дії. Контролер також взаємодіє з Моделлю, оновлюючи або отримуючи необхідні дані. Він координує взаємодію між Моделлю та Представленням.

Основна ідея шаблону MVC полягає у відокремленні логіки додатку (Модель) від його відображення (Представлення) та управління взаємодією з користувачем (Контролер). Це дозволяє досягти більшої модульності, розширюваності та повторного використання коду. Крім того, цей шаблон

дозволяє розробникам працювати над різними компонентами додатку незалежно один від одного, що покращує роботу в команді та забезпечує більшу стабільність програмного забезпечення [21].

3.2. Структура бази даних

Для ефективної роботи з даними у СУБД PostgreSQL необхідно побудувати структуру бази даних та нормалізувати її. Якщо схема БД недостатньо продумана, це призводить до написання складних та повільних запитів, а при збільшенні кількості даних буде зменшуватись швидкодія додатку [22].

На рис. 3.3 можна ознайомитись з повною схемою бази даних запропонованого програмного забезпечення.

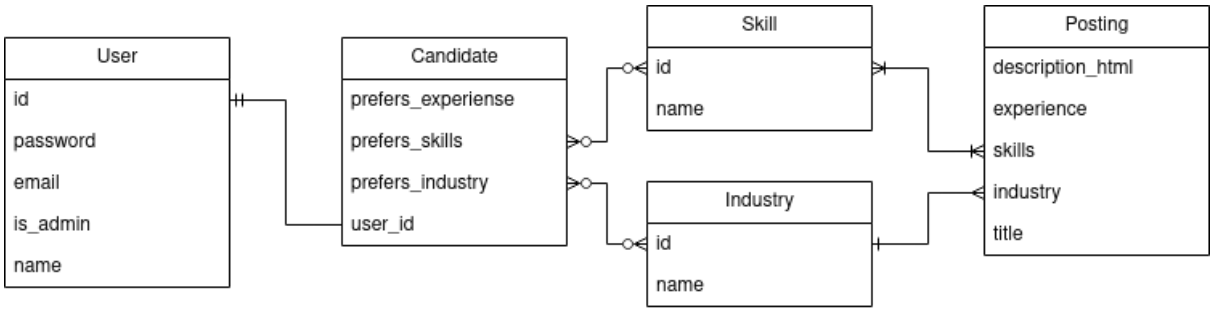


Рис. 3.3. Спрощена схема бази даних додатку

User

1) username: логін користувача (тип даних: рядок).
2) password: зашифрований пароль користувача (тип даних: рядок).
3) email: електронна адреса користувача (тип даних: рядок).
4) first_name: ім'я користувача (тип даних: рядок).
5) last_name: прізвище користувача (тип даних: рядок).
6) date_joined: дата реєстрації користувача (тип даних: дата та час).
7) is_active: прапорець, що вказує, чи є користувач активним (тип даних: булеве значення).

8) is_staff: прапорець, що вказує, чи має користувач доступ до адміністративної панелі (тип даних: булеве значення).
9) is_superuser: прапорець, що вказує, чи є користувач суперкористувачем з повними правами (тип даних: булеве значення).

Ці поля дозволяють зберігати основні дані про користувача, такі як логін, пароль, електронну адресу, ім'я, прізвище, а також додаткову інформацію про статус, доступ і реєстрацію.

Candidate

1) user: зв'язок один до одного з моделлю User з використанням зовнішнього ключа (тип даних: User).
2) prefers_min_exp: найменша бажана кількість років досвіду (тип даних: ціле число). Це поле може бути пустим (blank=True) або null (null=True).
3) prefers_max_exp: найбільша бажана кількість років досвіду (тип даних: ціле число). Це поле може бути пустим (blank=True) або null (null=True).
4) prefers_skills: зв'язок багато до багатьох з моделлю Skill через проміжну таблицю (тип даних: ManyToManyField). Це поле дозволяє зберігати багато навичок, які кандидат вважає бажаними.
5) prefers_industries: зв'язок багато до багатьох з моделлю Industry через проміжну таблицю (тип даних: ManyToManyField). Це поле дозволяє зберігати багато галузей, які кандидат вважає бажаними.

Модель Candidate додатково використовує моделі Skill та Industry для зберігання інформації про бажані навички та галузі.

Company:

1) id: унікальний ідентифікатор компанії, який генерується автоматично при створенні запису (тип даних: UUIDField).

2) name: назва компанії (тип даних: CharField з максимальною довжиною 100 символів). Це поле не може бути порожнім (blank=False).
3) description: опис компанії (тип даних: CharField з максимальною довжиною 300 символів). Це поле може бути порожнім (blank=True).
4) board_url: URL-адреса до дошки оголошень компанії (тип даних: URLField). Це поле не може бути порожнім (blank=False).
5) logo_url: URL-адреса до логотипу компанії (тип даних: URLField). Це поле може бути порожнім (blank=True).

Модель Company зберігає інформацію про компанії, включаючи їх унікальний ідентифікатор, назву, опис, URL-адресу до дошки оголошень та URL-адресу до логотипу.

Industry

1) id: унікальний ідентифікатор галузі, який генерується автоматично при створенні запису (тип даних: UUIDField).
2) name: назва галузі (тип даних: CharField з максимальною довжиною 300 символів). Це поле не може бути порожнім (blank=False).

Skill

1) id: унікальний ідентифікатор навички, який генерується автоматично при створенні запису (тип даних: UUIDField).
2) name: назва навички (тип даних: CharField з максимальною довжиною 300 символів). Це поле не може бути порожнім (blank=False).

Модель Skill зберігає інформацію про навички, включаючи їх унікальний ідентифікатор та назву.

Модель Posting має наступні поля:

1) id: унікальний ідентифікатор постингу, який генерується автоматично при створенні запису (тип даних: UUIDField).

2) title: заголовок постингу (тип даних: CharField з максимальною довжиною 300 символів). Це поле не може бути порожнім (blank=False).
3) description_html: HTML-опис постингу (тип даних: TextField). Це поле не може бути порожнім (blank=False).
4) industry: зв'язок з моделлю Industry через зовнішній ключ (тип даних: ForeignKey). Кожен постинг пов'язаний з однією галуззю.
5) skills: зв'язок з моделлю Skill через багато-до-багатьох відношення (тип даних: ManyToManyField). Кожен постинг може мати декілька навичок, а навички можуть бути пов'язані з декількома постингами.
6) experience_min: мінімальний досвід роботи, необов'язкове поле (тип даних: IntegerField, може бути порожнім (blank=True) та має значення за замовчуванням null=True).
7) experience_max: максимальний досвід роботи, необов'язкове поле (тип даних: IntegerField, може бути порожнім (blank=True) та має значення за замовчуванням null=True).
8) url: URL-адреса постингу (тип даних: TextField). Це поле не може бути порожнім (blank=False) та має унікальне значення (unique=True).
9) company: зв'язок з моделлю Company через зовнішній ключ (тип даних: ForeignKey). Кожен постинг пов'язаний з однією компанією.

Модель Posting зберігає інформацію про відкриті вакансії, включаючи їх унікальний ідентифікатор, заголовок, HTML-опис, галузь, навички, мінімальним і максимальним досвідом роботи, URL-адресу та компанію, пов'язану з постингом.

Оскільки для створення додатку використовується Django, то для проєктування було обрано підхід Code First, який активно підтримується

фреймворком. Було побудовані ORM класи, які відповідають описаним моделям.

В якості моделі для User було використано стандартну модель Django [23].

```
class Candidate(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)

    prefers_min_exp = models.IntegerField(blank=True, null=True)
    prefers_max_exp = models.IntegerField(blank=True, null=True)

    prefers_skills = models.ManyToManyField(Skill, blank=True)
    prefers_industries = models.ManyToManyField(Industry, blank=True)
```

Рис. 3.4. Модель кандидата

У моделі можна побачити зв’язки один-до-одного з моделлю User, та один-до-багатьох з моделями індустрії та навичок. Зв'язок один-до-одного з моделлю User є рекомендованим способом розширення базової моделі користувача у Django [24]. Зв’язок один до багатьох з навичками та індустріями зумовлений тим що, Кандидат може мати декілька індустрій та навичок, за якими він шукає роботу.

```
class Company(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid4)
    name = models.CharField(max_length=100, blank=False)
    description = models.CharField(max_length=300, blank=True)
    board_url = models.URLField(blank=False)
    logo_url = models.URLField(blank=True)
```

Рис. 3.5. Модель компанії

```python
class Industry(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid4)
    name = models.CharField(max_length=300, blank=False)
```

Рис. 3.6. Модель Індустрії

```python
class Posting(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid4)
    title = models.CharField(max_length=300, blank=False)
    description_html = models.TextField(blank=False)
    industry = models.ForeignKey(Industry, blank=False, on_delete=models.CASCADE)
    skills = models.ManyToManyField(Skill)
    experience_min = models.IntegerField(blank=True, null=True)
    experience_max = models.IntegerField(blank=True, null=True)
    url = models.TextField(blank=False, unique=True)
    company = models.ForeignKey(Company, on_delete=models.CASCADE)
```

Рис. 3.7. Модель Вакансії

Модель Вакансії зберігає дані про вакансію, та дані, які було отримано шляхом аналізу тексту вакансії. Це список навичок, які потребуються для цієї вакансії, рівень досвіду та індустрія, до якої відноситься вакансії. Зв’язок багато до багатьох зумовлений тим, що в одній вакансії можуть вимагати декілька навичок, та однакові навички можуть вимагати у різних вакансіях.

```python
class Skill(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid4)
    name = models.CharField(max_length=300, blank=False)
```

Рис. 3.8. Модель навички

3.3. Модуль “Scrapers and Analyzers”

Модуль Scrapers and Analyzers було вирішено виокремити в окремий serverless сервіс. Такий підхід є проміжним між монолітною та мікросервісною структурою додатку. Scraper and Analyzers не є мікросервісом, оскільки він не має повністю відокремленої бази даних від основного додатку [25], але в той же час це окремий додаток, що допомагає підтримувати принцип розподілення обов’язків. Крім цього, тільки цей сервіс має доступ до бази даних ElasticSearch, що спрощує структуру основного додатку.

На рис. 3.9 наведено приклад архітектури мікросервісного додатку.

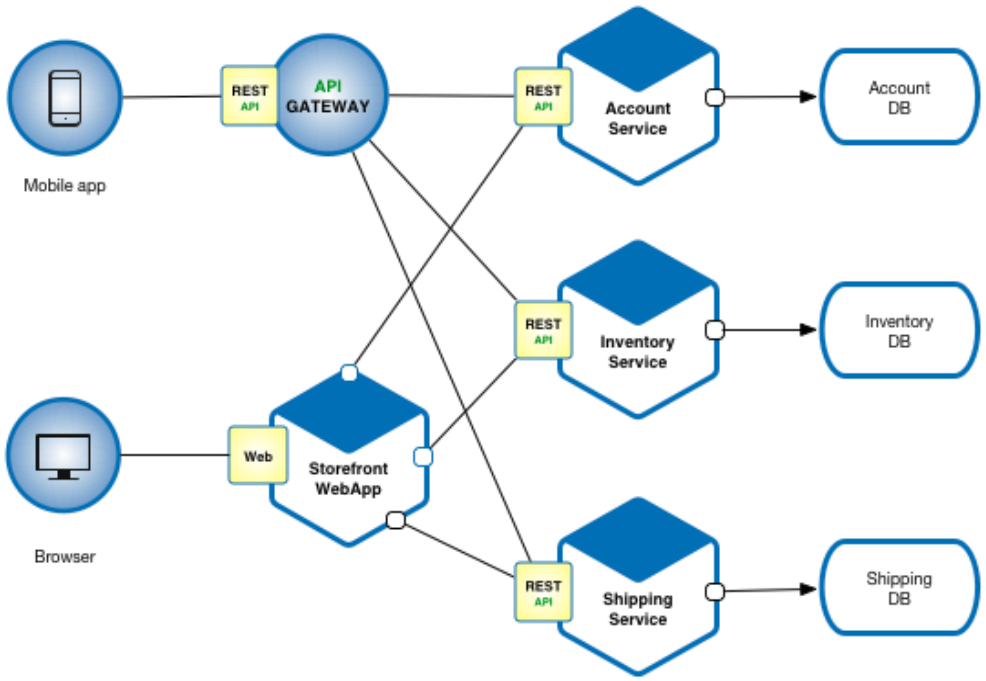


Рис. 3.9. Приклад архітектури мікросервісного додатку

В якості джерела вакансій було обрано платформу Lever, для якої було розроблено скрапер. Скраплені дані було проаналізовано за допомогою наївного байєсовського класифікатора, щоб визначити, до якої індустрії відноситься постинг. Крім цього, у тексті шукаються назви навичок та рівень досвіду, які додаються до проаналізованих даних. Після

закінчення класифікації, проаналізовані дані постингу індексуються в Elasticsearch.

Окремим endpoint цього модулю є endpoint рекомендацій, який отримує id користувача та будує запит до Elasticsearch згідно з його уподобаннями.

На рис. 3.10 наведено приклад архітектури serverless додатку.

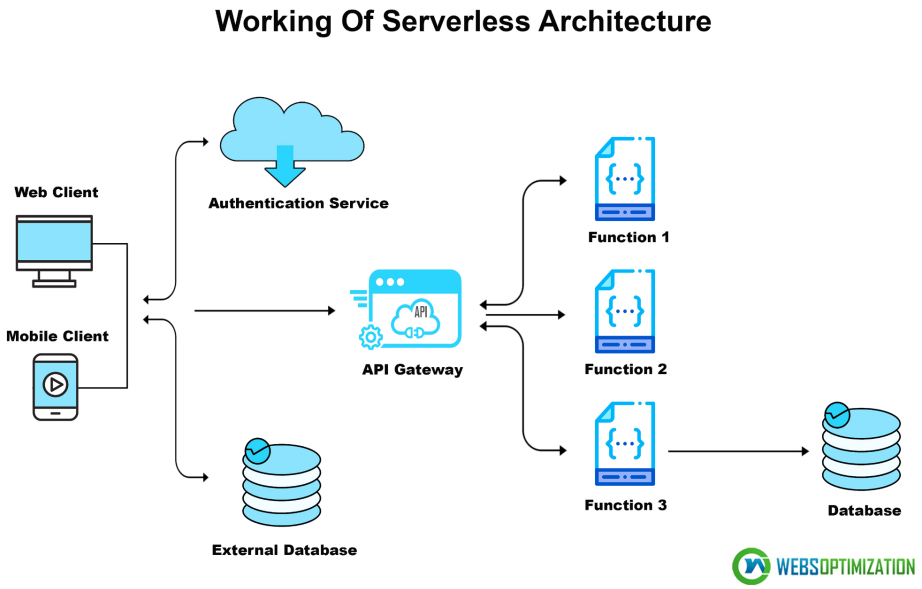


Рис. 3.10. Приклад архітектури додатку побудованого на основі serverless архітектури

3.4. Побудова наївного байєсовського класифікатора

Окремо від інших модулей, знаходиться модуль для побудови наївного баєсевського класифікатору, оскільки він використовується лише раз для тренування моделі, після чого він вже не є необхідним для виконання функціональних вимог до програми.

Для визначення індустрії, до якої належить вакансія, було обрано Multinomial наївний байєсівський класифікатор. В якості датасету була використана інформація про вакансії в Сполученому Королівстві [27].

3.5. Висновки до розділу 3

У даному розділі було проведено аналіз та порівняння популярних архітектур та описано структуру побудованого вебдодатку. Наведено структуру бази даних з поясненням логіки зв’язків між таблицями, структуру backend та serverless частини додатку. Було описано, яка зона відповідальності у кожного модуля.

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Тестування програмного забезпечення

Тестування додатків визначається як тип тестування програмного забезпечення, що проводиться за допомогою скриптів з метою виявлення помилок у програмному забезпеченні. Воно охоплює тести для всього додатка.

Це допомагає покращити якість додатків, знизити витрати, максимізувати прибутковість вкладень і заощадити час розробки. У сфері інженерії програмного забезпечення тестування додатків може бути виконано у різних категоріях, таких як GUI, функціональність, база даних (бекенд), тестування навантаження та інше. У випадку тестування додатків, життєвий цикл тестування включає різні фази, включаючи аналіз вимог, планування тестування, аналіз тестів, проєктування тестів, виконання тестів та звіт про помилки тощо [29].

4.2. Smoke Testing

Smoke Testing – це процес тестування програмного забезпечення, який визначає, чи є розгорнута збірка програмного забезпечення стабільною. Smoke Testing є підтвердженням для команди контролю якості (QA), щоб продовжити подальше тестування програмного забезпечення. Воно складається з мінімального набору тестів, які запускаються на кожній збірці для перевірки функціональності програмного забезпечення. Smoke Testing також відомий як "тестування підтвердження збірки" або "тестування перевірки наявності проблем". Простими словами, smoke-тести означають перевірку роботи важливих функцій та відсутності критичних проблем у збірці, яка перебуває на тестуванні. Це мініатюрне та швидке регресійне тестування основної функціональності. Це простий тест, який показує, що продукт готовий до тестування. Це допомагає

визначити, чи є збірка з недоліками, які роблять будь-яке подальше тестування марнотратством часу та ресурсів [30].

Для тестування функціональності розробленого вебдодатку було складено діаграми діяльності для кожної полі користувачів, щоб мати уявлення про критичні шляхи ПЗ.

Для того, щоб протестувати користувача, який має роль “Адміністратор” необхідно:

1. Зайти на сторінку входу до панелі адміністратора та надати необхідні для авторизації дані.
2. Провести CRUD операції над даними.
3. Провести адміністративні дії над користувачами, а саме активація/деактивація, змінити ролі.

Для тестування Користувача, необхідно (рис. 4.2):

1. Зайти на сторінку входу до системи та зареєструватися.
2. Авторизуватися.
3. Переглянути наявні оголошення про роботу.
4. Перевірити, що пагинация по оголошенням коректно працює.
5. Перейти на сторінку зміни уподобань.
6. Перевірити CRUD дії над уподобаннями.
7. Перейти на сторінку оголошень, та перевірити отримку персоналізованих оголошень.
8. Перевірити, що рекомендовані оголошення дійсно відповідають обраним уподобанням.

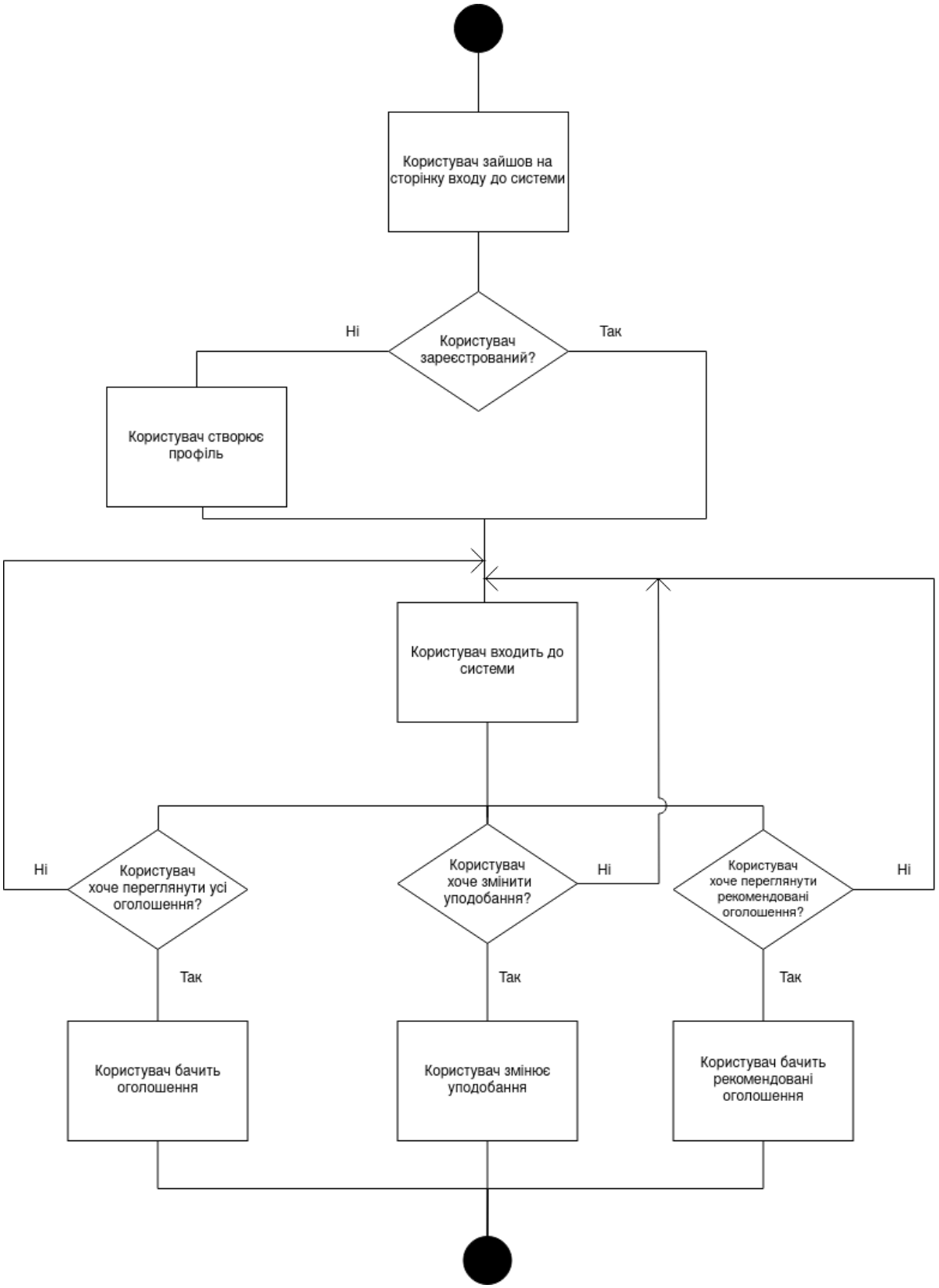


Рис. 4.2. Діаграма діяльності користувача

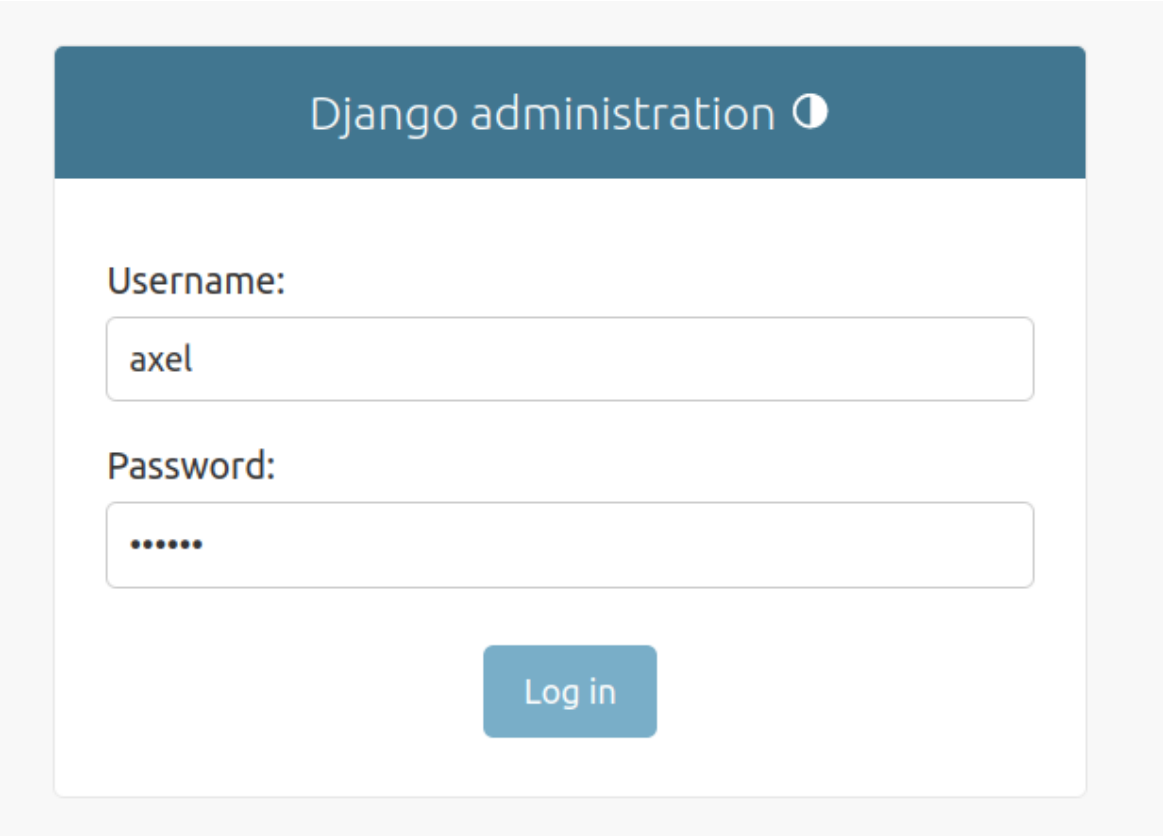


Рис. 4.3. Вхід до адмін панелі додатку

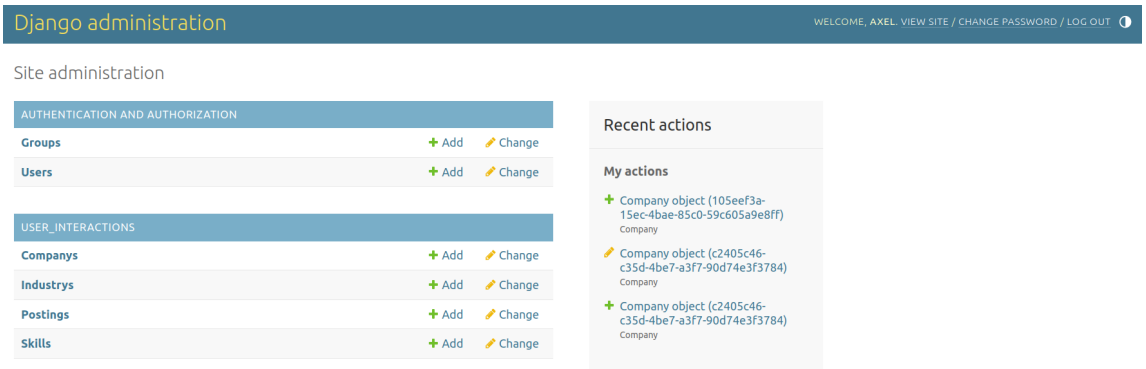


Рис. 4.4. Панель керування даними

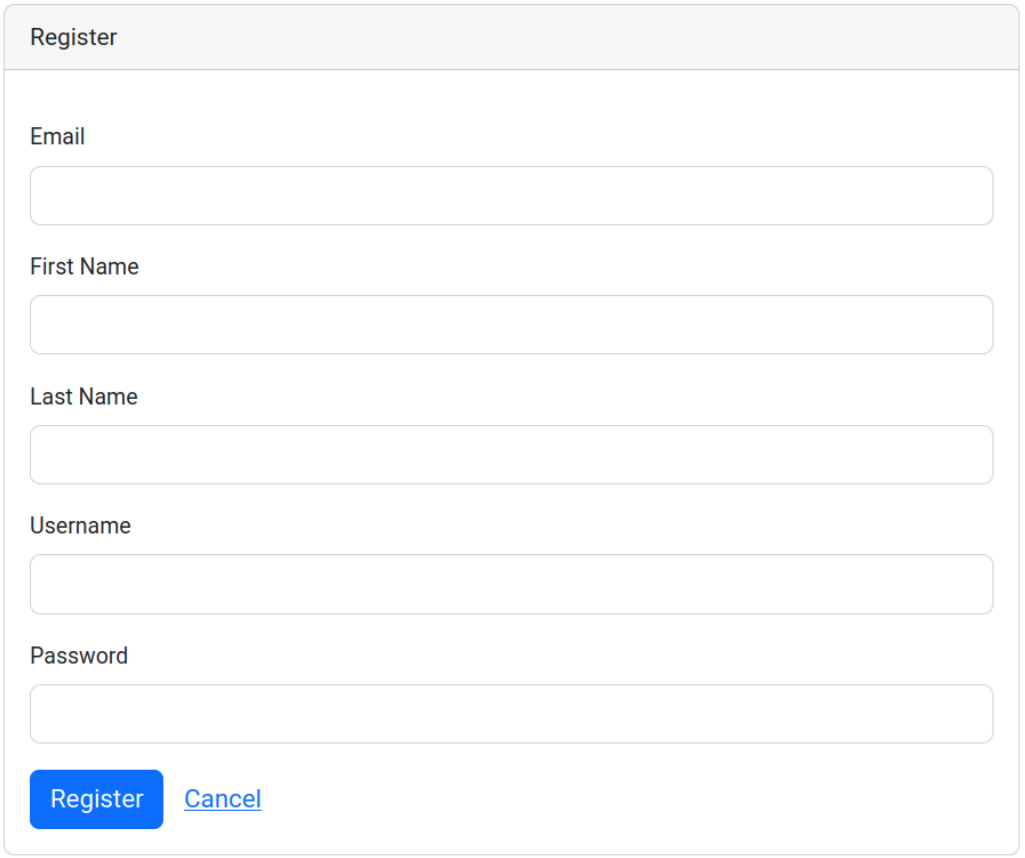


Рис. 4.5. Сторінка реєстрації

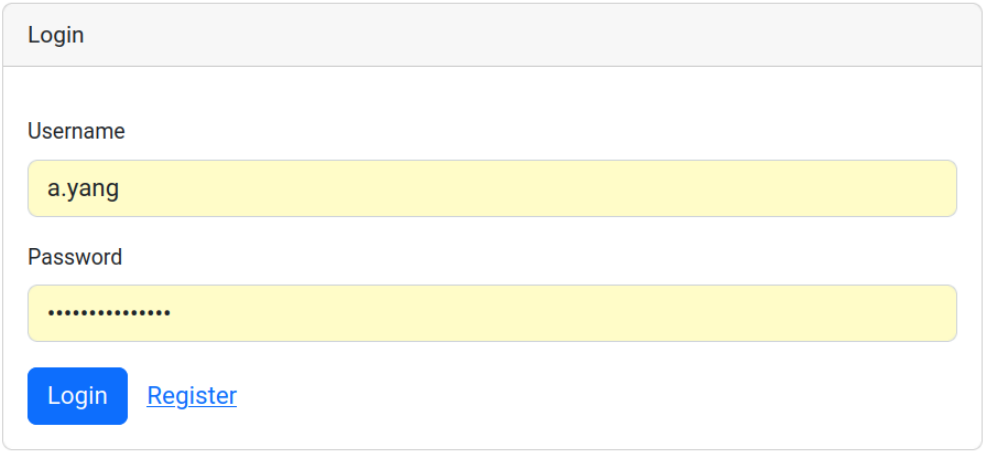


Рис. 4.6. Сторінка входу

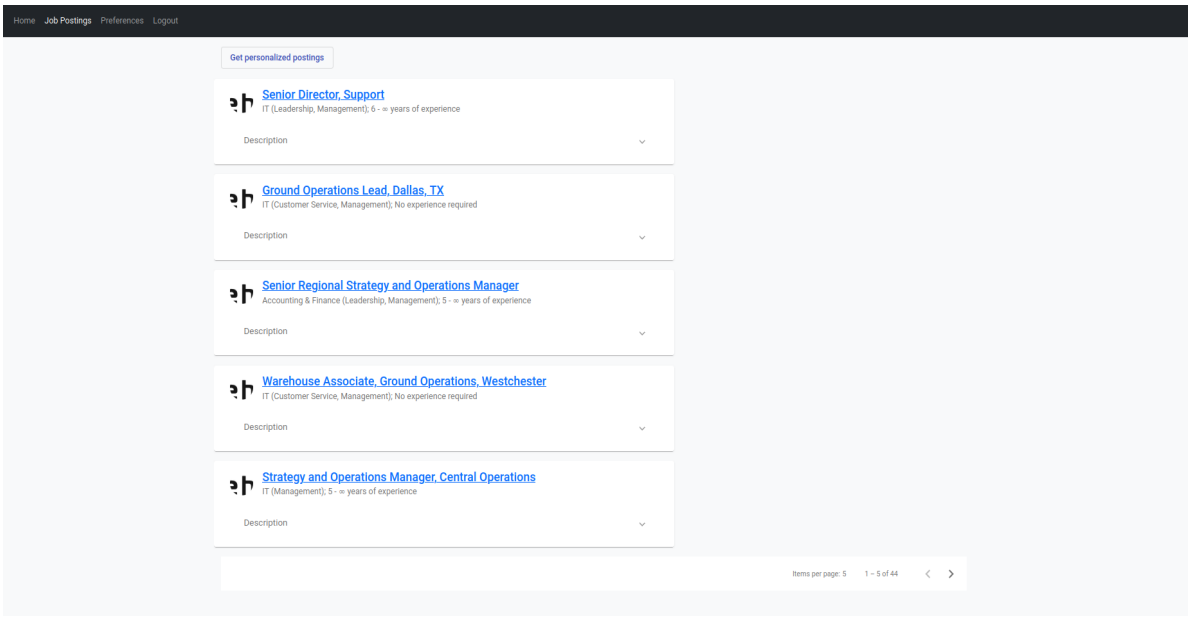


Рис. 4.7. Сторінка перегляду оголошень про роботу

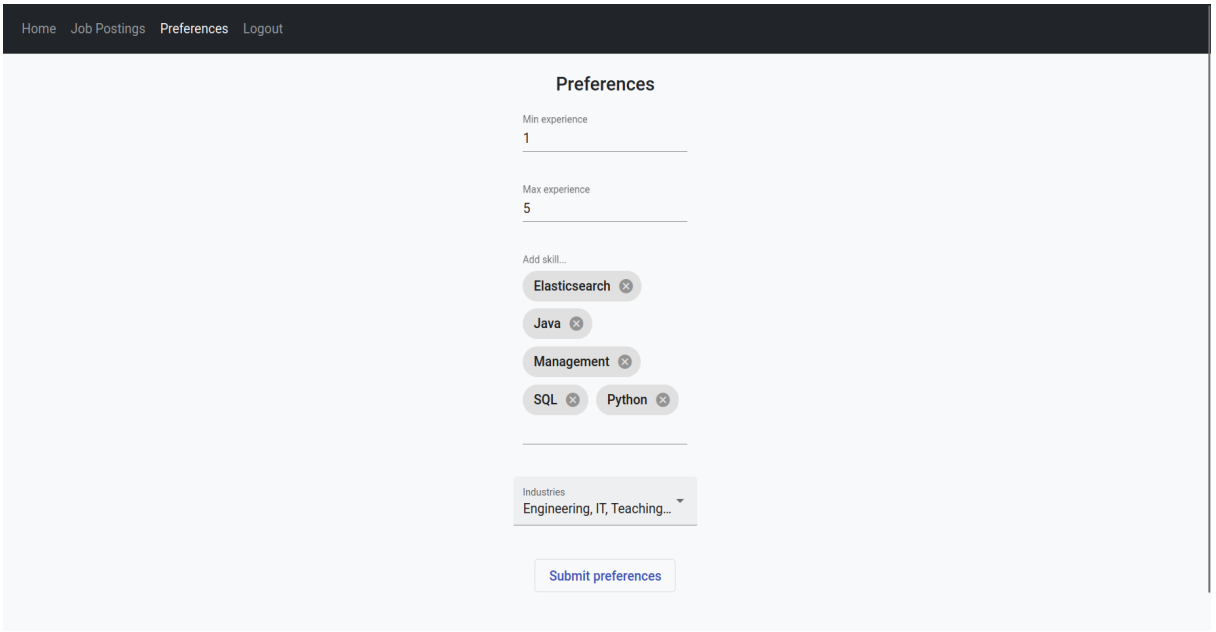


Рис. 4.8. Сторінка уподобань користувача

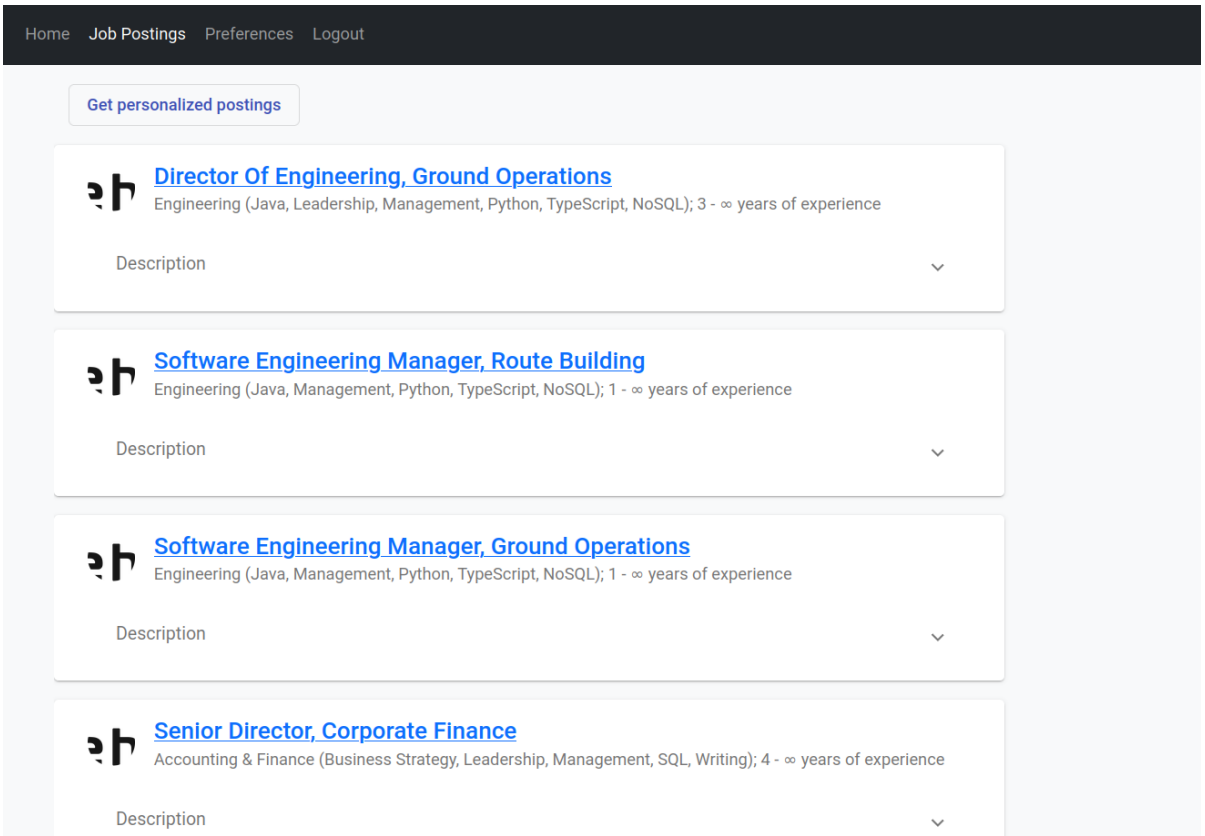


Рис. 4.3. Отримання рекомендованих вакансій

На рис. 4.3 можна побачити, що перші 3 вакансії є більш релевантними до уподобань користувача, ніж 4-а, оскільки мають більший перетин навичок з множиною уподобань користувача. Така поведінка пов’язана з формулою розрахунку релевантності документів до вказаних уподобань у ElasticSearch [34].

4.3. Рекомендації для вдосконалення

Для вдосконалення даного вебдодатку, можна виокремити наступні напрями роботи:

1. Вдосконалення класифікації оголошень, шляхом додавання додаткових параметрів, наприклад, місцезнаходження роботи, розділення навичок на необхідні та бажані, аналіз необхідного освітнього рівня для виконання роботи, необхідний рівень GPA та інші.

2. Впровадження модуля пошуку кандидатів для рекрутерів компанії.
3. Впровадження модуля додавання оголошень.
4. Впровадження email розсилки з рекомендованими вакансіями.

4.4. Висновки до розділу 4

У даному розділі було проведено Smoke тестування розробленого додатку. Було складено діаграми діяльності користувачів за роллю та визначено найбільш перспективні напрямки розвитку програми, а саме вдосконалення системи класифікації оголошень.

ВИСНОВКИ

Метою даного дипломного проєкту було розроблення вебзастосунку для підвищення швидкості та спрощення пошуку роботи.

У першому розділі було проведено аналіз існуючих рішень для виявлення їх сильних та слабких сторін, після чого, на основі даної оцінки, було визначено функціональні та нефункціональні вимоги до розроблюваного вебзастосунку.

У другому розділі було проведено аналіз та обґрунтування вибору засобів програмування. Було визначено інструменти, які найбільше підходять для реалізації вимог до програми.

У третьому розділі було проведено аналіз найпопулярніших архітектур програмного забезпечення та було обґрунтовано обрано архітектуру з точки зору виконання функціональних вимог до програми та розділення обов’язків між компонентами.

У четвертому розділі було протестовано розроблене програмне забезпечення. Для тестування було розроблено діаграми діяльності користувачів за ролями. Крім цього, було запропоновано подальші шляхи вдосконалення розробленого програмного забезпечення, та виокремлено найбільш перспективні

Головною перевагою побудованого вебзастосунку є агрегація оголошень з дощок оголошень різних компаній, та їх автоматичний аналіз з утворенням рекомендацій для користувачів.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. What is GPA and Why It Is so Important [Електронний ресурс]. — Режим доступу: https://www.mastersportal.com/articles/2126/what-is-a-gpa-and-why-is-it-so-important.html
2. What is LinkedIn and how I can use it [Електронний ресурс]. — Режим доступу: https://www.linkedin.com/help/linkedin/answer/a548441/what-is-linkedin-and-how-can-i-use-it-?lang=en
3. About Glassdoor [Електронний ресурс]. — Режим доступу: https://www.glassdoor.com/about/
4. What is Python? Executive Summary [Електронний ресурс]. — Режим доступу: https://www.python.org/doc/essays/blurb/.
5. PEP 206 Python Advanced Library [Електронний ресурс]. — Режим доступу: https://peps.python.org/pep-0206/.
6. Advantages of Interpreted languages [Електронний ресурс]. — Режим доступу: https://stackoverflow.com/questions/1610539/what-are-the-pros-and-cons-of-interpreted-languages.
7. The Pros and Cons of Python programming [Електронний ресурс]. — Режим доступу: https://www.linode.com/docs/guides/pros-and-cons-of-python/.
8. An introduction to JavaScript [Електронний ресурс]. — Режим доступу: https://javascript.info/intro.
9. Javascript vs Python [Електронний ресурс]. — Режим доступу: https://www.tiny.cloud/blog/python-vs-javascript/.
10. What is Java? Definition, Meaning & Features of Java Platforms [Електронний ресурс]. — Режим доступу: https://www.guru99.com/java-platform.html.

11. A tour of C# language [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.
12. Disadvantages of C# [Електронний ресурс]. — Режим доступу: https://www.codeguru.com/csharp/c-sharp-disadvantages/.
13. PostgreSQL: The Worlds Most Advanced Open Source Relational Database [Електронний ресурс]. — Режим доступу: https://www.postgresql.org/.
14. What is MySQL? [Електронний ресурс]. — Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.
15. MongoDB [Електронний ресурс]. — Режим доступу: https://www.tutorialspoint.com/mongodb/mongodb_overview.htm
16. What is Angular? [Електронний ресурс]. — Режим доступу: https://angular.io/guide/what-is-angular
17. Pycharm Features [Електронний ресурс]. — Режим доступу: https://www.jetbrains.com/pycharm/features/
18. Client Server Architecture [Електронний ресурс]. — Режим доступу: https://cio-wiki.org/wiki/Client_Server_Architecture
19. Business Logic [Електронний ресурс]. — Режим доступу: https://www.sciencedirect.com/topics/computer-science/business-logic-layer
20. Presentation layer [Електронний ресурс]. — Режим доступу: https://www.sciencedirect.com/topics/computer-science/presentation-layer
21. MVC Pattern Overview [Електронний ресурс]. — Режим доступу: https://riptutorial.com/mvc-pattern/learn/100000/overview
22. 15 Factors that affect Database Performance [Електронний ресурс]. — Режим доступу: https://ericvanier.com/news/15-factors-that-affect-database-performance/
23. django.contrib.auth [Електронний ресурс]. — Режим доступу: https://docs.djangoproject.com/en/4.2/ref/contrib/auth/

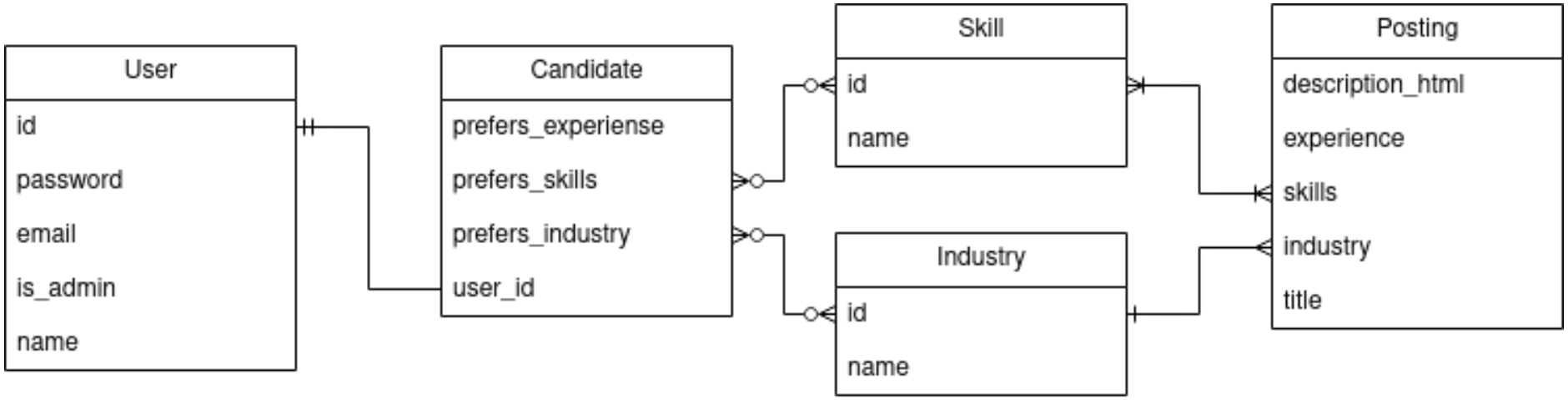
24. Customizing authentication in Django [Електронний ресурс]. — Режим доступу: https://docs.djangoproject.com/en/4.2/topics/auth/customizing/#extending-the-existing-user-model
25. What are microservices? [Електронний ресурс]. — Режим доступу: https://microservices.io/
26. What is serverless? [Електронний ресурс]. — Режим доступу: https://www.redhat.com/en/topics/cloud-native-apps/what-is-serverless
27. Text analytics explained job description data [Електронний ресурс]. — Режим доступу: https://www.kaggle.com/code/chadalee/text-analytics-explained-job-description-data/notebook
28. Multinomial naive bayes [Електронний ресурс]. — Режим доступу: https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes
29. Application testing [Електронний ресурс]. — Режим доступу: https://www.guru99.com/application-testing.html
30. What is smoke testing? [Електронний ресурс]. — Режим доступу: https://www.guru99.com/smoke-testing.html
31. About Indeed [Електронний ресурс]. — Режим доступу: https://www.indeed.com/about
32. Djinni [Електронний ресурс]. — Режим доступу: https://djinni.co/
33. Rule-based methods for text classifying in NLP [Електронний ресурс]. — Режим доступу: https://blog.pangeanic.com/rule-based-methods-for-text-classification-in-nlp
34. Function Score Query [Електронний ресурс]. — Режим доступу: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-function-score-query.html

35. What is Web Scraping and what it is used for? [Електронний ресурс]. — Режим доступу: https://www.parsehub.com/blog/what-is-web-scraping/

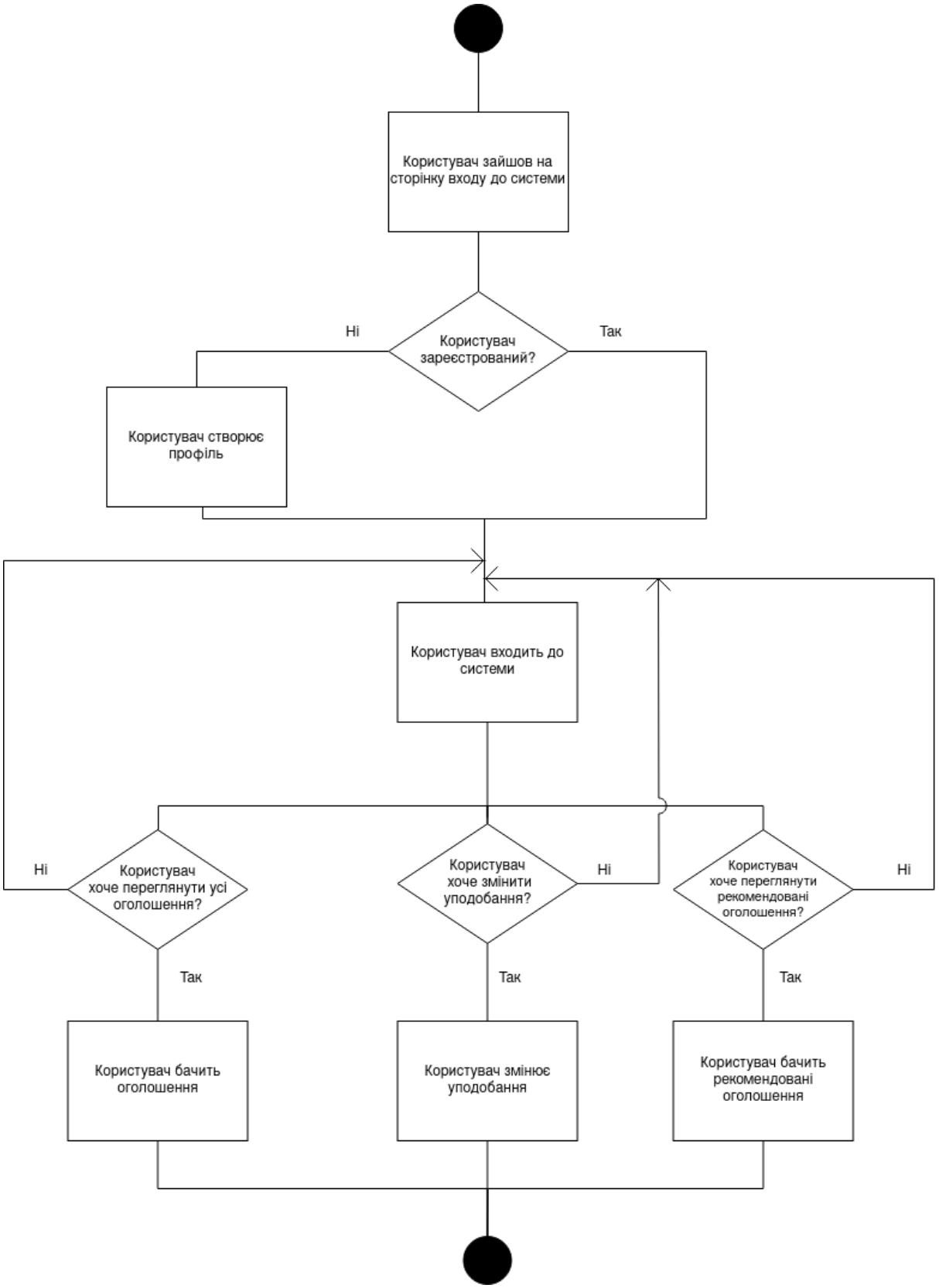
ДОДАТКИ

Додаток 1

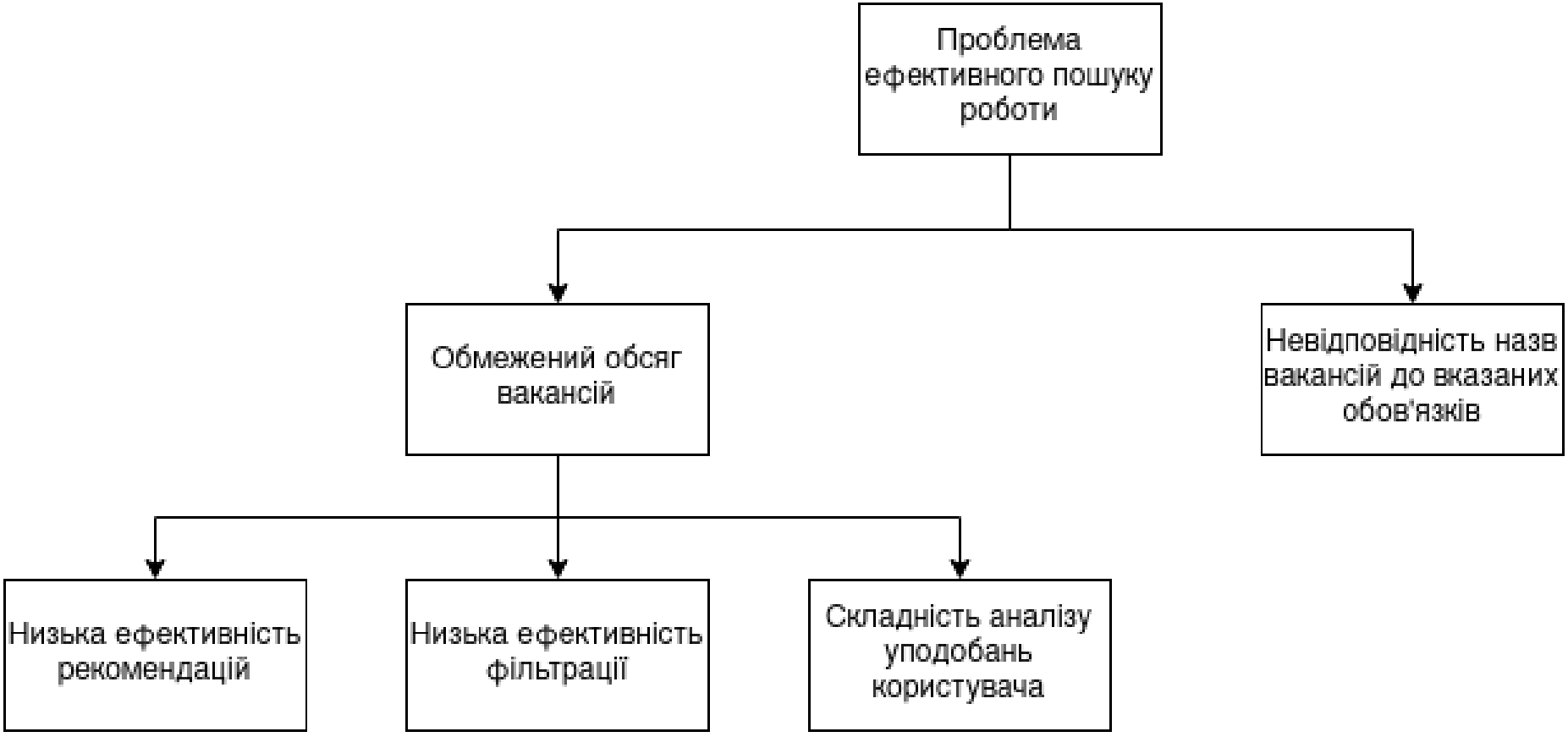
Копії графічних матеріалів



ДП.045430-06-99
Вебзастосунок для спрощення пошуку роботи. Структура бази даних. ERD-діаграма

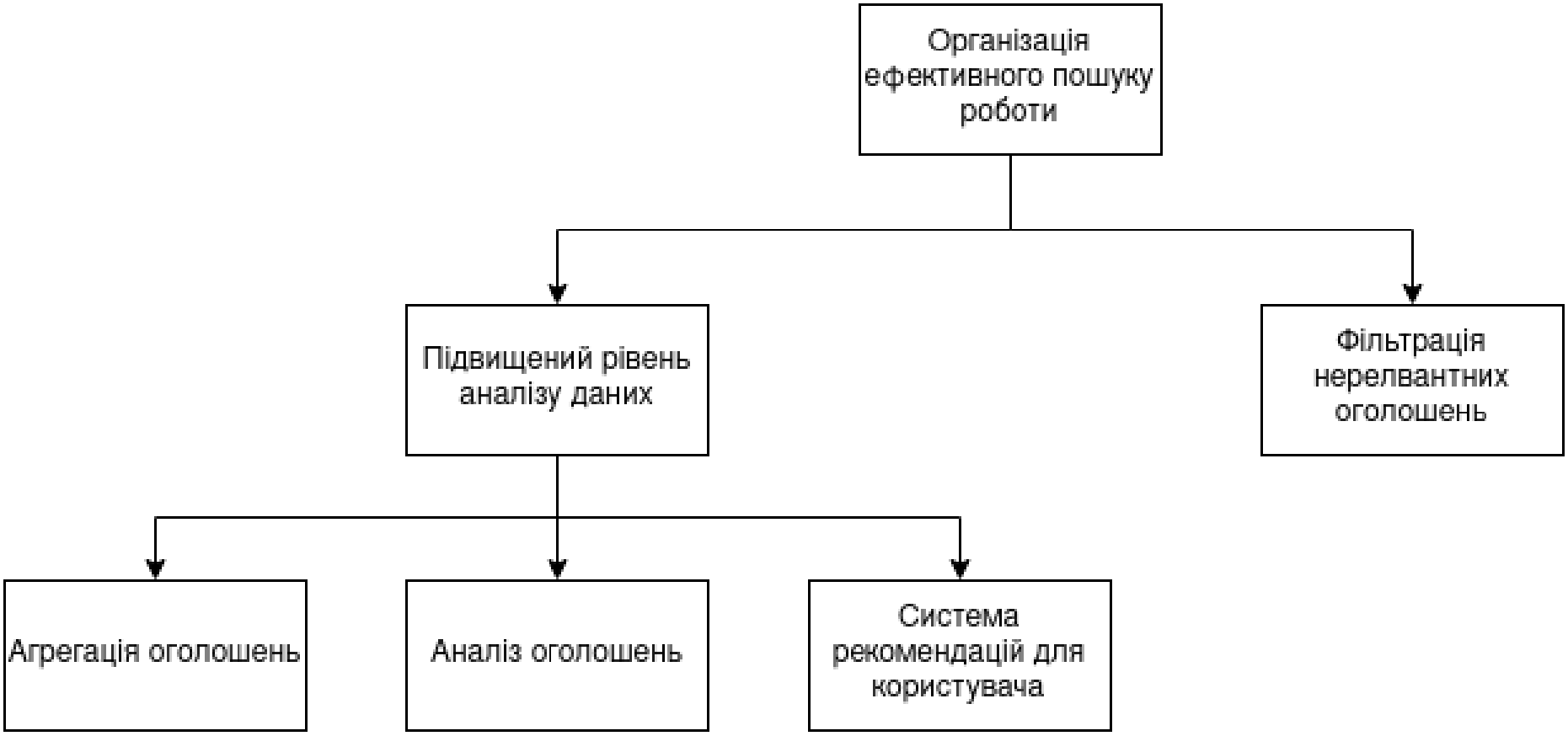


ДП.045440-08-98
Вебзастосунок для спрощення пошуку роботи
Алгоритм дій кандидата. Діаграма діяльності



Дерево проблем проєкту

Долгов О.Ю., група КП-93



Дерево цілей проєкту

Долгов О.Ю., група КП-93

Додаток 2

Лістинг програми

naive_bayes.py

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
import csv
import pickle
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()


SKIPPED_JOBS = {
	"Other/General Jobs",
	"Hospitality & Catering Jobs",
	"Retail Jobs",
	"Charity & Voluntary Jobs",
	"Maintenance Jobs",
	"Domestic help & Cleaning Jobs",
	"Social work Jobs",
	"Part time Jobs",
	"Graduate Jobs",
	"Logistics & Warehouse Jobs",
	"Trade & Construction Jobs",
	"Consultancy Jobs",
}

train = []
train_target = []


test = []
test_target = []

unique_categories = []

with open("Train_rev1.csv", "r") as f:
	reader = csv.DictReader(f)
	print("Length|:::", len(list(reader)))
	for index, row in enumerate(reader):
	if row['Category'] in SKIPPED_JOBS:
		continue

	if index < 50000:
		train.append(row["Title"])
		if row['Category'].removesuffix(" Jobs") not in unique_categories:
		unique_categories.append(row['Category'].removesuffix(" Jobs"))
		train_target.append(row['Category'].removesuffix(" Jobs"))
```

```
        elif 50000 <= index < 1000000:
                test.append(row["Title"])
                test_target.append(row['Category'].removesuffix(" Jobs"))

from collections import Counter
print("Counted:", Counter(train_target))
skills = set()
with open("skills", "r") as f:
        for line in f:
        skills.add(line.lower())

model = make_pipeline(TfidfVectorizer(), MultinomialNB())
model.fit(train, train_target)

print("Fitted", len(test), len(train))
labels = model.predict(test)
print("Labels in result:", Counter(labels))
correct = 0
for index, label in enumerate(labels):
        if test_target[index] == label:
                correct += 1
print("Result:", correct / len(test_target))
```

endpoints.py

```
import sqlalchemy.exc

from models import Company, Posting, Skill, PostingSkills
from models.candidate import Candidate, CandidateIndustries, CandidateSkills
from scrapers import default_dispatcher
import furl
from flask import Blueprint, abort, Response, request
from predictions.bayes import BAYES_MODEL
from dataclass_models.job_posting import JobPosting
from predictions.skills.skills_extractor import default_extractor, Extractor
from connections import db, es
from connections.es import ES_INDEX
from uuid import uuid4
from bool_query import BoolQuery, QueryOption

actions = Blueprint("actions", __name__)

INDUSTRIES = {
        "Engineering": "10b2f5d5-7538-4d7f-81a3-8e4151630c21",
        "PR, Advertising & Marketing": "1bec89e6-391c-4c90-b485-7073db3887ad",
        "Healthcare & Nursing": "3bc1b901-96d5-4440-b973-d83de517dec3",
        "Customer Services": "3eb2e177-30d6-413c-8951-45ded9c35359",
```

```
        "HR & Recruitment": "4a4fcfa9-40fc-4e44-aa36-ae6372e00f7c",
        "Travel": "4e77bee8-4ce8-48d7-b8a4-b72b05977698",
        "Teaching": "5c34d639-edc3-4dd3-a97c-234cc233968b",
        "Energy, Oil & Gas": "6a9554b5-738d-410e-8edf-e27c3f755cf7",
        "Sales": "758ed6e3-ff0f-4c46-87a8-0c44ce7f8d7b",
        "Admin": "9de279fe-c10a-423d-9e36-ba7b821332af",
        "Accounting & Finance": "9df6462e-ea93-4a5c-82ea-cc6122186f78",
        "IT": "afe4661b-99e9-4970-b922-4649f5f888a9",
        "Property": "b703ed7d-f5d6-4a03-9f72-99b2638500b1",
        "Manufacturing": "bf67aef5-dff1-4b6e-9496-f1cac6e99bad",
        "Creative & Design": "de936590-891d-4ee1-8731-09ce6c63f88c",
        "Legal": "e38c0b18-d385-4576-9f63-77666aa41ac3",
        "Scientific & QA": "ee1a93c5-472d-46d5-9b66-4a51c3d73458",
}


@actions.route("/", methods=["POST", "GET"])
def scrape_board():
        requested = request.get_json()
        company_id = requested.get("SCRAPE_COMPANY")
        company = db.session.query(Company).filter(Company.id == company_id).first()
        if company is None:
        abort(404)
        parsed_url = furl.furl(company.board_url)
        board = default_dispatcher.get_for_url(parsed_url)
        if board is None:
        abort(404)
        posting_urls, _ = board.get_postings_list(company.board_url)
        for url in set(posting_urls):
        job_posting = board.get_job_posting(str(url))
        try:
                make_predictions(job_posting,       skill_extractor=default_extractor,
company_id=company.id)
        except Exception as e:
                db.session.rollback()
                print("E:", e)
                print("posting_urls:", url, posting_urls)
        return Response()


def make_predictions(posting: JobPosting, skill_extractor: Extractor, company_id):
        industry_from_title, description = BAYES_MODEL.predict(  # noqa
        [posting.title]
        ), BAYES_MODEL.predict([posting.description_html])

        skill_ids = [
```

```
        skill_id             for             _,             skill_id             in
skill_extractor.extract_skills(job_posting=posting)
        ]
        exp_min, exp_max = skill_extractor.extract_experience(job_posting=posting)

        posting_id = uuid4()
        posting_db = Posting(
        id=posting_id,
        title=posting.title,
        description_html=posting.description_html,
        industry_id=INDUSTRIES[industry_from_title[0]],
        experience_min=exp_min or 0,
        experience_max=exp_max or -1,
        url=posting.url,
        company_id=company_id
        )
        db.session.add(posting_db)
        for skill_id in skill_ids:
        db.session.add(PostingSkills(skill_id=skill_id, posting_id=posting_id))
        db.session.commit()
        db.session.refresh(posting_db)
        es.index(
        index=ES_INDEX, id=str(posting_db.id), body=posting_db.convert_to_elastic()
        )
        return Response()


@actions.route("/create-index", methods=["POST"])
def create_index():
        set_mapping()
        return "Created"


def set_mapping():
        requested = request.get_json()
        force_delete = requested.get("FORCE_DELETE", False)
        mapping = {
        "mappings": {
                "properties": {
                "title": {"type": "text"},
                "description": {"type": "text"},
                "skills": {"type": "keyword"},
                "industry": {"type": "keyword"},
                "experience_min": {"type": "integer"},
                "experience_max": {"type": "integer"},
                }
        }
```

```
        }
        if force_delete:
        es.indices.delete(index=ES_INDEX)
        es.indices.create(
        index=ES_INDEX,
        body=mapping,
        )


@actions.route("/get-recommendations", methods=["POST"])
def get_recommendations():
        requested = request.get_json()
        candidate_id = requested.get("CANDIDATE_ID")

        candidate      =      db.session.query(Candidate).filter(Candidate.id      ==
candidate_id).first()
        if candidate is None:
        abort(404)

        query = BoolQuery()
        for                                    ind                                    in
db.session.query(CandidateIndustries).filter(CandidateIndustries.candidate_id    ==
candidate_id).all():
        query.add_term(QueryOption.should,                              key="industry",
value=str(ind.industry_id))
        for                                   skill                                   in
db.session.query(CandidateSkills).filter(CandidateSkills.candidate_id            ==
candidate_id).all():
        query.add_term(QueryOption.should, key="skills", value=str(skill.skill_id))
        query.add_range(QueryOption.must,                       key="experience_min",
gte=candidate.prefers_min_exp or -1, lte=candidate.prefers_max_exp or 100)
        query.add_range(QueryOption.must,     key="experience_max",     gte=     -1,
lte=candidate.prefers_max_exp or 100)
        query.to_dict()
        res = es.search(index=ES_INDEX, body={"query": query.to_dict()})
        result  =  [{"posting_id":  hit['_id'],  "score":  hit['_score']}  for  hit  in
res["hits"]["hits"]]
        return result
```

user_interactions.py

```
from user_interactions.models import Posting, Industry, Skill
from .serializers import JobPostingsSerializer
from rest_framework import generics
from rest_framework.permissions import IsAuthenticated
from user_interactions.serializers import IndustrySerializer, SkillSerializer
from django.utils.decorators import method_decorator
```

```
from django.views.decorators.cache import cache_page
from rest_framework.response import Response
from backend.settings import RECOMMENDATIONS_URL, RECOMMENDATIONS_API_KEY
import requests


class PostingsView(generics.ListAPIView):
	queryset = Posting.objects.all()
	permission_classes = (IsAuthenticated,)
	serializer_class = JobPostingsSerializer

	# @method_decorator(cache_page(60 * 5))
	def list(self, *args, **kwargs):
	return super(PostingsView, self).list(*args, **kwargs)


class PersonalizedPostingsView(generics.ListAPIView):
	serializer_class = JobPostingsSerializer

	# @method_decorator(cache_page(60 * 5))
	def list(self, *args, **kwargs):
	candidate_id = self.request.user.candidate.id
	result = requests.post(RECOMMENDATIONS_URL, headers={"x-api-key":
RECOMMENDATIONS_API_KEY}, json={"CANDIDATE_ID": candidate_id})
	posting_ids = []
	result_json = result.json()
	for item in result_json:
		posting_ids.append(item["posting_id"])
	postings = Posting.objects.filter(id__in=posting_ids).all()
	postings = sorted(postings, key=lambda posting:
posting_ids.index(str(posting.id)))
	return Response(JobPostingsSerializer(postings, many=True).data)


class IndustriesView(generics.ListAPIView):
	queryset = Industry.objects.all()
	permission_classes = (IsAuthenticated,)
	serializer_class = IndustrySerializer
	pagination_class = None


class SkillsView(generics.ListAPIView):
	queryset = Skill.objects.all()
	permission_classes = (IsAuthenticated,)
	serializer_class = SkillSerializer
	pagination_class = None
```

prefetences_component.ts

```
import {Component, OnInit, OnDestroy, Input, ViewChild, ElementRef} from '@angular/core';
import {COMMA, ENTER} from '@angular/cdk/keycodes';
import { MatCardModule } from "@angular/material/card";
import {ApiService} from '@app/_services';
import {Posting} from "@app/_models/posting";
import {FormBuilder, FormControl, FormGroup, Validators} from "@angular/forms";
import {PreferencesService} from "@app/_services/preference.service";
import {Preferences, PreferencesUpdate} from "@app/_models/preferences";
import {Skill} from "@app/_models/skill";
import {Industry} from "@app/_models/industry";
import {Observable} from "rxjs";
import {map, startWith} from "rxjs/operators";
import {MatChipInputEvent} from "@angular/material/chips";
import {MatAutocomplete, MatAutocompleteSelectedEvent} from "@angular/material/autocomplete";

@Component({ selector: 'app-posting', templateUrl: 'preferences.component.html', styleUrls: ['./preferences.component.css'] })
export class PreferencesComponent implements OnInit, OnDestroy {
  form!: FormGroup;
  skills: Skill[] = [];
  industries: Industry[] = [];
  industriesSelected = new FormControl('');
  myControl = new FormControl('');

  filteredSkills?: Observable<string[]>;

  visible = true;
  selectable = true;
  removable = true;
  addOnBlur = true;
  separatorKeysCodes: number[] = [ENTER, COMMA];
  fruitCtrl = new FormControl();
  filteredFruits: Observable<string[]>;
  selectedSkills: string[] = [];
  allSkills: string[] = [];
  mappedSkills: Record<string, Skill> = {};

  @ViewChild('fruitInput', {static: false}) fruitInput?: ElementRef<HTMLInputElement>;
  @ViewChild('auto', {static: false}) matAutocomplete?: MatAutocomplete;

  constructor(
```

```
    private formBuilder: FormBuilder,
    private preferenceService: PreferencesService,
    private apiService: ApiService
  ) {
    this.filteredFruits = this.fruitCtrl.valueChanges.pipe(
      startWith(null),
              map((fruit: string | null) => fruit ? this._filter(fruit) :
this.allSkills.slice()));
  }

  add(event: MatChipInputEvent): void {
    // Add fruit only when MatAutocomplete is not open
    // To make sure this does not conflict with OptionSelected Event
    // @ts-ignore
    if (!this.matAutocomplete.isOpen) {
      const input = event.input;
      const value = event.value;

      // Add our fruit
      if ((value || '').trim()) {
        this.selectedSkills.push(value.trim());
      }

      // Reset the input value
      if (input) {
        input.value = '';
      }

      this.fruitCtrl.setValue(null);
    }
  }

  remove(fruit: string): void {
    const index = this.selectedSkills.indexOf(fruit);

    if (index >= 0) {
      this.selectedSkills.splice(index, 1);
    }
  }

  selected(event: MatAutocompleteSelectedEvent): void {
    this.selectedSkills.push(event.option.viewValue);
    // @ts-ignore
    this.fruitInput.nativeElement.value = '';
    this.fruitCtrl.setValue(null);
  }
```

```
  private _filter(value: string): string[] {
    const filterValue = value.toLowerCase();

     return this.allSkills.filter(fruit => fruit.toLowerCase().indexOf(filterValue)
=== 0);
  }


  onSubmit(): void {
      let value = this.form.value;
      let prefs = new PreferencesUpdate();
      prefs.prefers_min_exp = value.prefers_min_exp ? value.prefers_min_exp : null;
      prefs.prefers_max_exp = value.prefers_max_exp ? value.prefers_max_exp : null;
        prefs.prefers_skills = this.selectedSkills.map((skillName): string => { //
@ts-ignore
        return this.mappedSkills[skillName].id})
      // @ts-ignore
         prefs.prefers_industries = this.industriesSelected.value === null ? [] :
this.industriesSelected.value;
         this.preferenceService.updateOrCreatePreferences(prefs).subscribe((res) =>
{})
  }

  ngOnDestroy(): void {
    return
  }
  ngOnInit(): void {
    this.form = this.formBuilder.group({
      prefers_min_exp: [''],
      prefers_max_exp: [''],
      prefers_skills: [''],
      prefers_industries: [''],
    });
    this.apiService.getIndustries().subscribe((res) => {
        this.industries = res;
      }
    )
    this.apiService.getSkills().subscribe((res) => {
        this.skills = res;
        for (let skill of this.skills) {
          this.mappedSkills[skill.name ? skill.name : ''] = skill
        }

        this.allSkills = res.map((skill) => skill.name ? skill.name : '')
      }
    )
    // @ts-ignore
```

```
    this.filteredSkills = this.myControl.valueChanges.pipe(
        startWith(null),
        map((fruit) => fruit ? this._filter(fruit) : this.skills?.slice().map(
          (item) => item.name
        )));

    this.preferenceService.getPreferences().subscribe((preferences) => {
      this.form.controls['prefers_min_exp'].setValue(preferences.prefers_min_exp);
      this.form.controls['prefers_max_exp'].setValue(preferences.prefers_max_exp);
      // @ts-ignore

this.industriesSelected.setValue(preferences.prefers_industries.map((industry)    =>
industry.id));
      // @ts-ignore
      this.selectedSkills = preferences.prefers_skills?.map((skill) => skill.name)
    })
  }
}
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

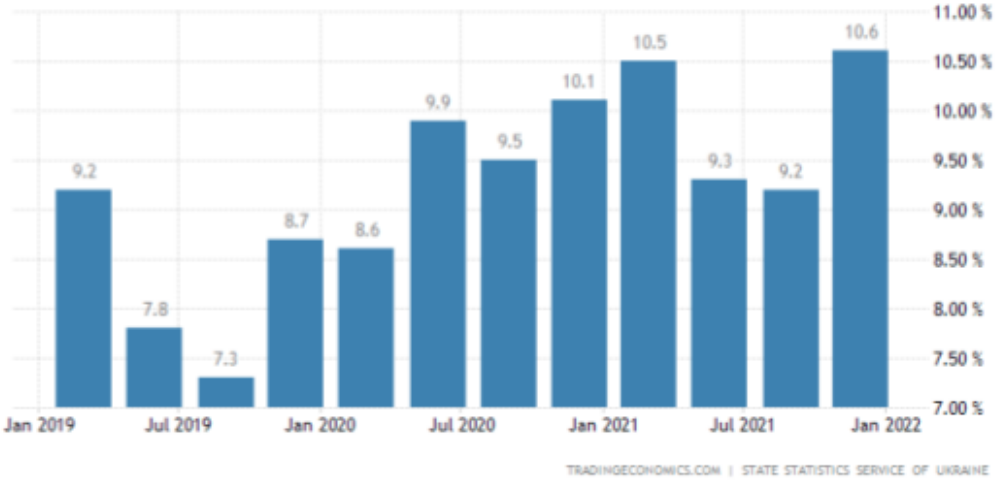
Вебзастосунок для спрощення пошуку роботи

Виконав: Долгов Олексій Юрійович

Доцент кафедри ПЗКС, к.т.н., доцент, Олещенко Любов Михайлівна

Київ – 2023

АКТУАЛЬНІСТЬ



Рівень безробіття в Україні

Рівень безробіття в світі

АНАЛОГИ

glassdoor

djinni

indeed®

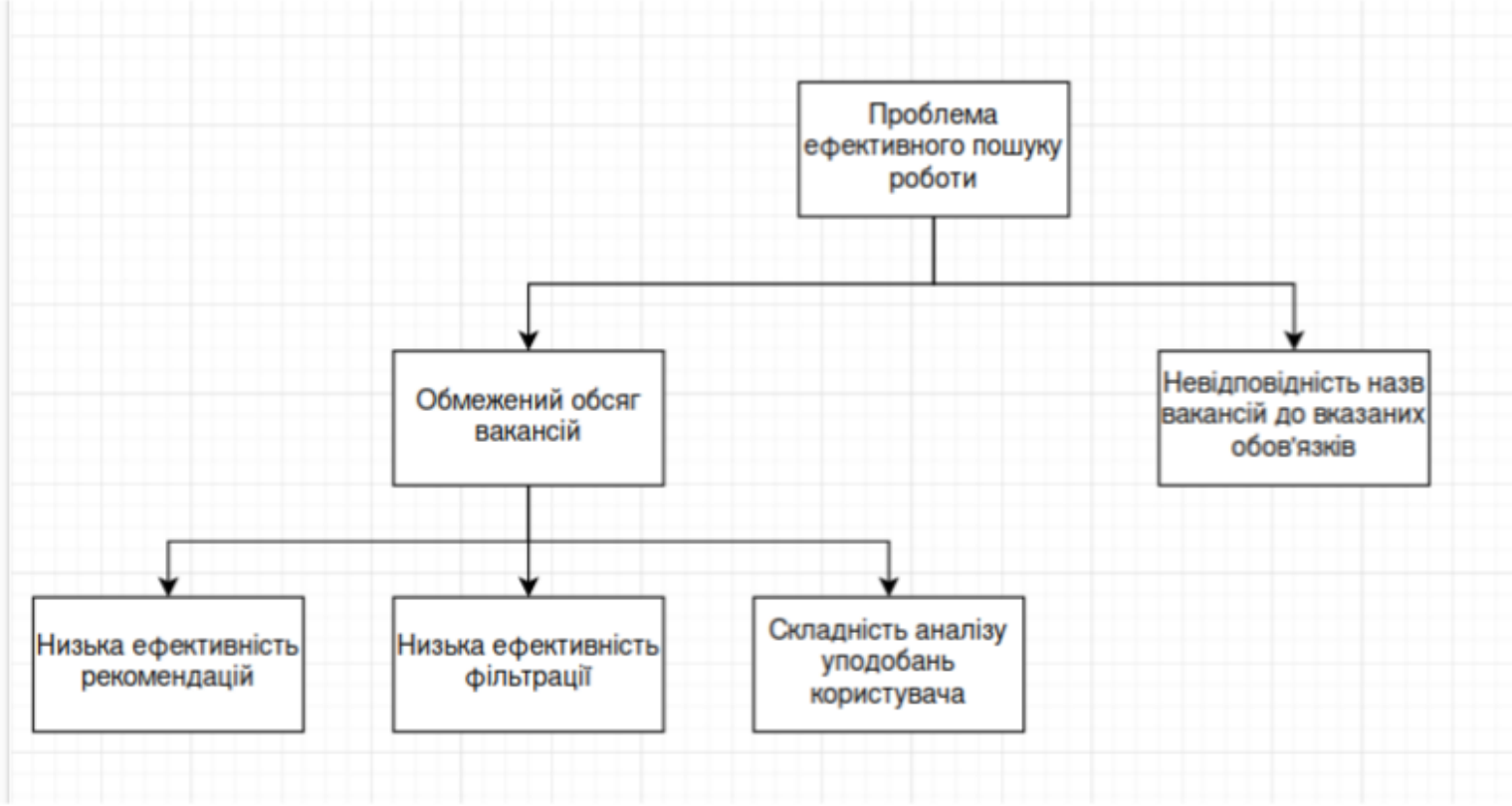
МЕТА ПРОЕКТУ

Створити програмне забезпечення для спрощення пошуку роботи з використанням агрегації та завдяки автоматичній класифікації оголошень.

Завдання

1. Дослідити та проаналізувати потреби людей, що шукають роботу.
2. Проаналізувати існуючі програмні рішення.
3. Визначити вимоги до розроблюваного ПЗ.
4. Розробити програмне забезпечення для спрощення пошуку роботи.
5. Провести тестування отриманого ПЗ та проаналізувати результати.
6. Знайти шляхи подальшого розвитку проєкту.

ДЕРЕВО ПРОБЛЕМ

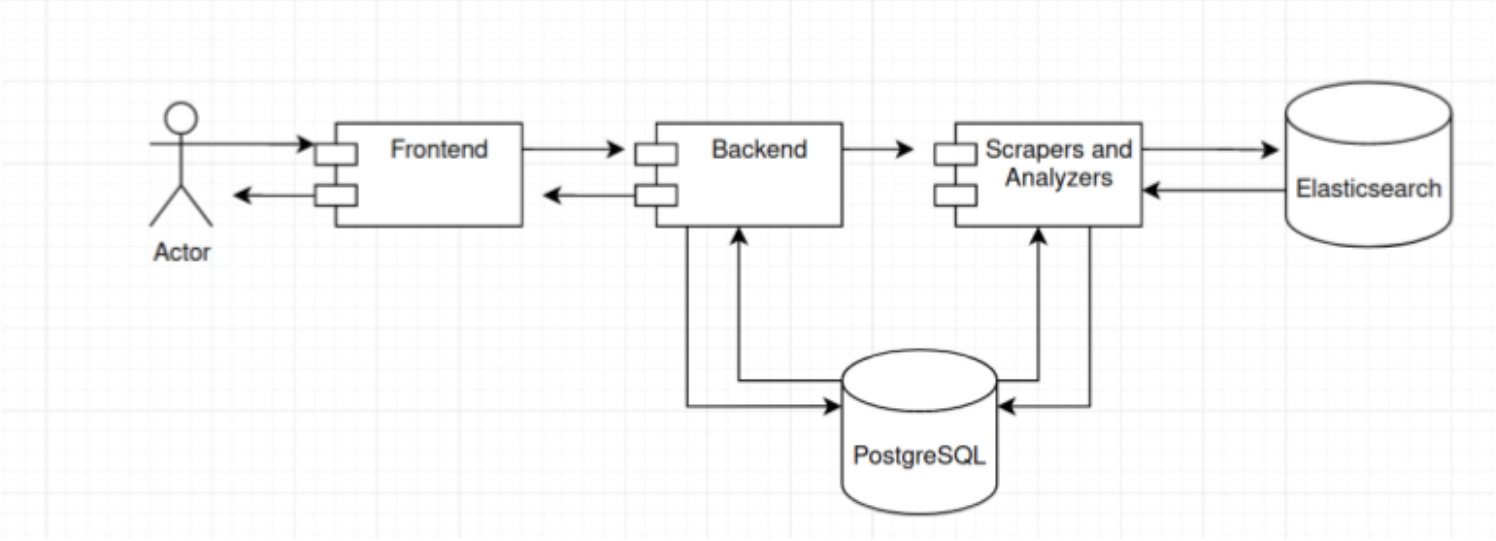


ВИБІР ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ

elasticsearch

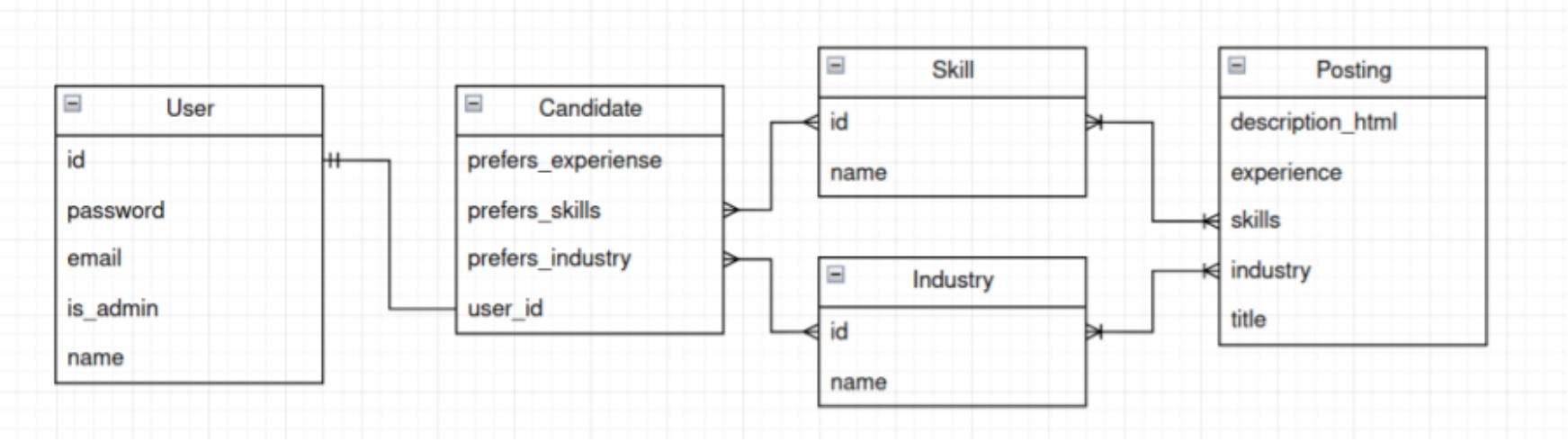
django

АРХІТЕКТУРА ПЗ



Спрощена архітектура додатку

СТРУКТУРА Бази Даних



Спрощена структура бази даних

Класифікатор

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Теорема Байєса

$$\mathrm{tf}(t,d) = \frac{n_t}{\sum_k n_k},$$

Міра TF

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{\, d_i \in D \mid t \in d_i \,\}|}$$

Міра IDF

Датасет

Train_rev1.csv (440.63 MB)

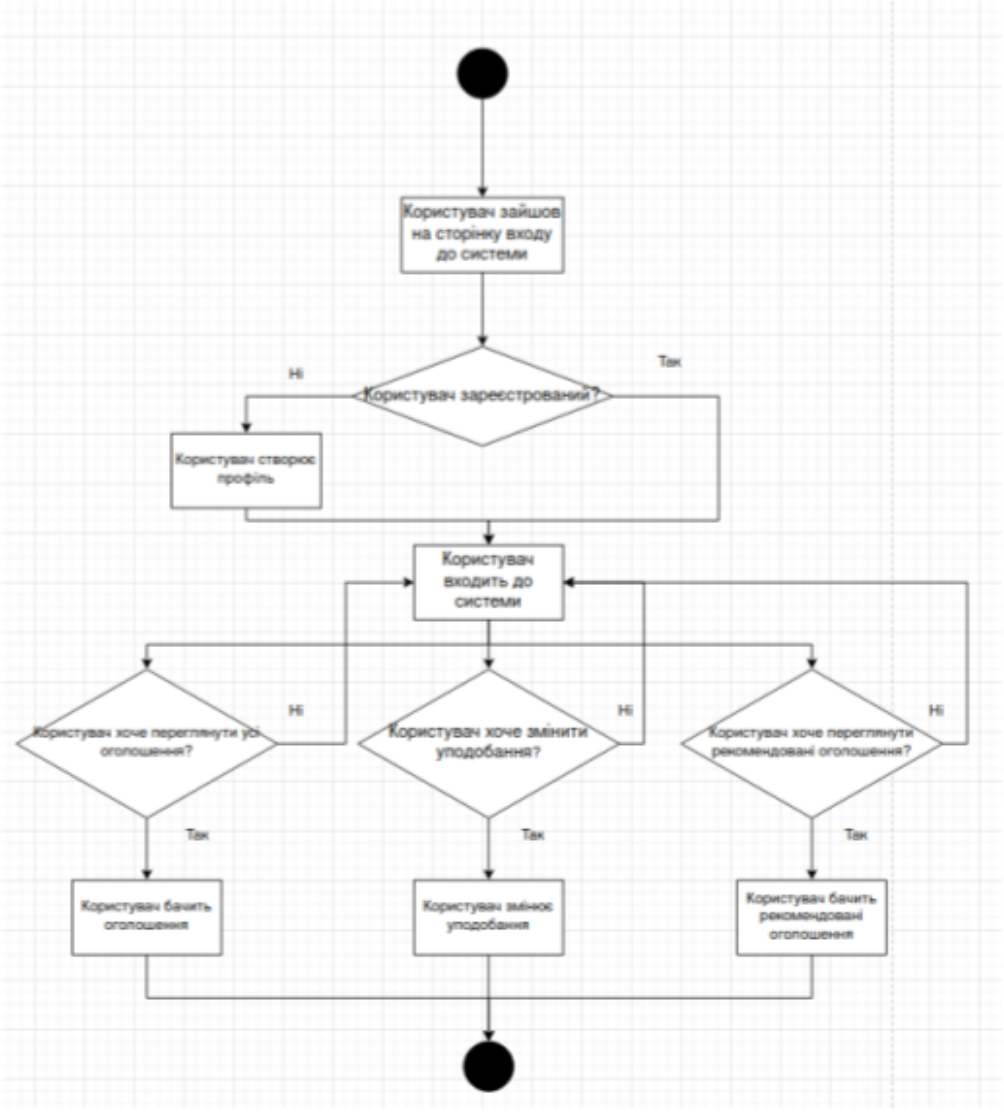
Detail | Compact | Column

10 of 12 columns

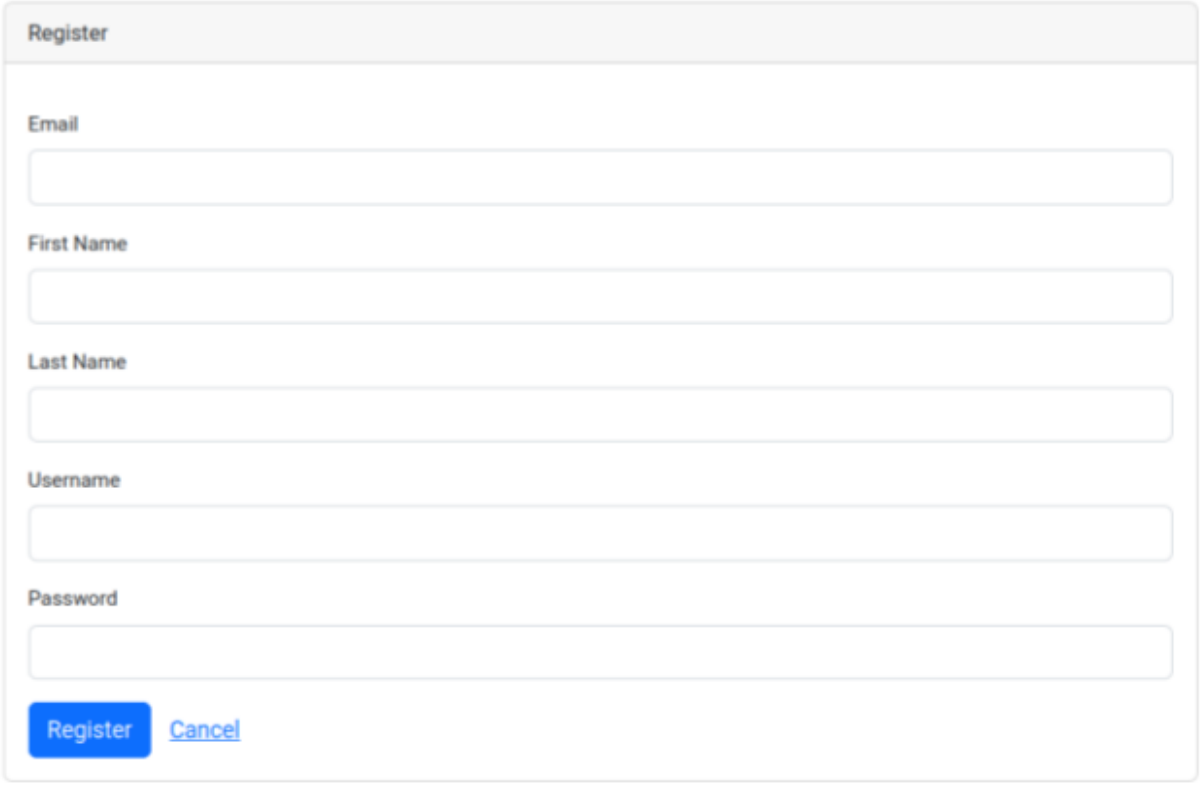
Title	FullDescri...	LocationR...	LocationN...	ContractT...	ContractTi...	Company	Category	SalaryRaw
Engineering Systems Analyst	Engineering Systems Analyst Dorking Surrey Salary ****K Our client is located in Dorking, Surrey and...	Dorking, Surrey, Surrey	Dorking		permanent	Gregory Martin International	Engineering Jobs	20000 - 30000/annum 20-30K
Stress Engineer Glasgow	Stress Engineer Glasgow Salary **** to **** We re currently looking for talented engineers to join o...	Glasgow, Scotland, Scotland	Glasgow		permanent	Gregory Martin International	Engineering Jobs	25000 - 35000/annum 25-35K
Modelling and simulation analyst	Mathematical Modeller / Simulation Analyst / Operational Analyst Basingstoke, Hampshire Up to ****K ...	Hampshire, South East, South East	Hampshire		permanent	Gregory Martin International	Engineering Jobs	20000 - 40000/annum 20-40K
Engineering Systems Analyst / Mathematical Modeller	Engineering Systems Analyst / Mathematical Modeller. Our client is a highly successful and respected...	Surrey, South East, South East	Surrey		permanent	Gregory Martin International	Engineering Jobs	25000 - 30000/annum 25K-30K negotiable

adzuna

ДІАГРАМА ДІЯЛЬНОСТІ. ВЗАЄМОДІЯ КОРИСТУВАЧА З ПРОГРАМОЮ



Сторінка реєстрації в системі



Сторінка входу в систему

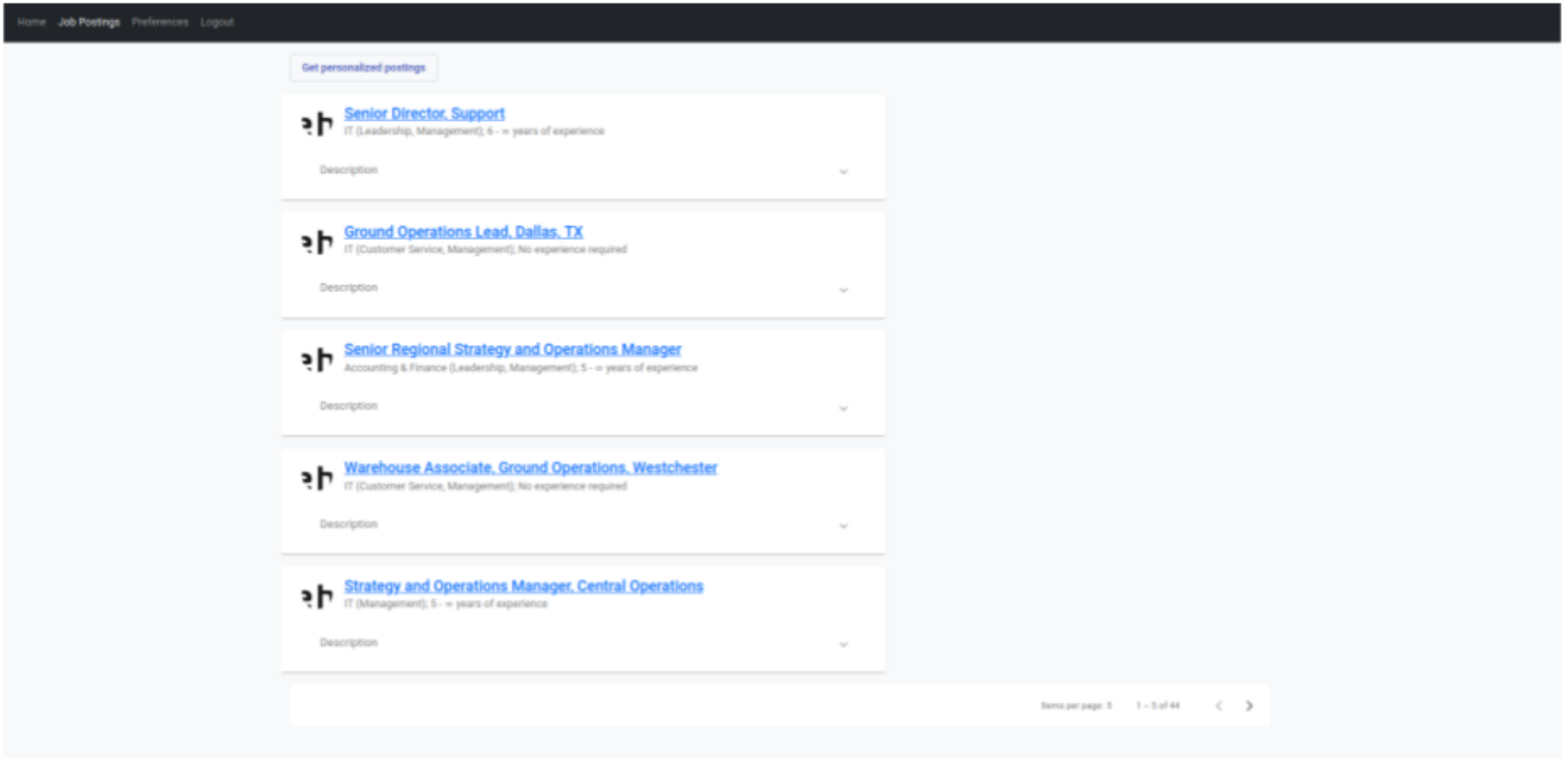
Login

Username

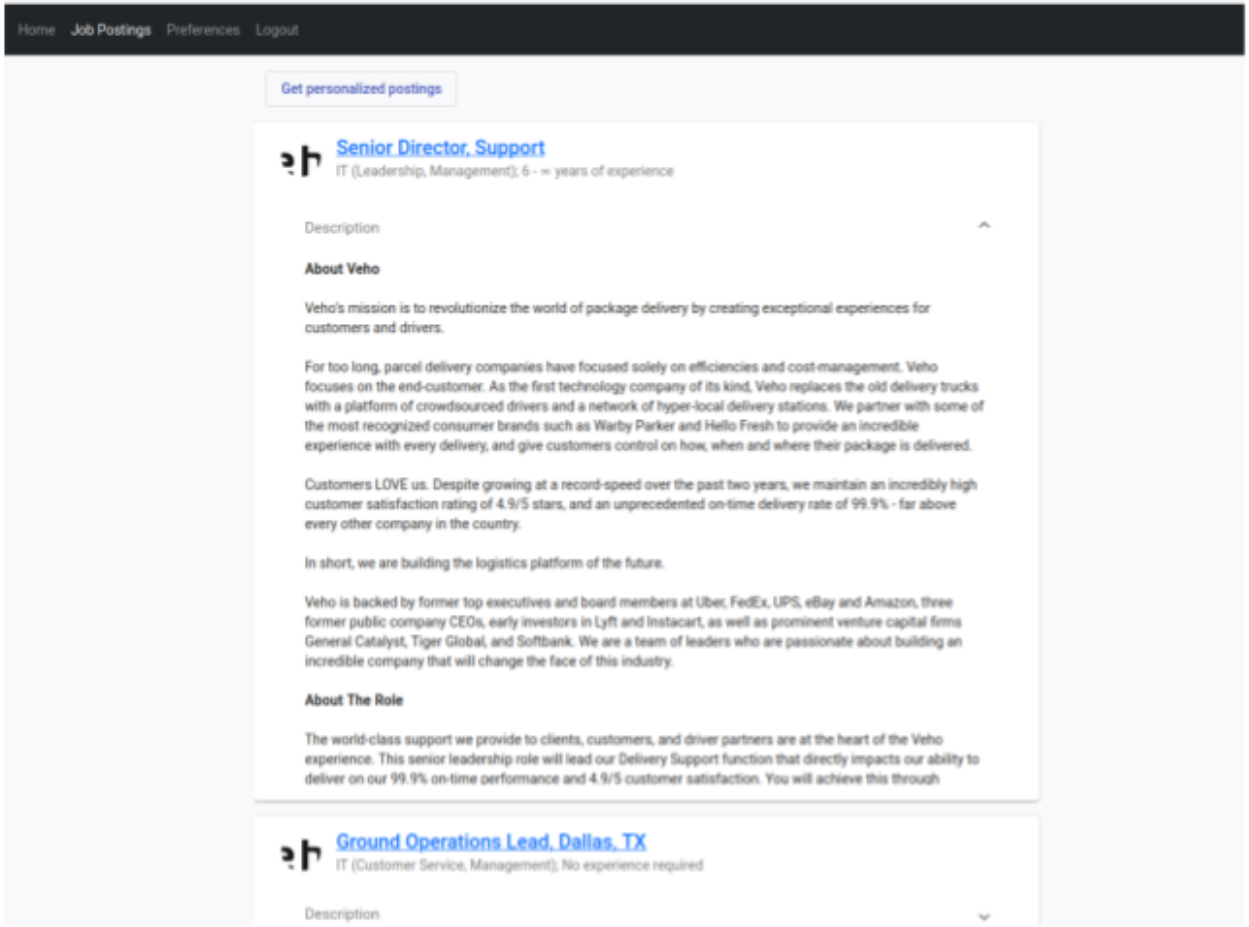
Password

Login Register

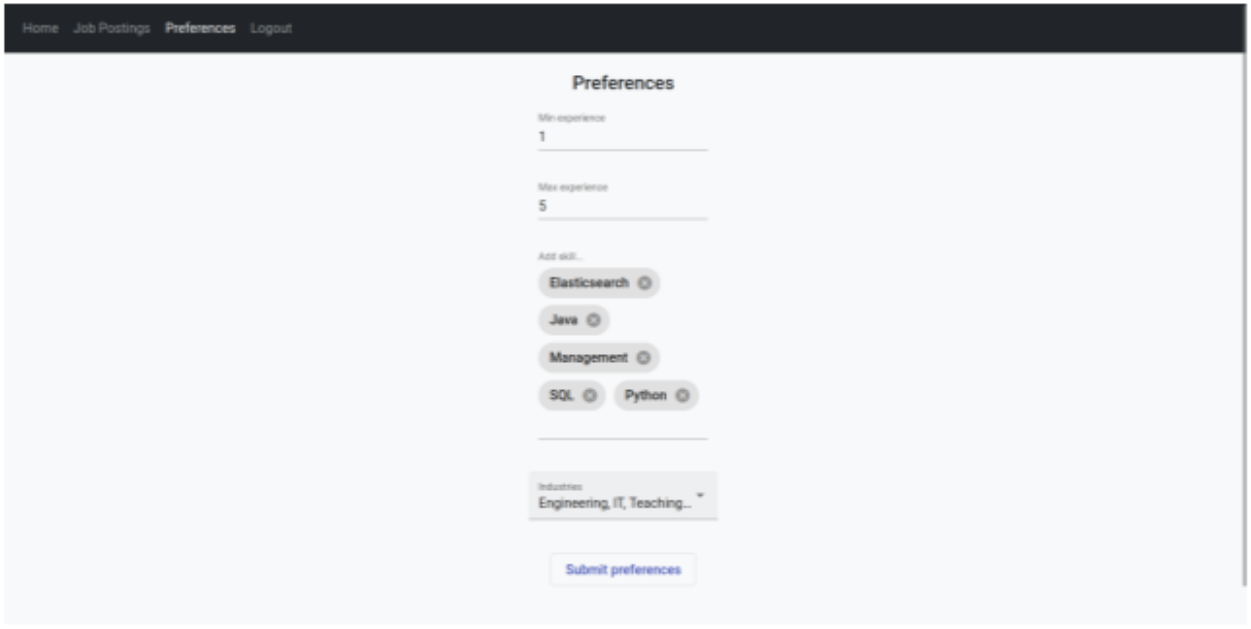
Сторінка перегляду вакансій



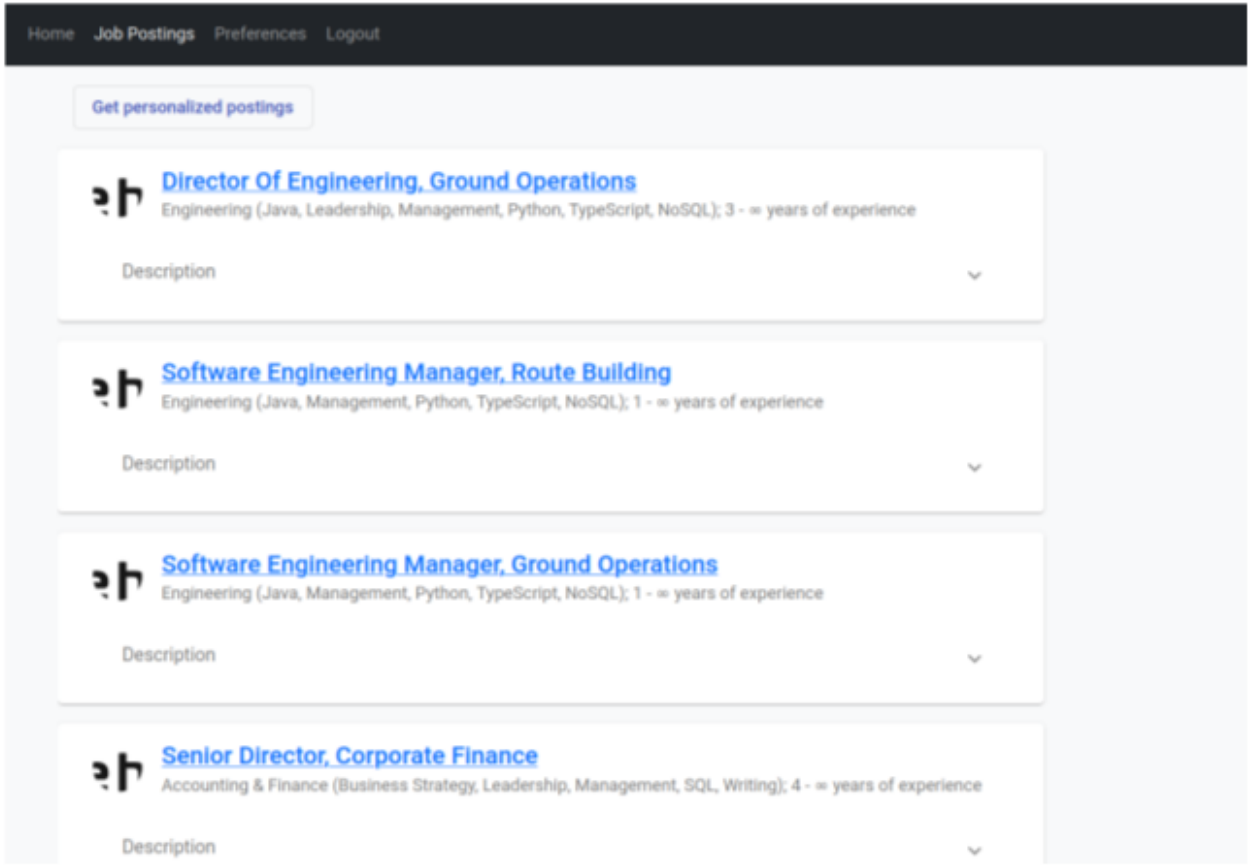
Сторінка перегляду вакансій



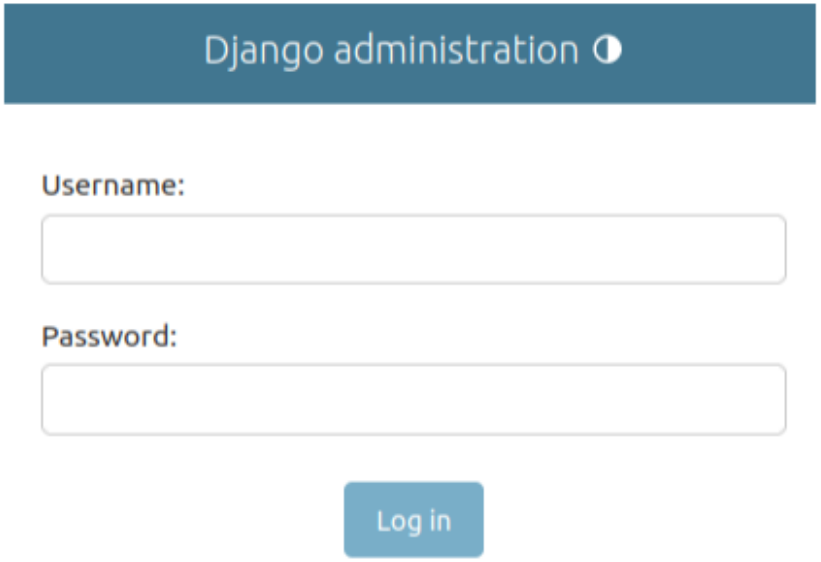
Сторінка уподобань користувача



Отримання персоналізованих вакансій



Сторінка входу до адмін панелі



Сторінка адмін панелі

Тестування

Було проведено димове тестування розробленого додатку. Програма успішно пройшла тестування. В результаті тестування не виявлено критичних помилок.

ШЛЯХИ ПОДАЛЬШОГО РОЗВИТКУ ПРОЄКТУ

- Вдосконалення класифікації оголошень, шляхом додавання додаткових параметрів, наприклад, місцезнаходження роботи, розділення навичок на необхідні та бажані, аналіз необхідного освітнього рівня для виконання роботи, необхідний рівень GPA та інші
- Впровадження модуля пошуку кандидатів для рекрутерів компанії
- Впровадження email розсилки з рекомендованими вакансіями

ВИСНОВКИ

В процесі роботи над даним проєктом було зроблено:

1. Було досліджено проблеми користувачів, що шукають роботу
2. Знайдено та проаналізовано існуючі програмні рішення
3. Сформовано і визначено вимоги до розроблюваного ПЗ.
4. Розроблено програмне забезпечення, що дозволяє пришвидшити пошук роботи
5. Проведено тестування отриманого ПЗ та проаналізовано результати, відповідно до яких ПЗ відповідає вимогам.
6. Знайдено шляхи подальшого розвитку проєкту.

Унікальність

UNICHECK

User name:
Олещенко Любов Михайлівна

Check ID:
1015433591

Check date:
05.06.2023 14:42:47 EEST

Check type:
Doc vs Internet + Library

Report date:
05.06.2023 17:36:49 EEST

User ID:
84299

File name: **Долгов_КП_93**

Page count: **35** Word count: **6753** Character count: **51211** File size: **37.95 KB** File ID: **1015083095**

6.92% Matches

Highest match: **2.21%** with Library source (File ID: **1008301142**)

1.44% Internet sources	133	Page 37
6.92% Library sources	228	Page 37

0% Quotes

No quotes found

Exclusion of references is off

0% Exclusions

No exclusions

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ВЕБЗАСТОСУНОК ДЛЯ СПРОЩЕННЯ ПОШУКУ РОБОТИ
Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Любов ОЛЕЩЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____Олексій ДОЛГОВ

2022

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Вебзастосунок для спрощення пошуку роботи з використанням таких технологій, як AWS Lambda, ElasticSearch, Naive Bayes Text Classification, web scraping, Django.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Забезпечення належного рівня безпеки даних.
2. Коректність взаємодії з базою даних.
3. Тестування верстки.
4. Тестування сумісності з різними браузерами.
5. Зручність роботи з вебдодатком.
6. Тестування зручності використання.
7. Відповідність вимогам технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1. Функціональне тестування, зокрема на рівні Critical path test (базове тестування).
2. Димове тестування (Smoke testing).
3. Тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.

Працездатність web-ресурсу перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати.
2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Статичного тестування коду.
4. Тестування вебдодатку в різних веббраузерах.
5. Тестування при різному навантаженні.
6. Тестування стабільності роботи при різних умовах.
7. Тестування зручності використання.
8. Тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ВЕБЗАСТОСУНОК ДЛЯ СПРОЩЕННЯ ПОШУКУ РОБОТИ
Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Любов ОЛЕЩЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ДОЛГОВ

2023

ЗМІСТ

1. Опис структури програмного забезпечення

Вебзастосунок для спрощення пошуку роботу складається з набору модулів. У структурі розробленого програмного забезпечення можна чітко виділити такі модулі:

- модуль авторизації, реєстрації;
- модуль зміни уподобань;
- модуль отримання персоналізованих рекомендацій;
- модуль скрапінгу і аналізу оголошень;
- модуль адміністрування вебсайту.

2. Процедура авторизації та реєстрації користувача

При переході на сторінку вебзастосунку, неавторизований користувач опиняється на сторінці авторизації (рис. 1).

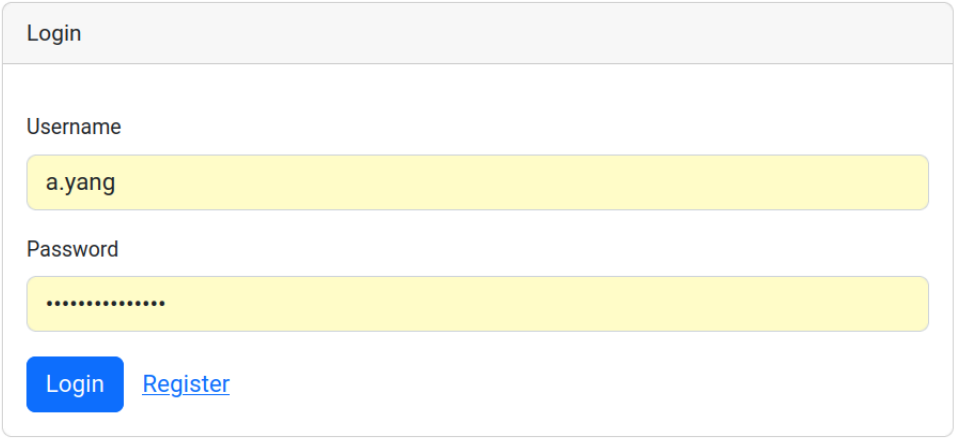


Рис. 1. Сторінка авторизації

На цій сторінці користувач повинен ввести свій логін та пароль. Якщо користувач ввів пароль вірно, його буде переадресовано на сторінку перегляду вакансій (рис. 2).

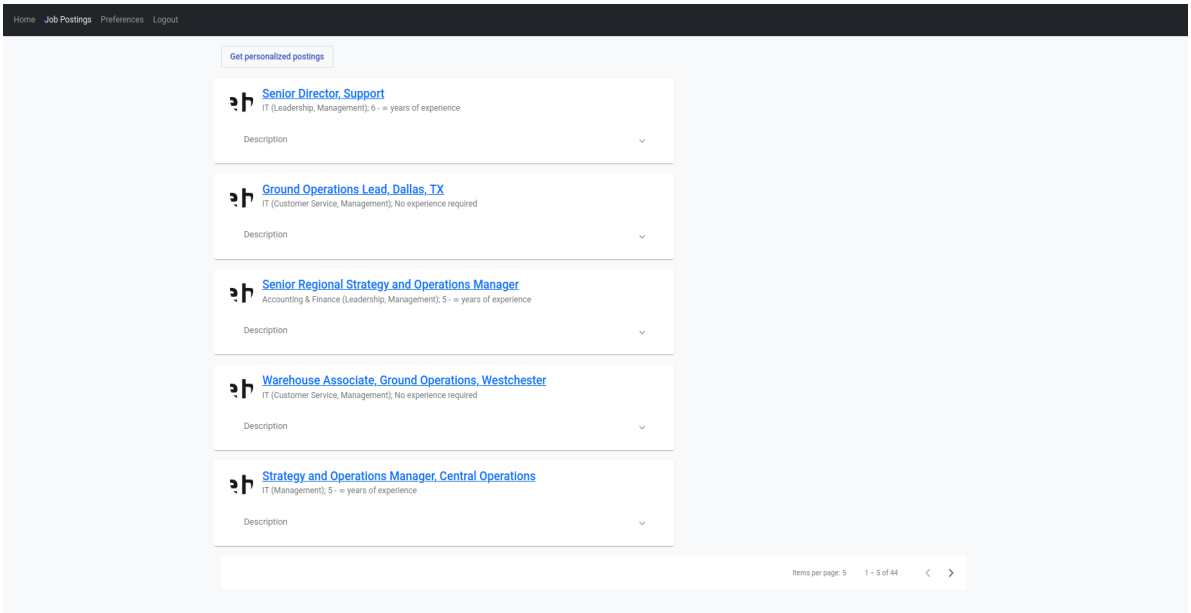


Рис. 2. Сторінка перегляду вакансій

Якщо ж користувач не зареєстрований, то він може перейти на сторінку реєстрації (рис. 3) та заповнити наступні поля:

- ім’я;
- фамілія;
- електронна адреса;
- юзернейм;
- пароль.

Після чого нажати “Register”. Якщо всі поля коректно заповнені, користувач отримує повідомлення про успішну реєстрацію.

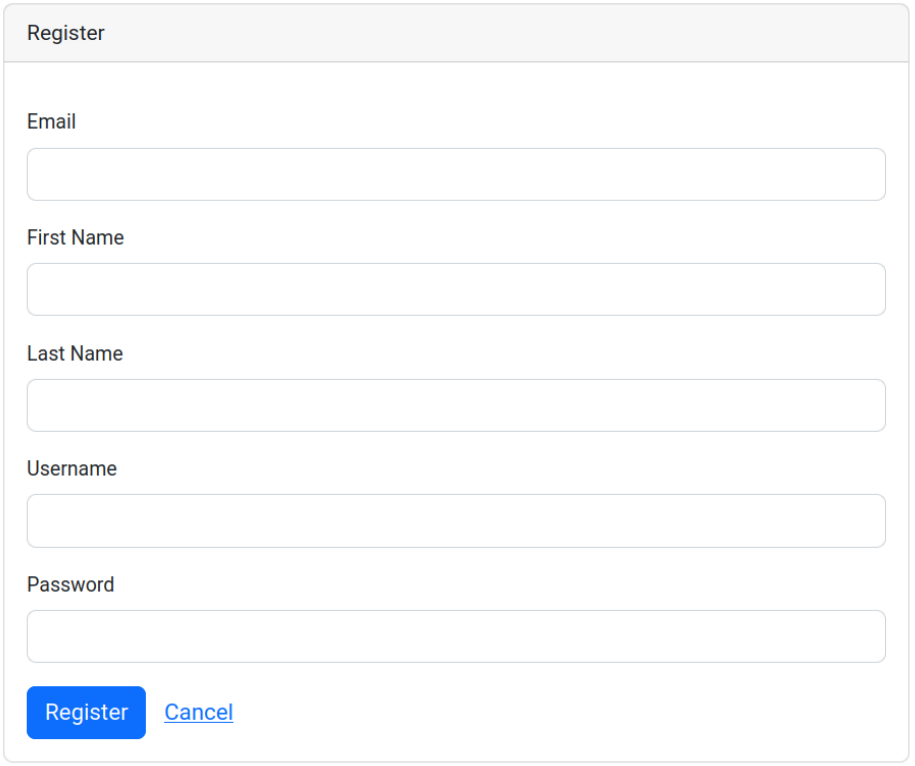


Рис. 3. Реєстрація користувача

3. Процедура зміни уподобань

Користувач може перейти на сторінку уподобань, та заповнити свої уподобання для отримання персоналізованих рекомендацій (рис. 4). Можна заповнити наступні уподобання:

- мінімальний досвід роботи;
- максимальний досвід роботи;
- навички;
- індустрії, в яких кандидат шукає вакансії.

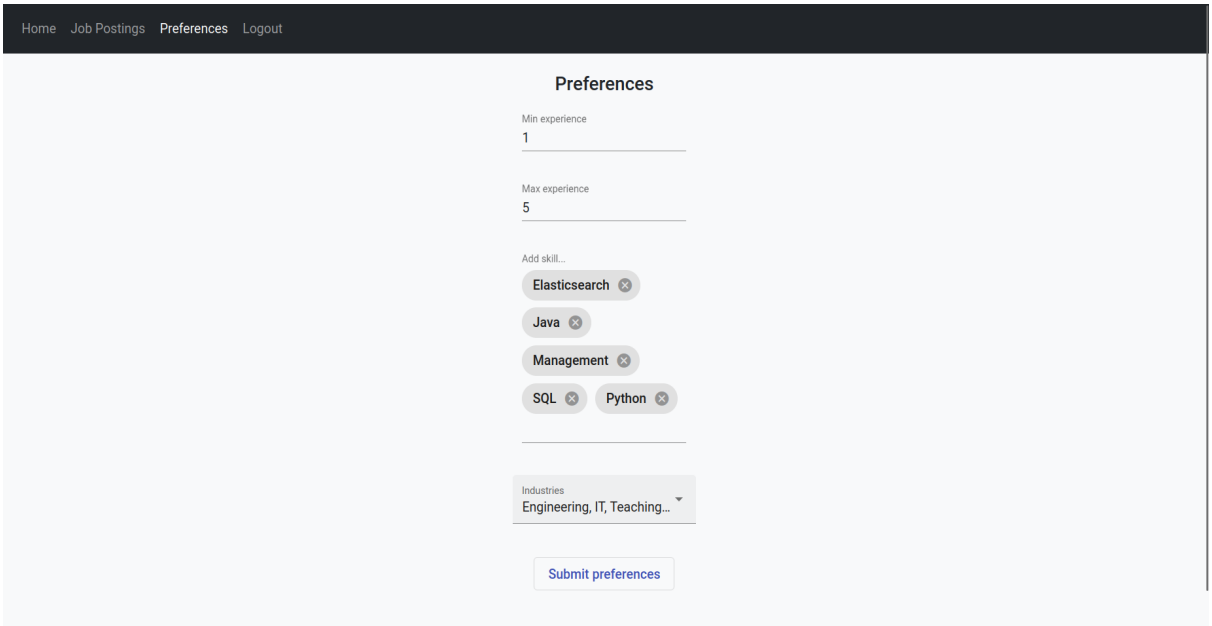


Рис. 4. Сторінка уподобань користувача

4. Процедура підписки на подію

Після того, як користувач заповнив свої уподобання, на сторінці перегляду вакансій він может переглянути персоналізовані відповідно до його уподобань вакансії (рис. 5).

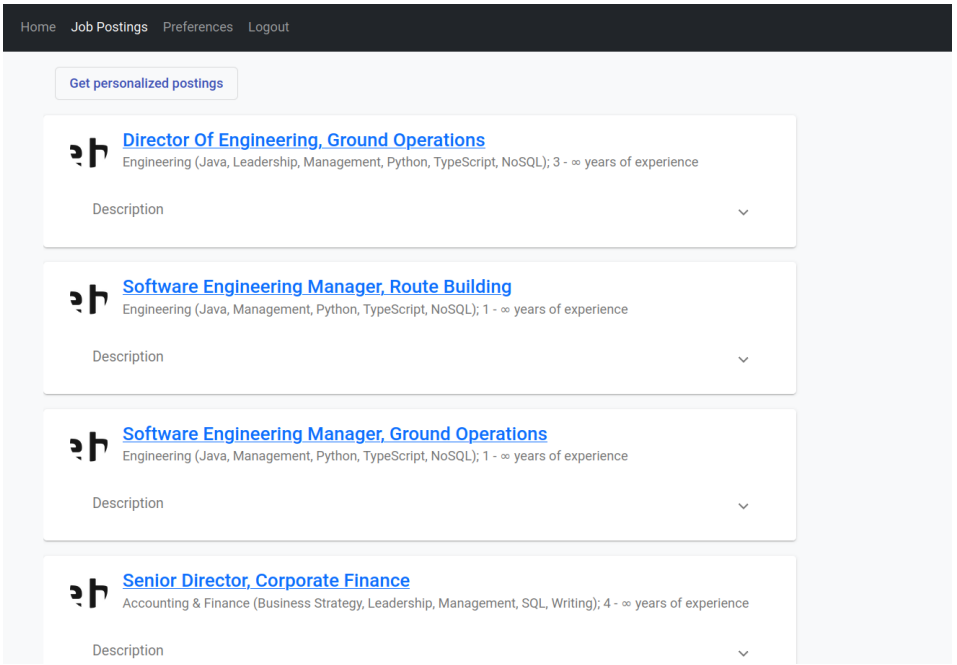


Рис. 5. Отримані персоналізовані вакансії після виконання дії “Get personalized postings”

5. Процедура виконання дій адміністратора

Для виконання дій адміністратора необхідно отримати або згенерувати акаунт адміністратора шляхом стандартних дій у фреймворуку Django. Після того, як було отримані дані для входу в акаунт, адміністратор може увійти в адмін панель через сторінку входу (рис. 6).

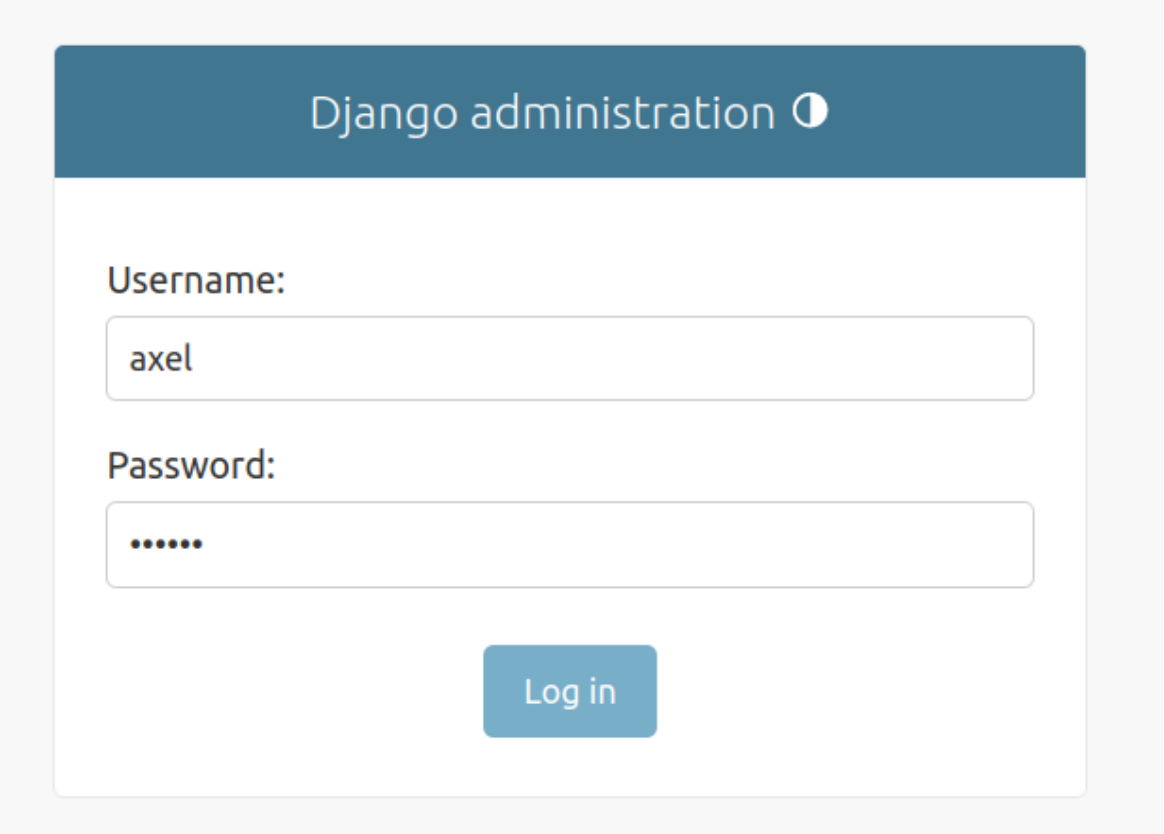


Рис. 6. Сторінка входу у панель адміністратора

Після успішного входу до панелі адміністратора, користувач переадресовується до сторінки керування даними (рис. 7), де він может додавати, вилучати, оновлювати та переглядати дані про користувачів, вакансії, навички та індустрії.

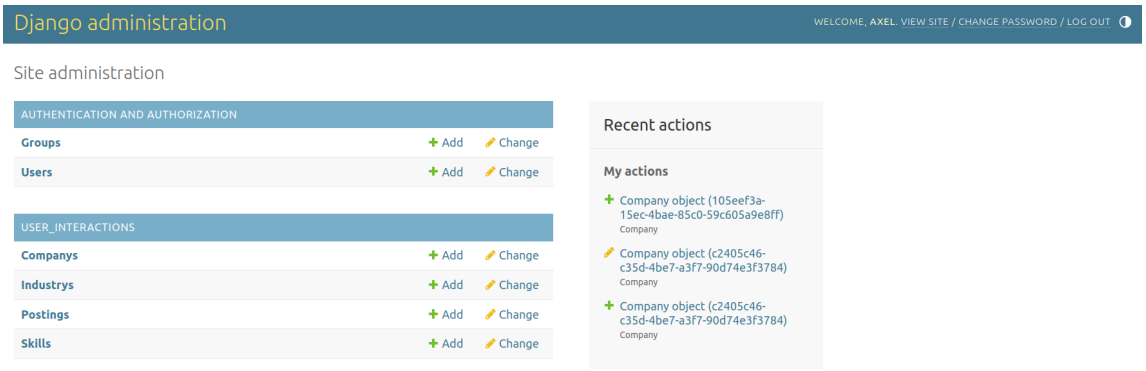


Рис. 7. Сторінка керування даними