

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ
ПОШУКУ РОБОТИ
SOFTWARE DEVELOPMENT FOR A JOB
SEARCH APPLICATION

Виконав: студент 4 курсу, групи КНТ-228

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Павелко О.О.

(прізвище та ініціали)

Керівник Федорончак Т.В.

(прізвище та ініціали)

Рецензент Терещенко Е.В.

(прізвище та ініціали)

2022

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

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

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

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА

Павелко Олександра Олександровича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка програмного застосунку для пошуку роботи.
Software Development for a Job Search Application

керівник проекту (роботи) Федорончак Тетяна Василівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “27” квітня 2022 року № 102

2. Строк подання студентом проекту (роботи) 07 червня 2022 року

3. Вихідні дані до проекту (роботи) рекомендована література, технічне
Завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми.
3. Розробка програми. 4. Експлуатація та тестування програми.

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	Федорончак Т.В., доцент		
Нормоконтроль	Белова А.В., асистент		

7. Дата видачі завдання “ 28 ” лютого 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Експлуатація та тестування програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент

(підпис) Павелко О.О.
(прізвище та ініціали)

Керівник проекту (роботи)

(підпис) Федорончак Т.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
121 с., 5 табл., 68 рис., 3 дод., 18 джерел.

ВЕБСАЙТ, БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ
ДАНИХ, ПОШУК РОБОТИ, ВАКАНСІЯ, КОМПАНІЯ, РОБОТОДАВЕЦЬ,
КАНДИДАТ, PYTHON, DJANGO, SQLITE.

Об'єкт дослідження – системи для пошуку та створення вакансій.

Предмет дослідження – вебсайт, що надає користувачам можливість створення та пошуку робочих місць.

Мета роботи – створення програмної системи для можливості пошуку вакансій для кандидатів на робоче місце з метою знайти роботу та пошуку кандидатів для роботодавців з метою знайти співробітника.

Матеріали, методи та технічні засоби: мова програмування Python, технологія створення вебзастосунків Django, реляційна база даних SQLite та інтегроване середовище розробки PyCharm.

Результати. Створено програмний вебзастосунок, який дозволяє роботодавцям завдяки створенню вакансій знаходити співробітників, а кандидатам на вакансію – робоче місце.

Висновки. Проведено аналіз та порівняння з існуючими подібними застосунками. Розроблено програмну систему, яка призначена для спрощення пошуку роботи та співробітників.

Галузь використання – проект може буде використано для малого та великого бізнесу, підприємств, які хочуть знайти співробітників.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 121 pages, 5 tables, 68 figures, 3 appendices, 18 sources.

WEBSITE, DATABASE, DATABASE MANAGEMENT SYSTEM, JOB SEARCH, VACANCY, COMPANY, EMPLOYER, CANDIDATE, DJANGO, SQLITE.

The object of research is systems for finding and creating vacancies.

The subject of the study is a website that allows users to create and search for jobs.

The purpose of the work is to create a software system for the possibility of finding vacancies for job candidates in order to find a job and search for candidates for employers in order to find an employee.

Materials, methods and technical tools: Python programming language, Django web application development technology, SQLite relational database and PyCharm integrated development environment.

Results. A software web application has been created that allows employers to find employees and job vacancies by creating vacancies.

Conclusions. An analysis and comparison with existing similar applications. A software system has been developed to simplify job search and staffing.

Field of use - the project can be used for small and large businesses, businesses that want to find employees.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	8
ВСТУП	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1 Опис предметної області	10
1.2 Огляд програмних аналогів	10
1.2.1 Програмний продукт «Work.ua»	11
1.2.2 Програмний продукт «Rabota.ua»	12
1.2.3 Програмний продукт «Olx.ua»	13
1.2.4 Порівняльна таблиця існуючих програмних продуктів	14
1.3 Постановка завдання роботи	14
1.4 Висновки до розділу 1	15
2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ	16
2.1 Основні засоби розробки	10
2.2 Вибір сховища даних	18
2.3 Проєктування архітектури програмного забезпечення	19
2.4 Проєктування бази даних	24
2.5 Аналіз і обґрунтування вибору середовища розробки	28
2.6 Висновки до розділу 2	30
3 РОЗРОБКА ПРОГРАМИ	31
3.1 Алгоритм функціонування програмного забезпечення	31
3.2 Опис структури проєкту	34
3.3 Розробка візуальної частини програми	37
3.3.1 Бібліотека «Bootstrap»	38
3.3.2 Бібліотека «FontAwesome»	39
3.3.3 Django Template Language	39
3.4 Висновки до розділу 3	43
4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ	44
4.1 Призначення програми	44

4.2 Тестування програми	45
4.3 Виконання програми.....	47
4.4 Тестування в критичних в середовищах.....	56
4.5 Висновки до розділу 4	56
ВИСНОВКИ.....	58
Перелік джерел посилання	59
ДОДАТОК А Технічне завдання	61
ДОДАТОК Б Текст програми	65
ДОДАТОК В Слайди презентації	109

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- IDE – Integrated Development Environment;
- MVC – Model-View-Controller;
- MTV – Model-Template-View;
- CRUD – Create, Read, Update, Delete operations;
- URL – Uniform Resource Locator;
- HTML – Hyper Text Markup Language;
- БД – база даних;
- СУБД – система управління базами даних;
- ПЗ – програмне забезпечення.

ВСТУП

Умови робочої праці, а також вимоги до неї постійно змінюються. Але саме робота зробила людину – людиною. Згодом робота стала буденністю кожної людини. Саме вона дозволяє людині робити внесок у розвиток та підтримку цивілізації.

Разом з еволюцією людини виник термін «вакансія» – наявність незайнятого робочого місця чи посади, на яку може бути прийнятий новий працівник. Ще якихось 30 – 40 років тому, кожна людина, яка хотіла знайти роботу, повинна була шукати оголошення про неї в листівках або газетах. Не завжди ця людина, читаючи оголошення про роботу, розуміла у чому конкретно полягають вимоги щодо неї і яку саме зарплату вона зможе отримати. Цей процес міг тривати нескінченно довго. На сьогодні користувачеві для пошуку роботи необхідно просто відкрити відповідну вкладку в браузері і знайти вакансію, що зацікавила його.

Проте робота буває різною, а значить кожна людина хоче отримати умови робочого процесу якомога комфортніше. Вдало обране робоче місце відкриває нові межі, впливає на настрій та здоров'я, і навіть визначає умови життя людини найближчим часом. Саме тому кожна людина витрачає багато часу на пошук такого робочого місця, де на її думку вона зможе вдало себе реалізувати.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У сучасному світі неможливо уявити життя без роботи. Однак, щоб отримати бажане робоче місце, потрібно докласти зусиль для його пошуку. Пошук роботи – це важливий, але недооцінений етап, оскільки саме він визначає подальше кар'єрне зростання.

Вільне робоче місце визначається вакансією підприємства чи фірми. Однак, як це часто буває в житті, для отримання бажаного завжди є фактори та критерії до кандидата. У випадку вакансії, до кандидата висувається низка вимог для отримання робочого місця. В основному це ряд певних навичок, спрямованих на область їхнього професійного застосування.

Актуальність даної теми полягає в розробці та створенні програмного забезпечення, що являє собою рішення для пошуку вакансій для тих, хто шукає роботу, і пошуку потенційних працівників для фірм та підприємств.

Створення програмного забезпечення дозволяє заздалегідь визначати дефіцит вакансій та кваліфікованих спеціалістів, тим самим підкреслити важливість їх пошуку та зробити процес їх пошуку більш зручним і швидким.

Метою бакалаврської роботи є покращення візуалізації інформації шляхом розробки вебзастосунку для якісного пошуку та аналізу вакансій. Для більш зручного пошуку вирішувалися такі завдання: можливість реєстрації власної компанії для роботодавця, можливість пошуку потенційних працівників та вакансій за фільтрами, надання детальної інформації про вакансії і кандидатів.

1.2 Огляд програмних аналогів

На сьогодні існує безліч реалізацій у вигляді вебзастосунків, тому для порівняння будуть використані вебсайти для пошуку роботи. Проаналізуємо три найпопулярніші аналоги вебсайтів для пошуку роботи та працівників: «Work.ua», «Rabota.ua», «Olx.ua».

1.2.1 Програмний продукт «Work.ua»

Одним з аналогів розроблюваної програми є програма «Work.ua» [1]. «Work.ua» – це вебзастосунок для пошуку вакансій. Це найпопулярніший застосунок зі зручним та інтуїтивно зрозумілим інтерфейсом. Більшість українців вважають «Work.ua» найкращим сайтом пошуку роботи. У ньому всього в кілька дій кандидат формує запит для пошуку вакансії: вводить назву бажаної вакансії і обирає фільтри для більш точного результату.

На рис. 1.1 наведено користувацький інтерфейс програми «Work.ua».

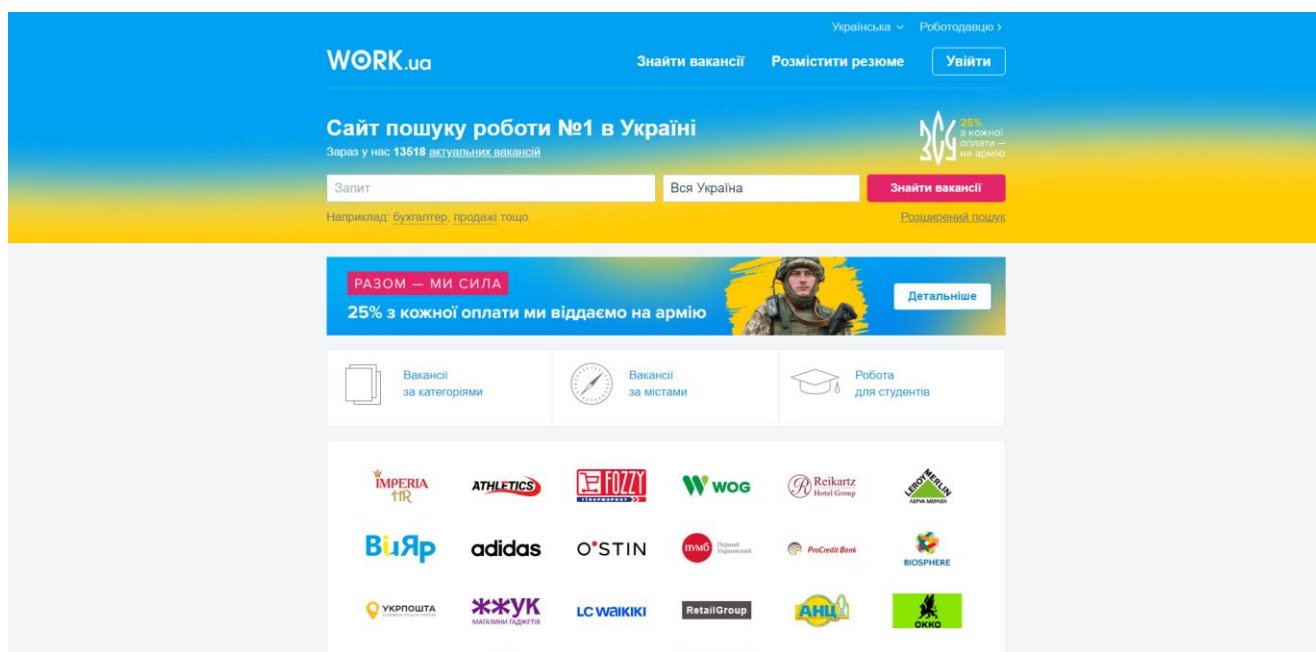


Рисунок 1.1 – Головна сторінка вебзастосунку «Work.ua»

Переваги «Work.ua»:

- можливість реєстрації компанії для подальшого додавання вакансій;
- для підписаних на компанію користувачів є можливість оповіщення про нову вакансію на електронну пошту.

1.2.2 Програмний продукт «Rabota.ua»

Наступним аналогом є «Rabota.ua» [2]. Функціонально він дуже схожий на попередній застосунок. Головними відмінностями є можливість перегляду розташування фірми чи підприємства на карті за вказаною в вакансії адресою та наявність платної версії. Застосунок надає детальну статистику та графіки зарплат.

Платна версія прибирає обмеження на кількість вакансій, які можна переглянути. Так само для роботодавців платна версія може підняти вище позицію вакансії у списку.

На рис. 1.2 наведено користувацький інтерфейс програми «Rabota.ua».

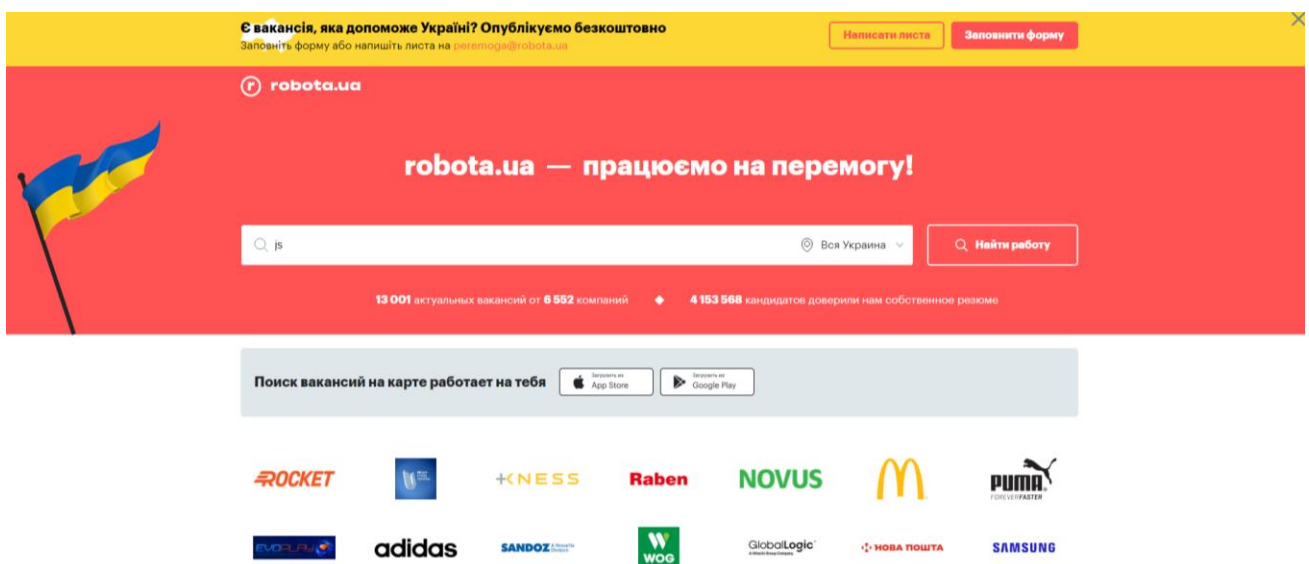


Рисунок 1.2 – Головна сторінка вебзастосунку «Rabota.ua»

Переваги «Rabota.ua»:

- наявність мобільної версії. Мобільна версія дає можливість перегляду адреси вакансії на карті, що дозволяє знайти роботу якомога ближче до місця проживання;
- вбудований конструктор резюме для шукачів роботи. Конструктор пропонує кілька шаблонів на вибір для створення резюме, після чого додає резюме до списку кандидатів.

Головний і, можливо, єдиний недолік програми – платність. Безкоштовна версія обмежує кількість вакансій, яка може бути переглянута користувачем.

1.2.3 Програмний продукт «Olx.ua»

Ще одним аналогом є програма «Olx.ua» [3]. Olx.ua — платформа онлайн-оголошень, яка об'єднує людей для купівлі, продажу чи обміну товарами та послугами. Оголошення класифікуються за такими категоріями, як «Робота», «Нерухомість», «Транспорт», «Запчастини для транспорту», «Тварини», «Будинок та сад», «Електроніка», «Бізнес та послуги», «Мода та стиль», «Віддам задарма» та «Обмін». Сайти OLX відвідують понад 330 мільйонів людей із 40+ країн світу.

На рис. 1.3 наведено користувацький інтерфейс програми «Olx.ua».

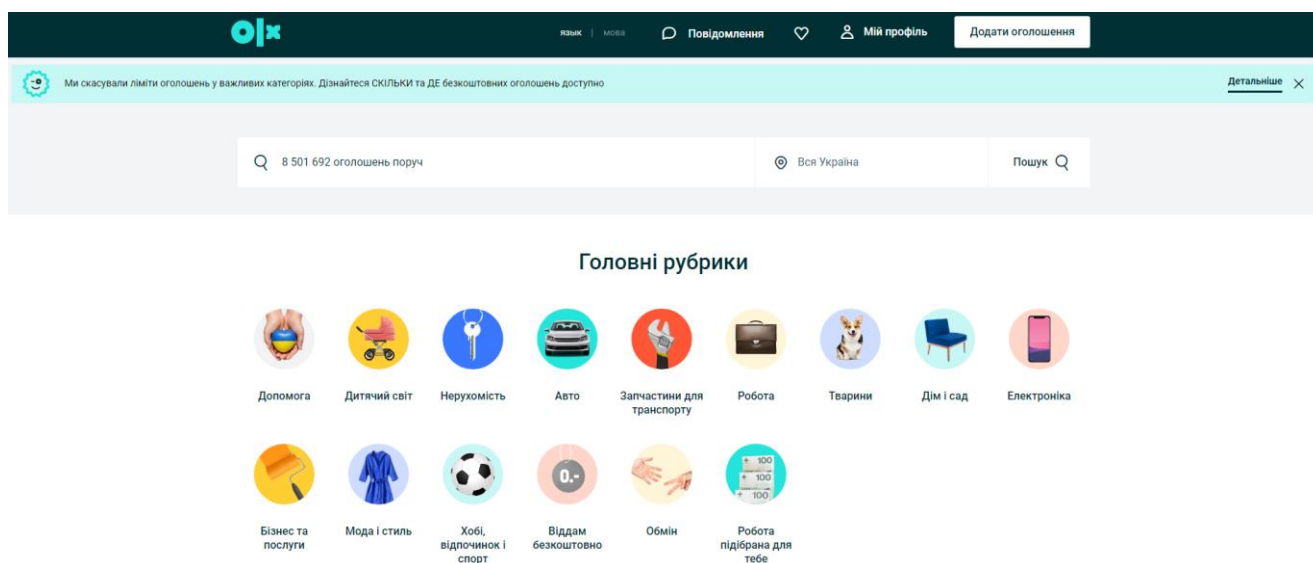


Рисунок 1.3 – Головна сторінка сайту «Olx.ua»

Сильні сторони «Olx.ua»:

- наявність чату між кандидатом та роботодавцем;
- можливість розміщення вакансії без реєстрації власної фірми.

Недоліки «Olx.ua»:

- відсутність фільтрів для пошуку;

- низька популярність сервісу у сфері пошуку роботи.

1.2.4 Порівняльна таблиця існуючих програмних продуктів

Вище було наведено найбільш популярні на сьогоднішній день вебзастосунки, які дозволяють користувачеві знайти роботу. Але крім переваг, більшість програм мають і недоліки. Наприклад, відсутність можливості створення резюме за допомогою вбудованного конструктору або відсутність чату між роботодавцем та кандидатом. Також дійсно існують переваги характерні для усіх програм: всі програми виконують основні функції, підтримують українську мову. В табл. 1.1 проведено порівняння вже існуючих програмних продуктів.

Таблиця 1.1 — Порівняння програмних продуктів

	«Work.ua»	«Rabota.ua»	«Olx.ua»
Виконання основного функціоналу (створення вакансій, перегляд кандидатів)	+	+	-
Багатоплатформність	-	+	+
Зручний інтерфейс	+	+	-
Платність	-	+	-
Підтримка української мови	+	+	+
Пошук вакансій за фільтрами	+	+	-
Вбудований конструктор резюме	-	+	-

1.3 Постановка завдання роботи

Оскільки завданням роботи є реалізація застосунку для пошуку роботи та потенційних кандидатів, програма повинна забезпечувати:

- реєстрацію нового користувача;
- авторизацію в програмі;
- роботу адміністратора;
- створення компанії користувача;
- додавання вакансій до компанії, їх перегляд, редагування та видалення, пошук за фільтрами;
- додавання кандидатів до системи, їх перегляд, редагування та видалення, пошук за фільтрами.

1.4 Висновки до розділу 1

В даному розділі було досліджено предметну область. Проаналізовано основні програмні аналоги та визначено їх базові принципи, переваги і недоліки. На базі цього аналізу було сформовано список переваг, який буде реалізовано при розробці відповідного програмного забезпечення, а саме, безкоштовність, зручний та простий інтерфейс, швидкодія та виконання основних функцій.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Основні засоби розробки

Звичайно, що програмне забезпечення для пошуку роботи має бути представлене у вигляді вебзастосунку, оскільки це підвищить зручність роботи з сервісом, створивши універсальний засіб, який зможуть використовувати як роботодавці, так і кандидати сервісу пошуку роботи з будь-якої точки, з будь-якого пристрою. Для розроблення вебзастосунку було розглянуто два фреймворки, що спрощують даний процес: Django та Spring. Результати порівняння розглянутих фреймворків зведено до табл. 2.1 [4].

Таблиця 2.1 — Порівняння фреймворків для створення вебзастосунку

	«Django»	«Spring»
Область застосування	Веб розробка, штучний інтелект	Веб розробка, системне адміністрування
Мова програмування	Python	Java
Шаблон проектування	Model-Template-View	Model-View-Controller
Швидкість роботи	Більша	Менша
Авторизація користувачів	Вбудована	Необхідно реалізовувати самостійно
Панель адміністратора	Вбудована	Необхідно реалізовувати самостійно
Локалізація текстових ресурсів	Вбудована	Необхідно реалізовувати самостійно

Django створює власну ієрархію будування проекту з одного або кількох додатків, які слід розділяти на мікросервіси. З погляду архітектури побудови вебзастосунків, цей підхід не має аналогів, тому цей фреймворк має багато відмінностей від інших (наприклад, Spring або ASP.NET). Також, що не властиво класичним фреймворкам, обробники URL Django конфігуруються за допомогою спеціальних маршрутів, а не задаються розробником з файлу контролерів.

Архітектура Django багато успадкувала від патерну «Модель-Представлення-Контролер» (MVC). Маючи невеликі відмінності, Django надає

рівень представлення(View), який в свою чергу реалізовується в MVC рівнем контролеру моделі. Подання зовнішньої частини реалізується в Django за допомогою рівня шаблонів (Templates). Через цю рівневу архітектуру Django нерідко називають «Модель-Шаблон-Представлення» (MTV) [5], [6].

Технологія Django має наступні позитивні характеристики і переваги:

- швидкість розробки: в порівнянні з іншими вебфреймворками, Django надає найбільш швидку реалізацію базового функціоналу. Такий функціонал може включати проектування архітектури проекту, автоматичну реалізацію CRUD операцій, тощо. При цьому Django надає можливість розширення базового функціоналу та економить час і ресурси на кожному з цих етапів. Таким чином, цей фреймворк можна назвати найкращим рішенням для розробників, для яких питання швидкої реалізації є одним із головних;
- повна комплектація: Django надає купу додаткових функцій, які сприяють швидкій реалізації таких компонентів вебзастосунку, як авторизація користувача, адмін-панель, локалізація. Завдяки цим аспектам можна швидко здійснити кожен з етапів розробки;
- багатofункціональність: менеджер контенту, складні міграції до бази даних, наукові обчислювальні інструменти, автодоповнення коду — все це Django надає за замовченням.

Враховуючи, що фреймворк Django дозволяє створювати вебзастосунки максимально швидко, а також може використовуватися на рівні підприємств, що важливо для сервісу пошуку роботи, саме його було обрано в якості вебфреймворку для розробки вебсайту. Даний фреймворк може реалізовуватися тільки за допомогою мови Python, одже саме її було обрано. В результаті порівняння з іншими мовами програмування було обрано саме мову Python, оскільки вона:

- підтримує безліч підходів програмування, зокрема об'єктно-орієнтоване, процедурне, функціональне, структурне, тощо;
- дозволяє розділяти проект на модулі, які можна об'єднувати в пакети;

- дозволяє отримати інформацію про тип і внутрішню структуру будь-якого об'єкта в процесі роботи програми;
- надає велику стандартну бібліотеку, яка має набір модулів для роботи з різними мережевими протоколами, шифруванням файлів, базами даних, текстовими кодуваннями, регулярними виразами, а також підтримує юніт-тестування;
- включає різноманітні функції, такі як ітератори та лямбда-вирази, обробку винятків, управління контекстом виконання, декоратори та багато іншого;
- вимагає набагато менший обсяг написаного коду, ніж у випадку з іншими популярними мовами (Java, C#). Це помітно прискорює розробку, дозволяючи створювати складні системи швидше, ніж іншими мовами програмування;
- є універсальним. Python – це мова, що інтерпретується, вона використовується для розробки практично під усі сучасні платформи. Він не потребує перекомпіляції та його код можна писати у звичайному текстовому документі [7].

2.2 Вибір сховища даних

У якості сховища даних буде доцільно використовувати реляційну СУБД (систему управління базами даних), тобто таку, яка надасть можливість створювати зв'язки між таблицями бази даних. По-перше, це спростить процес розробки бази даних, так як реаліційна СУБД вже буде мати реалізований формат зберігання даних. По-друге, модель даних саме реляційної СУБД може бути легко перетворена на об'єкт таблиці бази даних, а потім так само легко відображена на відповідній сторінці. По-третє, Python надає велику кількість ORM-бібліотек для роботи саме з реляційними СУБД, такими, як SQLite, PostgreSQL, MySQL [8].

Але кожна з них має істотні відмінності, такі як тип бази даних, наявність функцій, що розширюють функціонал взаємодії з базою даних та підтримувані мови.

В табл. 2.2 проведено порівняння найпопулярніших реляційних СУБД.

Таблиця 2.2 — Порівняння реляційних СУБД

	SQLite	MySQL	PostgreSQL
Тип бази даних	Файлова	Серверна	Серверна
Операційна система сервера	Безсерверна	Linux, Windows, OS X, Solaris, FreeBSD	FreeBSD, Linux, Windows, OS X, Solaris, Unix
Підтримувані мови	ActionScript, Python, C, C#, C++, PHP, Java, JavaScript,	C, C++, Delphi, Perl, Java, PHP, Lua, .NET, Node.js	.NET, C, C++, Delphi, Java, Python, PHP, Perl
Видання	Community з функцією PRO підтримки	Community free standard та Enterprise edition	Community з функцією PRO допомоги

Використовуючи результати порівняння у табл. 2.2 було обрано СУБД SQLite.

2.3 Проєктування архітектури програмного забезпечення

Концепція патерну Model – Template – View (MTV) передбачає поділ проєкту на три компоненти (рис. 2.1):

- Model представляє клас, що забезпечує доступ до даних, обробляється шаром роботи з базою даних. Модель є незалежною від інших компонентів, будь-які зміни інших файлів проєкту не вплинуть на роботу моделі;
- View – це частина, яка визначає які дані отримувати та як їх відображати, обробляється уявленнями та шаблонами;
- Template – це візуальна частина або призначений для користувача інтерфейс програми, який викликається слідуючи до створеної схеми URL

шляхів, і викликає відповідну функцію Python для зазначеного URL, яка на виході відображає html-сторінку, яку користувач бачить, зайшовши на сайт [9].

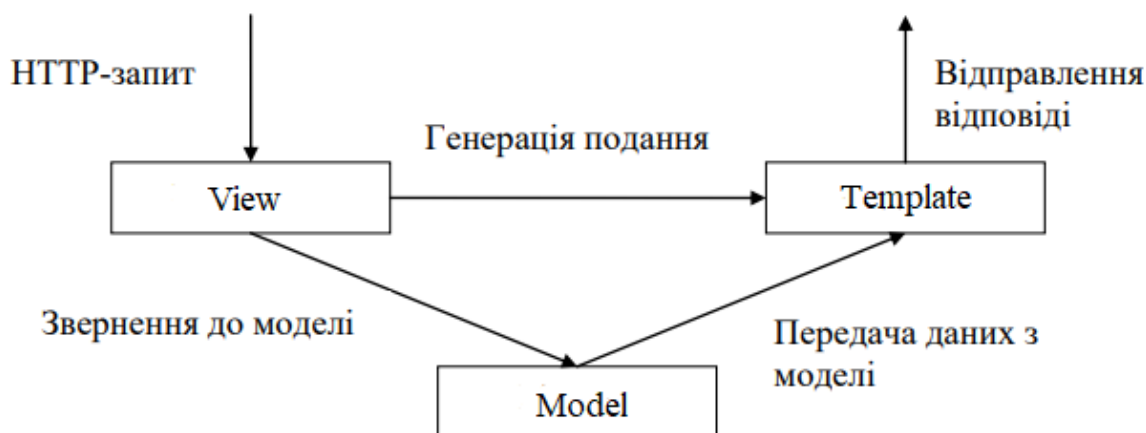


Рисунок 2.1 – Шаблон MTV

Для реалізації функціоналу програми розроблювалися такі модулі:

- модуль база даних;
- модуль адміністрування сайтом;
- модуль реєстрація;
- модуль аутентифікація;
- модуль створення основних сутностей;
- модуль створення сторінок основних сутностей;
- модуль пошуку вакансій та кандидатів за фільтрами.

Модуль “База даних” включає основні таблиці, такі як: користувач, кандидат, роботодавець, компанія, вакансія та допоміжні таблиці, такі як: тип зайнятості, тип кандидату, країна, місто, валюта, категорія.

Модуль “Реєстрації” призначений для розмежування прав доступу, а саме можливості користуватися сайтом.

Модуль “Аутентифікація” призначений для входу, а саме дає можливість користуватися сайтом залежно від типу особистого запису.

Модуль “Створення основних сутностей” призначений для введення інформації шляхом заповнення відповідних форм для подальшого додавання сутностей.

Модуль “Створення сторінок основних сутностей” призначений для створення відповідних сторінок для відображення основних сутностей.

Модуль ”Пошук вакансій та кандидатів за фільтрами” призначений для більш точного пошуку вакансій та кандидатів.

Всі користувачі були розділені на три великі групи: адміністратори, роботодавці та кандидати. Адміністратори мають розширені права роботи з системою. Роботодавці можуть зареєструвати свою компанію, для якої надалі зможуть додавати вакансії та контакти, а також зможуть скористатися пошуком кандидатів на робоче місце з можливістю їх збереження. Кандидати обмежені пошуком вакансій, переглядом усіх вакансій, конкретної вакансії та компанії, прив'язаної до певної вакансії. Кандидати так само можуть зберігати вакансії, що сподобалися. Користувачам, які не авторизувалися доступний функціонал перегляду вакансій. На рис. 2.2 зображена структура модулів програми.

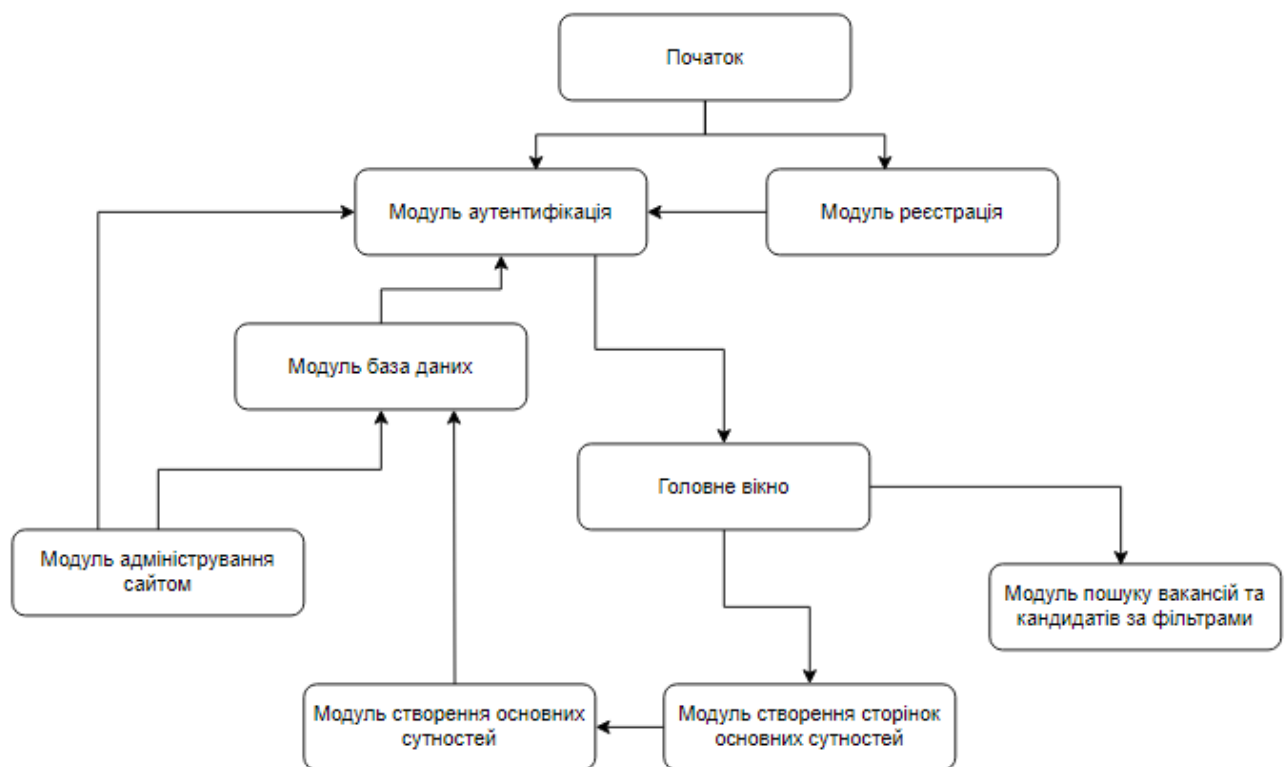


Рисунок 2.2 – Структура модулів програми

На рис. 2.3 наведено діаграму прецедентів для роботи адміністраторів з системою.

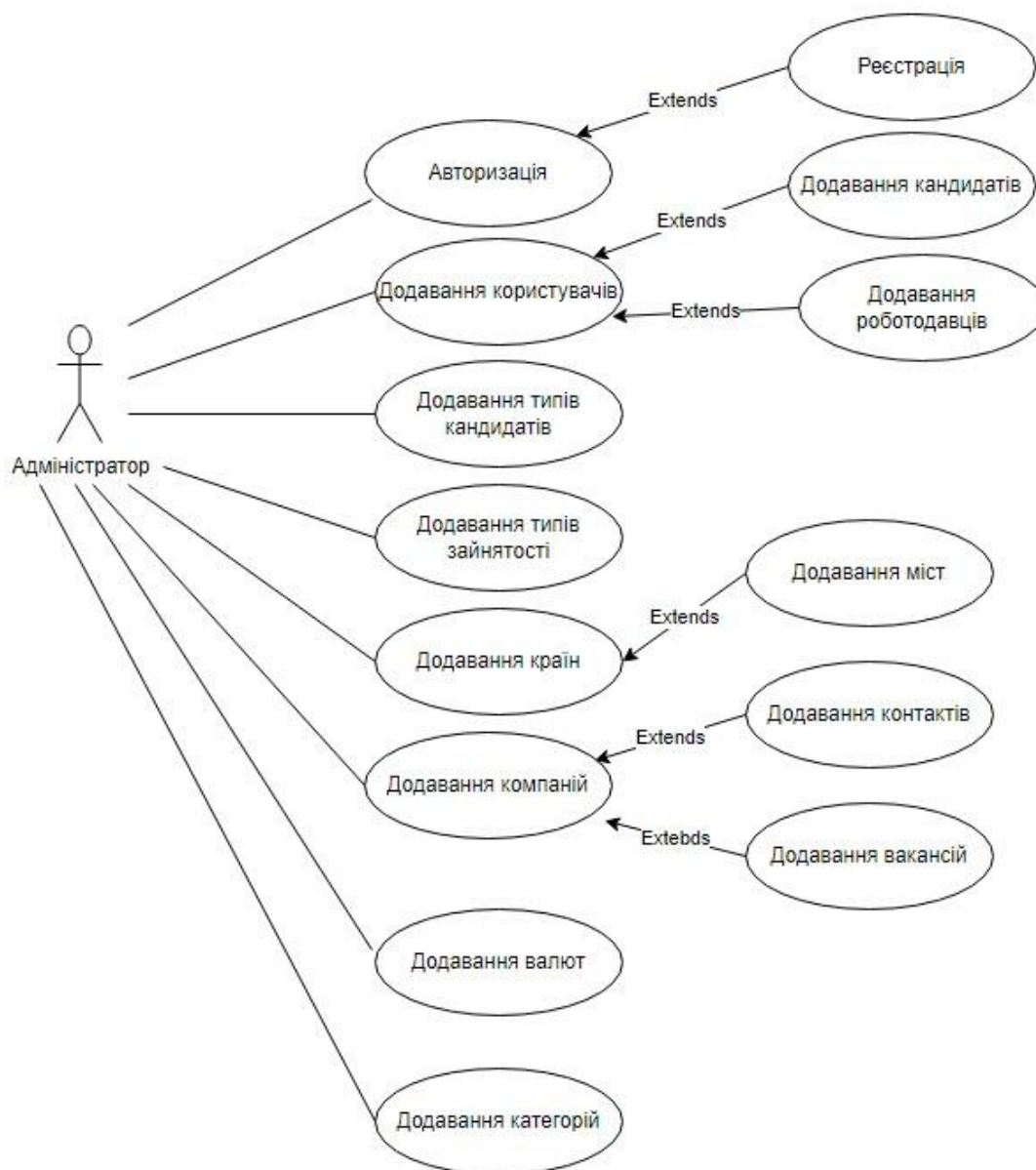


Рисунок 2.3 – Діаграма прецедентів «Робота адміністратора з системою»

На рис. 2.4 наведено діаграму прецедентів для роботи роботодавців з системою.

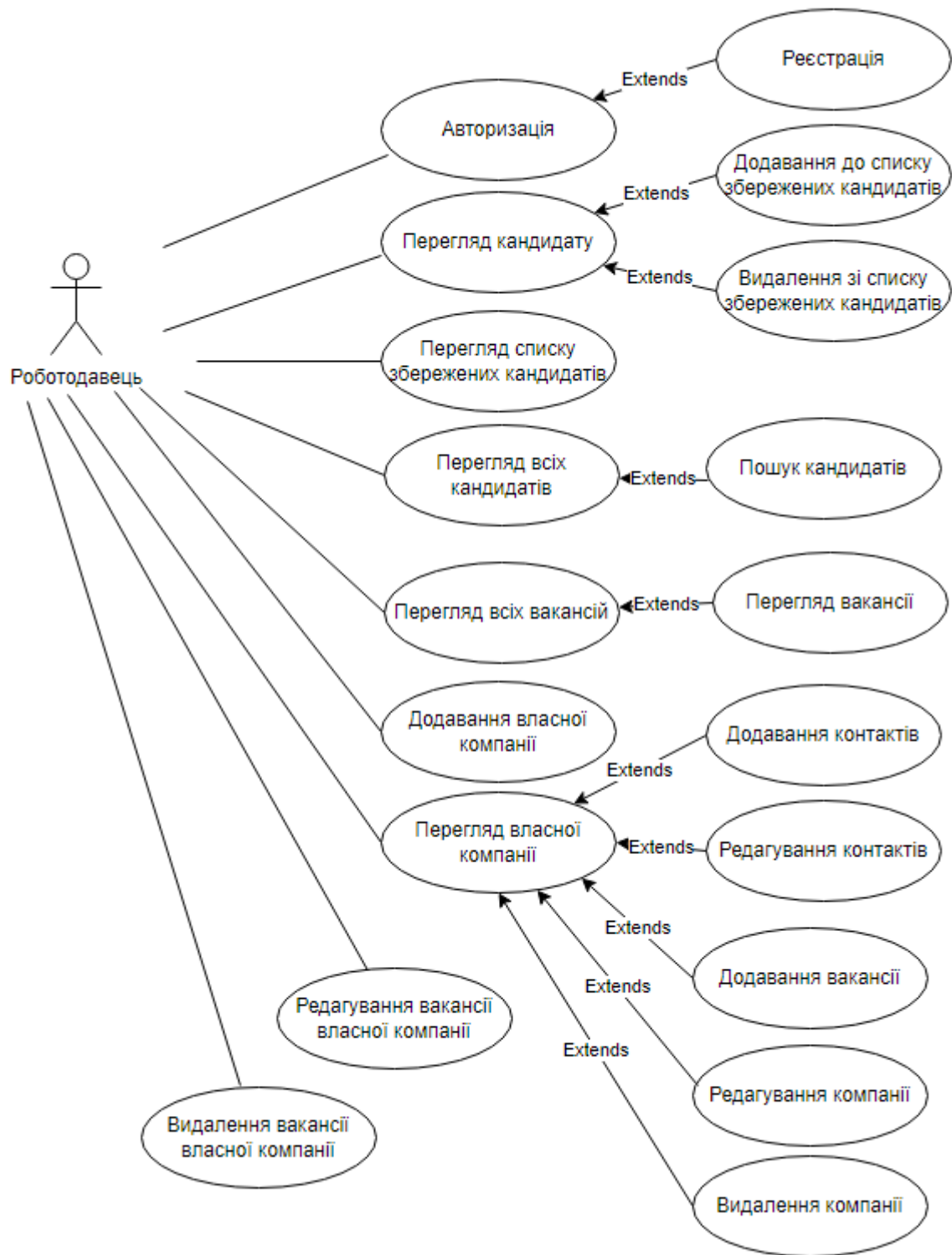


Рисунок 2.4 – Діаграма прецедентів «Робота роботодавця з системою»

На рис. 2.5 наведено діаграму прецедентів для роботи кандидатів з системою.

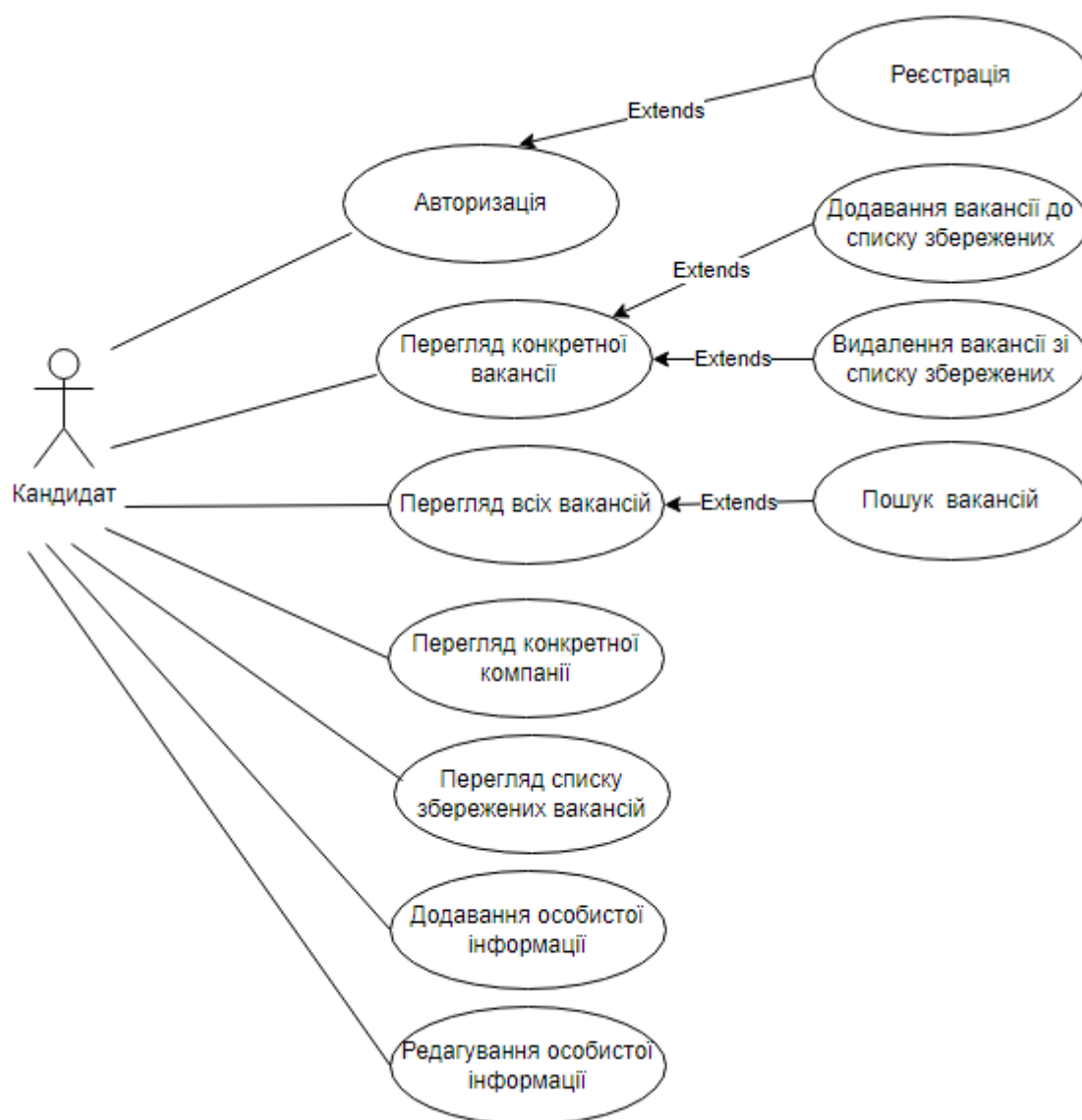


Рисунок 2.5 – Діаграма прецедентів «Робота кандидата з системою»

2.4 Проектування бази даних

Робота вебзастосунку передбачає створення бази даних для збереження всіх необхідних даних між сеансами роботи програми. За результатами проведеного аналізу предметної області було виділено наступні сутності:

- User – користувач;
- Employer – роботодавець;
- Candidate – кандидат на отримання вакансії;
- Vacancy – вакансія;
- Company – компанія роботодавця;

- Contacts – контакти компанії;
- EmploymentType – тип зайнятості;
- CandidateType – тип кандидату;
- Currency – валюта;
- Category – категорія;
- City – місто;
- Country – країна.

Сутність User повинна мати наступні поля:

- id – унікальний ідентифікаційний номер користувача;
- is_candidate – позначає чи є користувач кандидатом;
- is_employer – позначає чи є користувач роботодавцем;
- candidate – кандидат, до якого прив'язаний користувач;
- employer – роботодавець, до якого прив'язаний користувач.

Сутність Employer повинна мати наступні поля:

- id – унікальний ідентифікаційний номер роботодавця;
- creation_date – дата створення роботодавця;
- saved_candidates – збережені кандидати;
- company – компанія роботодавця.

Сутність Candidate повинна мати наступні поля:

- id – унікальний ідентифікаційний номер користувача;
- phone – номер телефону кандидата;
- creation_date – дата створення кандидата;
- edu_lvls – масив підказок рівнів освіти;
- education_level – рівень освіти кандидата;
- exp_lvls – масив підказок рівнів досвіду;
- experience_level – досвід кандидата;
- image – фото кандидата;
- saved_vacancies – збережені вакансії;
- desired_vacancy – бажана вакансія кандидата;
- city – місто де проживає кандидат.

Сутність Vacancy повинна мати наступні поля:

- id – унікальний ідентифікаційний номер вакансії;
- title – назва вакансії;
- description – опис вакансії;
- employment_type – тип зайнятості;
- creation_date – дата створення вакансії;
- salary – заробітня плата;
- currency – валюта;
- candidate_types – типи кандидатів, які підходять до вакансії;
- edu_lvls – масив підказок рівнів освіти;
- education_level – рівень освіти кандидата;
- exp_lvls – масив підказок рівнів досвіду, які підходять до вакансії;
- experience_level – рівень досвіду, необхідний для отримання вакансії;
- categories – категорії вакансії;
- company – компанія.

Сутність Company повинна мати наступні поля:

- id – унікальний ідентифікаційний номер компанії;
- title – назва компанії;
- website_url – адреса вебсайту компанії;
- image – картинка компанії;
- categories – категорії, в яких спеціалізується компанія ;
- contacts – контакти компанії.

Сутність Contacts повинна мати наступні поля:

- id – унікальний ідентифікаційний номер контактів;
- phone1 – перший номер телефону;
- phone2 – другий номер телефону;
- email – адреса електронної пошти.

Сутність CandidateType повинна мати наступні поля:

- id – унікальний ідентифікаційний номер типу кандидата;
- title – назва типу кандидата.

Сутність EmploymentType повинна мати наступні поля:

- id – унікальний ідентифікаційний номер типу зайнятості;
- title – назва типу зайнятості.

Сутність Currency повинна мати наступні поля:

- id – унікальний ідентифікаційний номер валюти;
- title – назва валюти.

Сутність Category повинна мати наступні поля:

- id – унікальний ідентифікаційний номер категорії;
- title – назва категорії.

Сутність County повинна мати наступні поля:

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

Сутність City повинна мати наступні поля:

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

Саме така структура таблиць дозволяє застосовувати різні підходи користування сайтом як роботодавцю, так і кандидату. Наприклад, роботодавець відразу після реєстрації повинен зареєструвати свою компанію, після чого зможе додавати до неї вакансії та контакти для зв'язку з кандидатами. Кандидат, у свою чергу, після реєстрації повинен надати свої персональні дані, такі як місто проживання, фотографія, рівень освіти, рівень досвіду роботи та інформацію про бажану вакансію. Також кандидат зможе переглядати компанії, шукати вакансії, зберігати їх собі.

Результуючу схему бази даних наведено на рис. 2.6. Для проєктування було використано систему управління базами даних SQLite, оскільки вона надає всі необхідні засоби для роботи з реляційними базами даних, а виходячи з наявності вебзастосунку, не виникає необхідності забезпечення віддаленого доступу до даних саме засобами системи управління базами даних.

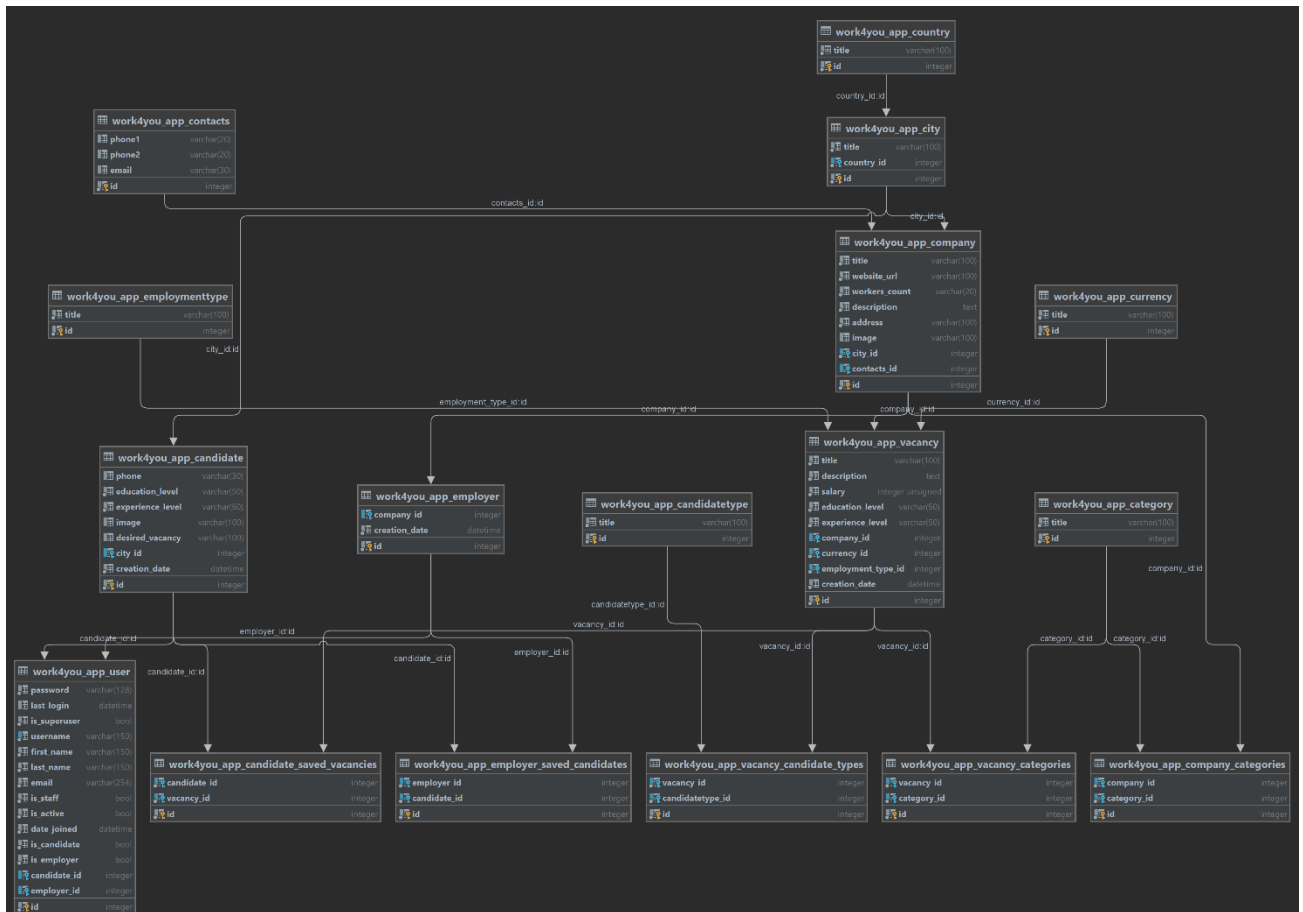


Рисунок 2.6 – Схема бази даних

2.5 Аналіз і обґрунтування вибору середовища розробки

Для розробки ПЗ мовою програмування Python на сьогодні існує велика кількість IDE (інтегрованих середовищ розробки) та редакторів. Відмінність між ними в тому, що інтегроване середовище завантажує весь проект цілком, тому може надавати зручну навігацію між його файлами. А редактори редагують конкретний файл і самі файли не знають про зв'язки між іншими файлами та це і є головною складністю при розробці ПЗ. Розглянемо деякі популярні з них, а саме Visual Studio, PyCharm та Subline Text 3 [10].

PyCharm демонструє себе, як інтегроване середовище розробки для мови Python. Середовище PyCharm надає такі засоби: графічний інтерфейс, термінал та систему контролю версіями. PyCharm розроблений компанією JetBrains на основі IntelliJ IDEA, має потужне ком'юніті, блог, та підтримується більшістю сучасних операційних систем [11].

Visual Studio Code – це стартовий майданчик для написання, налагодження та складання коду, а також подальшої публікації програм. Крім стандартного редактора і відладчика, які є в більшості середовищ IDE, Visual Studio включає термінали, компілятори, засоби автозавершення коду, графічні конструктори і багато інших функцій для поліпшення процесу розробки. Однією з суттєвих переваг Visual Studio Code є можливість працювати з системою контролю версій, не виходячи з самого редактору [12].

Sublime Text 3 – це повнофункціональний текстовий редактор. Підходить для редагування локальних файлів, а також підтримує плагіни для програмування мовою програмування Python. Sublime Text 3 надає можливість завантажувати різні плагіни та персоналізувати свою середу розробки. Різні функції, які підтримуються в Sublime Text 3: підсвічування синтаксису, автоматичний відступ, розпізнавання типів файлів, плагіни та пакети. Але він має суттєвий недолік, а саме відсутність динамічного автодоповнення коду [13].

Порівняння вищеописаних середовищ розробки наведено у табл. 2.3.

Таблиця 2.3 – Порівняння вищеописаних середовищ розробки

	PyCharm	Visual Studio	Sublime Text 3
Тип програми	Інтегроване середовище розробки	Інтегроване середовище розробки	Текстовий редактор
Доступні платформи	Windows, Linux, OS X	Windows, Linux	Windows, Linux, OS X
Автодоповнення коду	+	+	+
Термінал	+	+	-
Відлагоджувач Windows	+	+	-
Система контролю версіями	+	+	-
Можливість створення графічних інтерфейсів	Не вбудована в середовище, але є можливість розширення	Вбудована в середовище	Не вбудована

Враховуючи особливості, наведені вище, а також наявність великої кількості інструментів для швидкої розробки вебзастосунку мовою Python для розробки було обрано інтегроване середовище розробки PyCharm. Головним фактором обрання PyCharm є те, що він надає відлагоджувач Windows та дозволяє використовувати термінал безпосередньо у процесі розробки.

2.6 Висновки до розділу 2

Проектування програмного забезпечення для сервісу пошуку роботи було виконано у вигляді вебзастосунку. Для реалізації вебзастосунку було використано фреймворк Django, попередньо виконавши аналіз альтернатив для реалізації вебзастосунків. Для реалізації даної технології було прийнято рішення використовувати мову програмування Python. Визначено функціональні вимоги до програмного забезпечення та виконано їх моделювання за допомогою діаграм прецедентів.

3 РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програмного забезпечення

Функціонал вебсайту напряду залежить від ролі користувача. При розробці системи було визначено три можливі ролі користувача: адміністратор, роботодавець та кандидат на робоче місце. Незареєстрований користувач за замовчуванням вважається кандидатом, однак йому недоступні деякі функції, наприклад такі, як збереження вакансій. В подальшому нижче розглянуто стандартний потік, тобто використання всіх функціональних можливостей програмного забезпечення. Для кожної такої можливості перевіряється, чи дійсно користувач з таким обліковим записом може її використовувати. Якщо ні, то робота з нею не відбувається.

Спочатку відбувається реєстрація облікового запису. Для коректної роботи системи треба спочатку її наповнити даними. Це основна робота адміністратора сайту. Створити адміністратора в Django можна однією командою у командному рядку — `python manage.py createsuperuser`. Після виклику цієї команди треба ввести логін, електронну пошту та пароль. Створивши адміністратора можна авторизуватись як адміністратор системи та заповнити таблиці необхідними даними. На рис. 3.1 наведено функціональну схему програми при авторизації адміністратора.

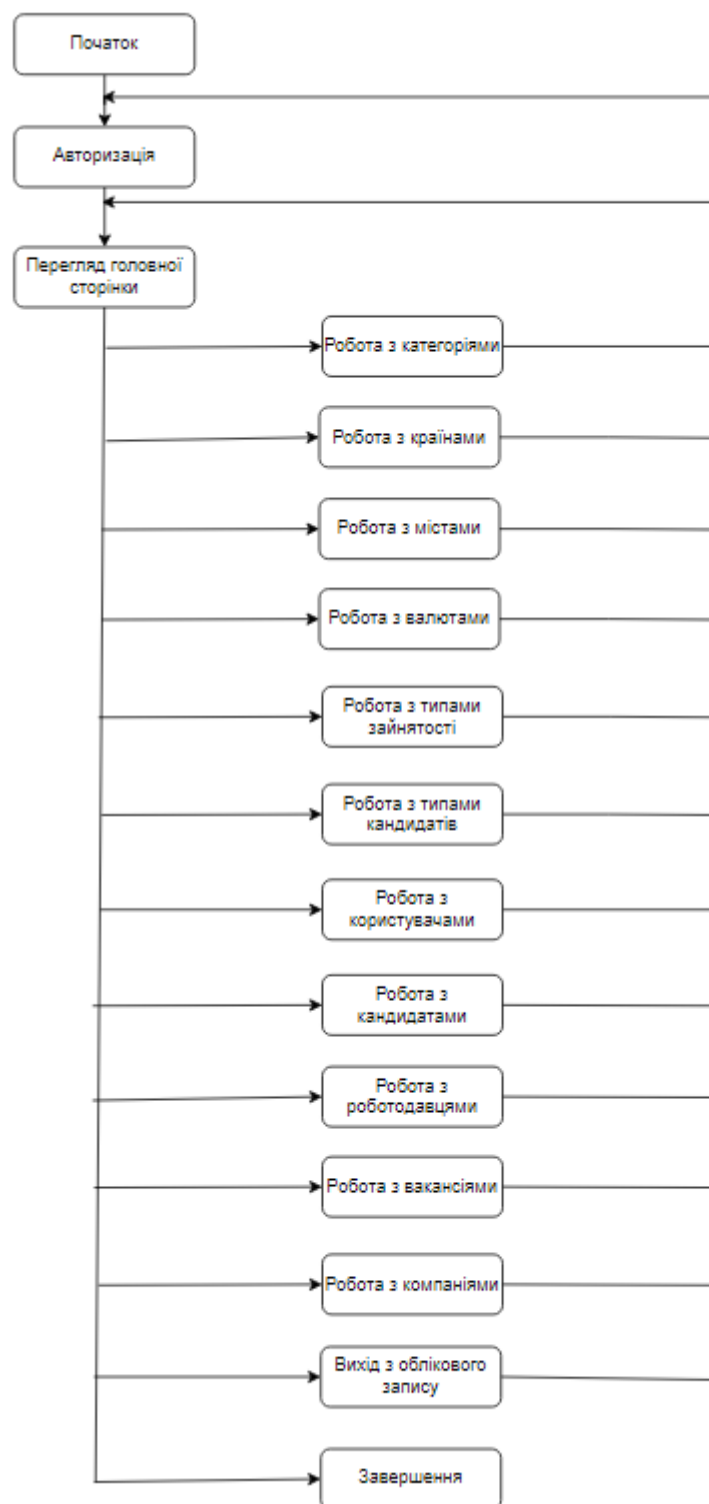


Рисунок 3.1 – Функціональна схема програми роботи адміністратора

Далі користувач може зареєструватися в системі, після чого авторизуватися, обравши при цьому роль профілю, за якою він хоче увійти до свого облікового запису: кандидат чи роботодавець.

Вказавши роботодавця як роль профілю, користувач відразу потрапляє на

сторінку реєстрації своєї компанії, заповнює всі необхідні поля, після чого компанія вважається зареєстрованою, а користувач потрапляє на сторінку її перегляду. На цій сторінці йому буде доступне додавання контактних даних та вакансій своєї компанії, їх перегляд, а також можливість редагування та видалення як самої компанії, так і окремо взятих її вакансій. Також користувачеві який вибрав роботодавця як роль профілю буде доступний пошук кандидатів для своєї компанії. Цією функцією можна скористатися, перейшовши за відповідним посиланням з навігаційного меню, після чого, ввівши необхідний запит і вибравши необхідні параметри, система знайде необхідних кандидатів. Вибравши необхідного, можна подивитися його персональні дані, такі як рівень освіти, досвіду роботи, а також номер телефону та адресу електронної пошти. На рис. 3.2 наведено функціональну схему програми при виборі профілю роботодавця.

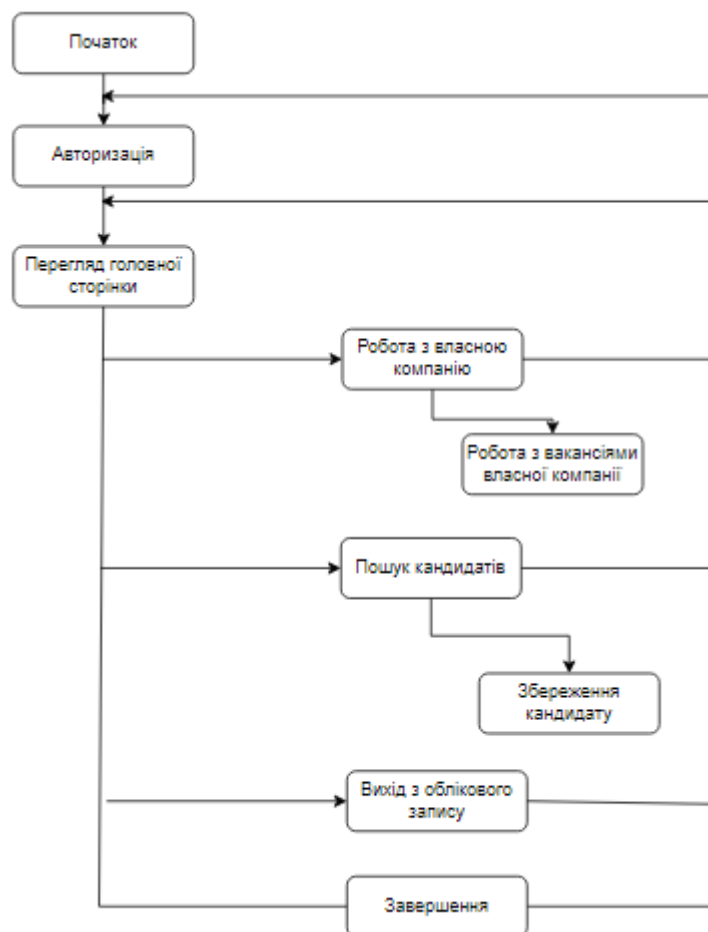


Рисунок 3.2 – Функціональна схема програми роботи роботодавця

Як було сказано вище, при авторизації можна також обрати роль профілю кандидата. В такому випадку користувач відразу потрапить на сторінку з формою заповнення особистої інформації. Завершивши цей етап, користувач зможе переглядати та шукати вакансії, зберігати їх, а також переглядати прив'язані до них компанії. На рис. 3.3 наведено функціональну схему програми при виборі профілю кандидата.

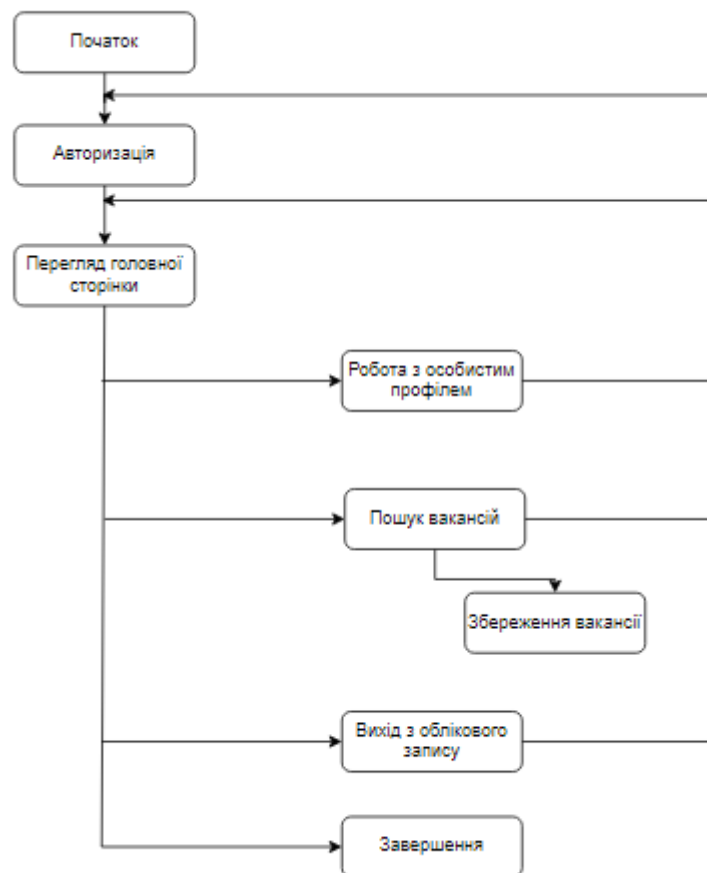


Рисунок 3.3 – Функціональна схема програми роботи кандидата

3.2 Опис структури проекту

На попередньому етапі була реалізована архітектура проекту. Структура кореневої папки проекту наведена на рис. 3.4.

work4you	update .gitignore	7 hours ago
work4you_app	add authorization, user roles, entity pages, change styles	7 hours ago
.gitignore	update .gitignore	7 hours ago
manage.py	create models, project files	15 days ago

Рисунок 3.4 – Структура кореневої папки проекту

Весь код, що стосується налаштувань проекту, винесено в окремий пакет «work4you», а вся логіка застосунку винесена в пакет «work4you_app».

Пакет «work4you» містить файли `asgi.py` та `wsgi.py`, що відповідають за налаштування, конфігурацію проекту та інформацію про встановлені модулі, а файл `urls.py` містить всі можливі маршрути, доступні користувачу [14]. Структура класів пакету наведена на рис. 3.5.

__init__.py	create models, project files	16 days ago
asgi.py	update .gitignore	23 hours ago
urls.py	add authorization, user roles, entity pages, change styles	23 hours ago
wsgi.py	update .gitignore	23 hours ago

Рисунок 3.5 – Структура класів пакету «work4you»

Міграції бази даних, файли сторінок а також статичні файли було розміщено у директоріях «migrations», «templates/work4you» та «static» пакету «work4you_app» відповідно. Також цей пакет містить основні структурні класи проекту, такі, як `admin.py`, `apps.py`, `forms.py`, `models.py` та `views.py`. Структура класів пакету наведена на рис. 3.6.

migrations	add authorization, user roles, entity pages, change styles	22 hours ago
static	add authorization, user roles, entity pages, change styles	22 hours ago
templates/work4you_app	add authorization, user roles, entity pages, change styles	22 hours ago
__init__.py	create models, project files	16 days ago
admin.py	add authorization, user roles, entity pages, change styles	22 hours ago
apps.py	create models, project files	16 days ago
forms.py	add authorization, user roles, entity pages, change styles	22 hours ago
models.py	add authorization, user roles, entity pages, change styles	22 hours ago
views.py	add authorization, user roles, entity pages, change styles	22 hours ago

Рисунок 3.6 – Структура класів пакету «work4you_app»

Така структура файлів пакету, що відповідає за логіку застосунку, є стандартною в проєкті Django. Клас `admin.py` відповідає за сутності, які будуть доступні адміністратору системи для керування їх поведінкою через адмін-панель. Файл `apps.py` зберігає конфігурацію налаштувань проєкту, а `forms.py` зберігає в собі класи форм, які прив'язані до моделей. Класи форм спрощують створення моделей шляхом генерації HTML-форми для заповнення вказаних полів моделі. Клас `models.py` містить в собі всі сутності у вигляді класів, які Django конвертує в об'єкти таблиць бази даних. В свою чергу файл `views.py` відповідає за логіку застосунку, в ньому містяться функції, які викликаються при відправленні користувачем відповідного запиту через адресний рядок браузера. Оскільки цей клас містить логіку застосунку, важливо описати саме його функції. В процесі реалізації було визначено наступні функції:

- `register_page(request)` – метод, який відповідає за реєстрацію користувача шляхом переходу на сторінку реєстрації. Функція, як і усі інші функції цього класу приймає першим параметром `request`, який зберігає параметри рядку запитів;
- `login_page(request)` – метод, який забезпечує авторизацію користувача. Авторизуватися можна лише як роботодавець або кандидат на робоче місце. Залежно від вибору, користувачеві буде доступний різний функціонал;
- `logout_user(request)` – метод, який відповідає за вихід з облікового запису;
- `get_vacancies(request)` – метод, який повертає всі існуючі вакансії;
- `get_vacancy(request, vacancy_id)` – метод, який повертає вакансію за вказаним `id`;
- `get_company_vacancies(request, company_id)`, – метод, який повертає всі вакансії компанії за її вказаним `id`;
- `find_vacancies(request)` – метод, який викликається після того, як користувач скористався пошуком вакансій. Відповідно до запиту за вказаними фільтрами, будуть знайдені вакансії;

- `get_company(request, company_id)` – метод, який повертає компанію за вказаним `id`;
- `add_company(request)` – метод, який додає компанію до бази даних;
- `add_company_vacancy(request, company_id)` – метод, який додає вакансію до компанії за її вказаним `id`;
- `get_candidates(request)` – метод, який повертає всіх кандидатів;
- `get_candidate(request, candidate_id)` – метод, який повертає кандидата за вказаним `id`;
- `add_candidate(request)` – метод, який додає кандидата до бази даних;
- `save_vacancy(request, candidate_id, vacancy_id)` – метод, який додає вакансію до списку збережених для кандидата за вказаним `id`;
- `delete_vacancy(request, vacancy_id)` – метод, який видаляє зі списку збережених вакансію кандидата за вказаним `id`;
- `get_saved_vacancies(request, candidate_id)` – метод, який повертає список збережених вакансій для кандидата;
- `add_contacts(request, company_id)` – метод, який дозволяє роботодавцю додати контакти до своєї компанії;
- `edit_contacts(request, company_id)` – метод, який дозволяє роботодавцю редагувати контакти своєї компанії.

3.3 Розробка візуальної частини програми

Оскільки на попередніх етапах було визначено основні моделі системи, такі, як вакансія (`Vacancy`), компанія (`Company`), та кандидат (`Candidate`), саме для них будуть створені відповідні сторінки.

Для структурованості проекту в пакеті «`work4you_app`» в директорії «`templates/work4you_app`» було створено додаткові директорії, в яких будуть зберігатися HTML-сторінки моделей. Структура HTML файлів директорії «`templates/work4you_app`» наведена на рис. 3.7.

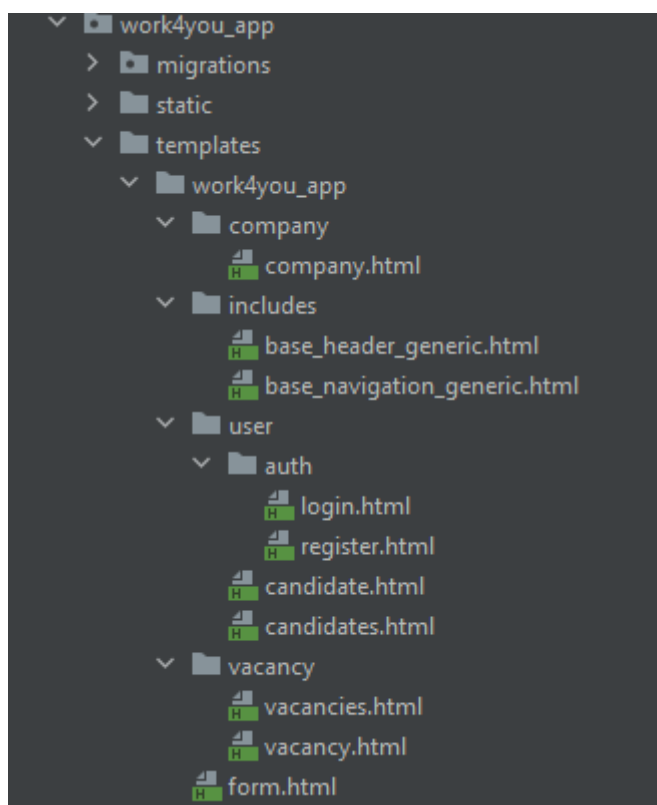


Рисунок 3.7 – Структура HTML файлів директорії «templates/work4you_app»

3.3.1 Бібліотека Bootstrap

Оскільки система не потребує складного графічного інтерфейсу, основною технологією створення стилів для сторінок став Bootstrap. Bootstrap – це вільний набір інструментів для створення сайтів та вебзастосунків. Включає HTML та CSS шаблони оформлення для форм, кнопок, міток, блоків навігації та інших компонентів інтерфейсу, включаючи JavaScript-розширення [15].

Використання фреймворків та бібліотек значно полегшує роботу розробникам та дозволяє розробляти проекти швидше. Bootstrap пропонує готові рішення: їх використання дозволяє скоротити час, що витрачається на створення макета сайту.

Bootstrap дозволяє створювати адаптивні вебсайти. Дизайн сайту коректно відображатиметься на екранах пристроїв різних розмірів незалежно від їх діагоналі. Також сайти, створені за допомогою Bootstrap, будуть однаково відображатися у всіх сучасних браузерах.

Bootstrap дозволяє писати якісний та зрозумілий код, який легко зрозуміє інший розробник. Це значно спрощує розробку у команді.

3.3.2 Бібліотека FontAwesome

FontAwesome – бібліотека, яка дозволяє використовувати спеціальні масштабовані векторні іконки на вебсторінках. Оскільки іконки FontAwesome векторні, вони відображатимуться коректно на екрані будь-якого розміру [16].

Векторна графіка – це метод створення комп'ютерного зображення, яке будується за допомогою обробки математичного опису у спеціальному програмному забезпеченні. На відміну від растрового малюнка, який представляє набір пікселів різних кольорів, векторне зображення – це набір примітивів (ліній, кривих, квадратів, кіл), описаних математичними формулами. Малюнок формується у вигляді файлу, який містить дані про координати точок та про лінію, що перетинає опорні точки. Окрім цього, файл містить інформацію про застосовані ефекти.

Такий принцип побудови цифрового зображення забезпечує можливість масштабування, зміни розміру без зниження якості, зміни форми і перегруповання примітивів для створення нових малюнків на базі вихідних об'єктів. Крім цього, такий файл займає набагато менше місця в порівнянні з растровим, в якому міститься значний обсяг відомостей, включаючи дані про кожен пікселі, його координати, колір та інші параметри [17].

3.3.3 Django Template Language

Django використовує динамічний спосіб генерації розмітки HTML. Найпоширеніший підхід ґрунтується на шаблонах. Шаблон (template) дозволяє за допомогою спеціального синтаксису визначити який блок HTML коду застосувати в тому чи іншому випадку. Це досягається конструкціями `{% if %}`,

{% else %}}, {% load static %} та {% include %}, які можна викликати безпосередньо всередині коду HTML [18].

Спочатку були визначені файли `base_header_generic.html` і `base_navigation_generic.html`, що містять інформацію про підключені файли і розмітку навігаційної панелі відповідно. Вміст цих файлів буде використано всередині інших сторінок, тому вони були винесені в окремі файли. Далі, за допомогою шаблонних команд `{% load static %}` та `{% includes %}`, написаних усередині HTML розмітки, Django дозволить уникнути повторення коду та згенерує вміст цих файлів усередині всіх інших.

Далі були створені сторінки для реєстрації та авторизації користувача, які за своєю сутністю є звичайними формами з текстовими полями (рис. 3.8 та 3.9 відповідно).

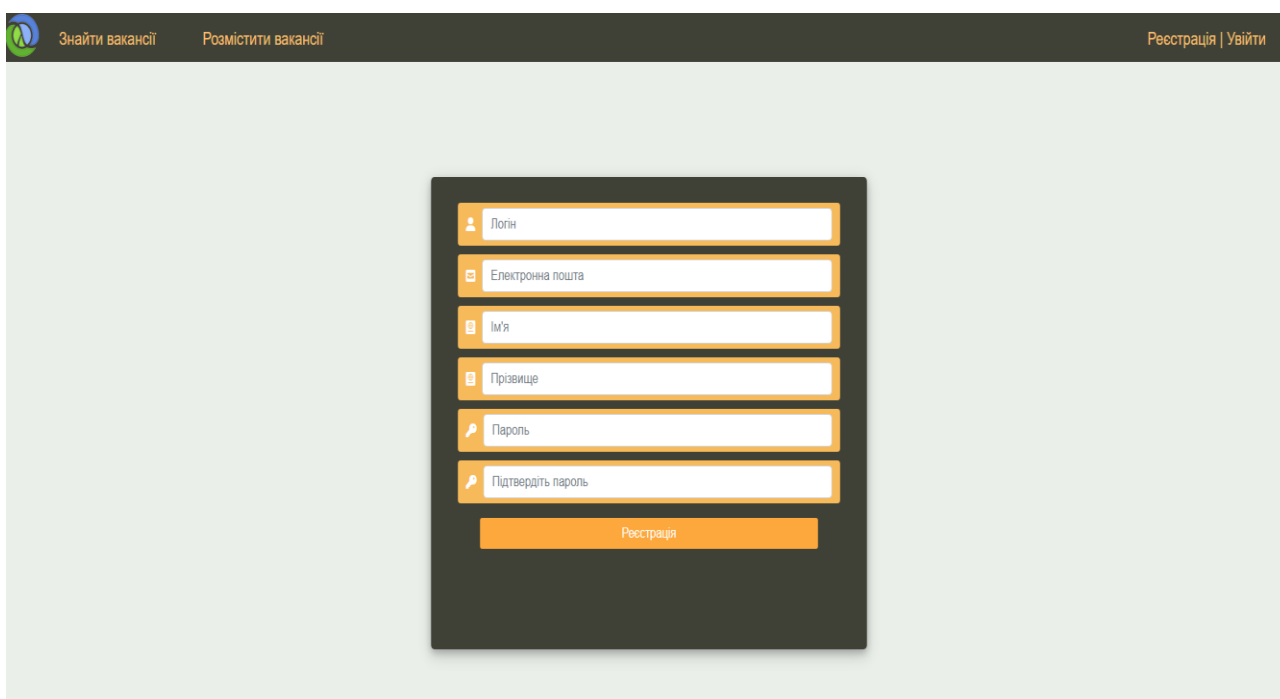


Рисунок 3.8 – Сторінка реєстрації користувача

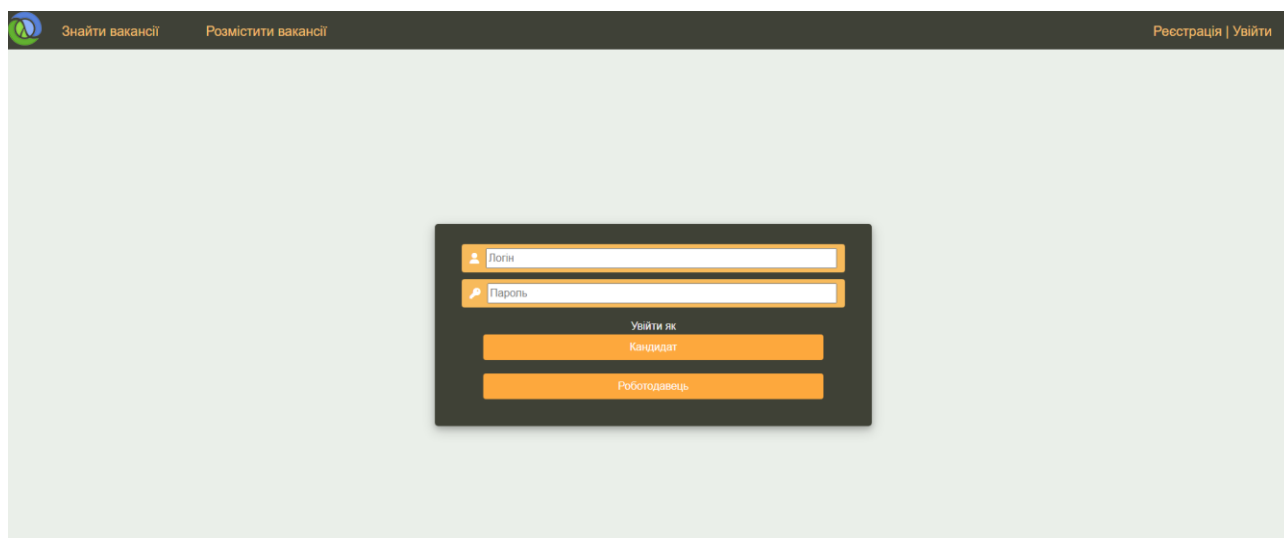


Рисунок 3.9 – Сторінка авторизації користувача

Після авторизації йде головна сторінка, яка залежить від обраної ролі на сторінці авторизації. Якщо була обрана роль «роботодавець», користувач потрапить на сторінку пошуку кандидатів (рис. 3.10).

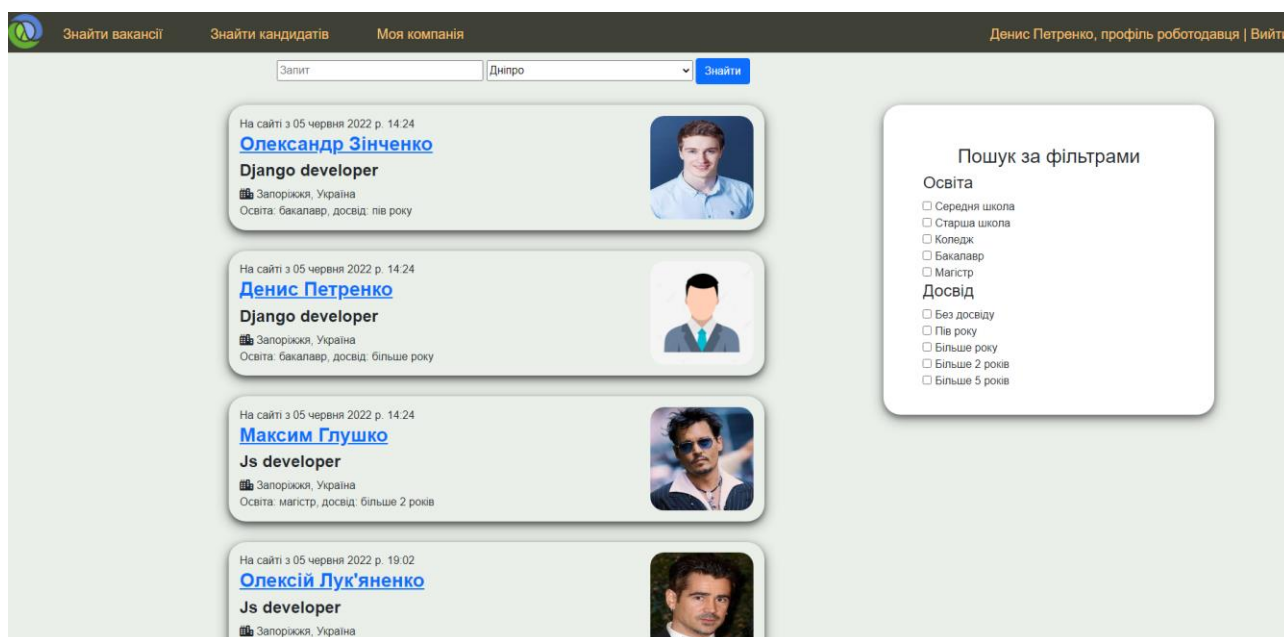


Рисунок 3.10 – Сторінка пошуку кандидатів

Навпаки, якщо при авторизації було обрано роль «кандидат», користувач потрапить на сторінку пошуку вакансій (рис. 3.11).

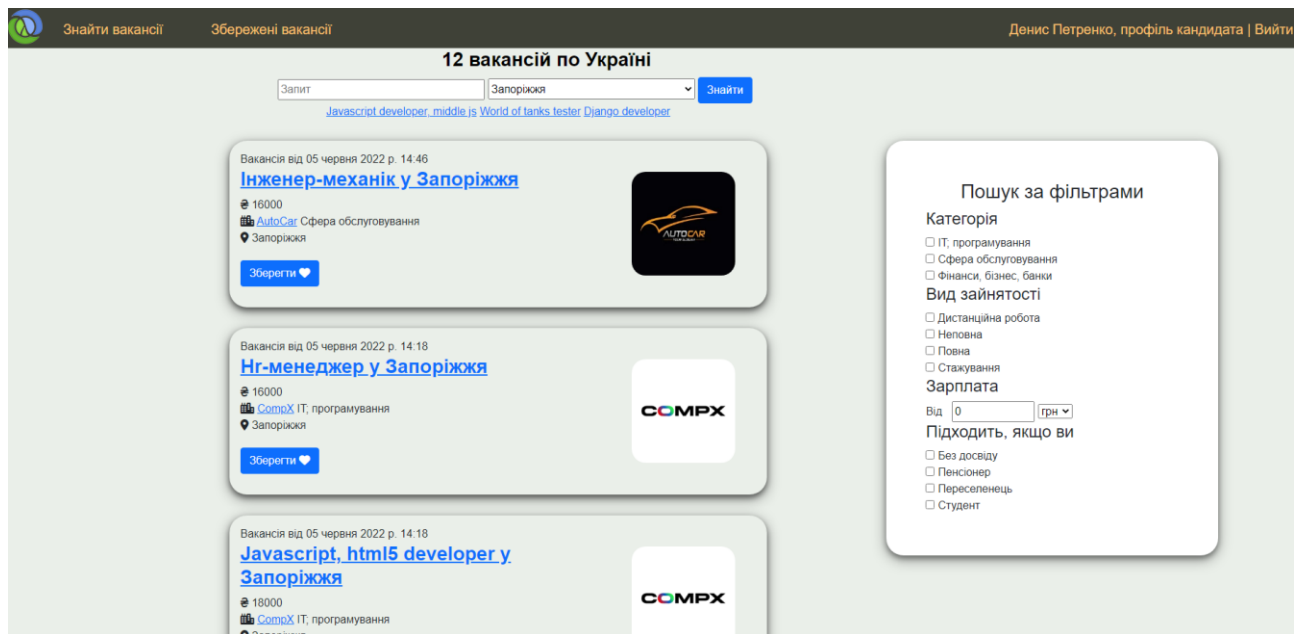


Рисунок 3.11 – Сторінка пошуку вакансій

Для створення кандидатів, вакансій, компаній та контактів компаній було створено сторінку `form.html`, яка, залежно від вказаної моделі, генерує потрібні поля HTML форми. Приклад використання файлу `form.html` для однієї з моделей наведено на рис. 3.12.

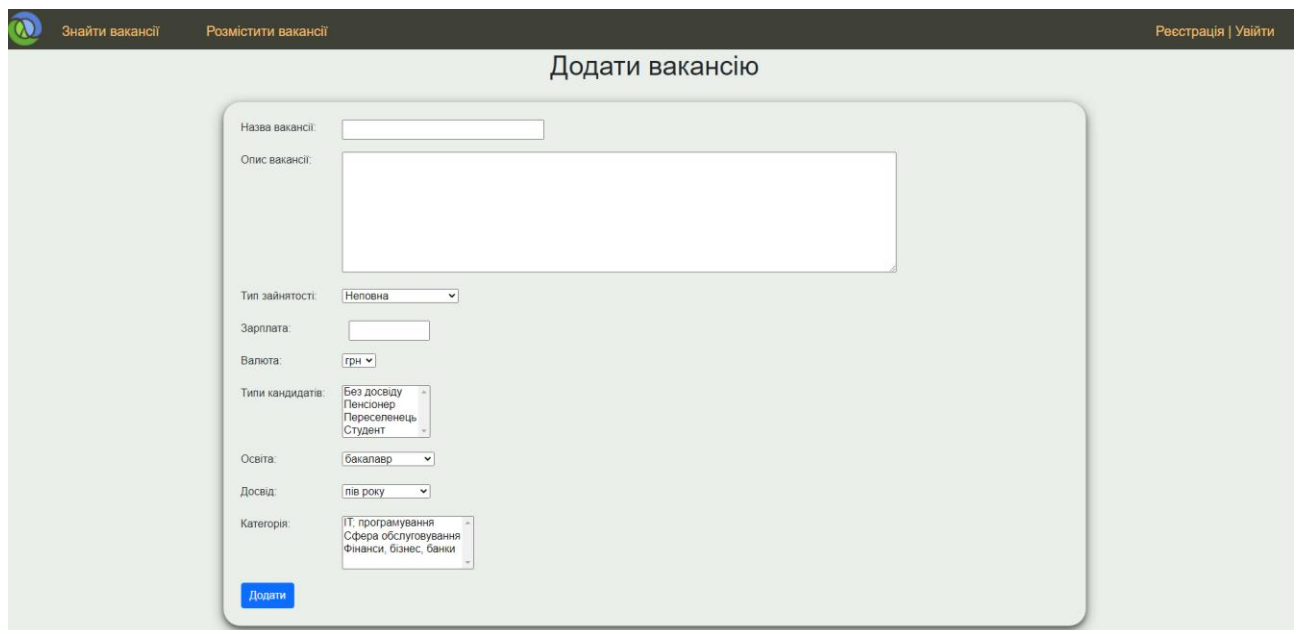


Рисунок 3.12 – Сторінка створення вакансії

Після створення відповідних сутностей доцільно було зробити сторінки їх відображення. Нижче продемонстровано результат відображення однієї з сутностей (рис. 3.13).

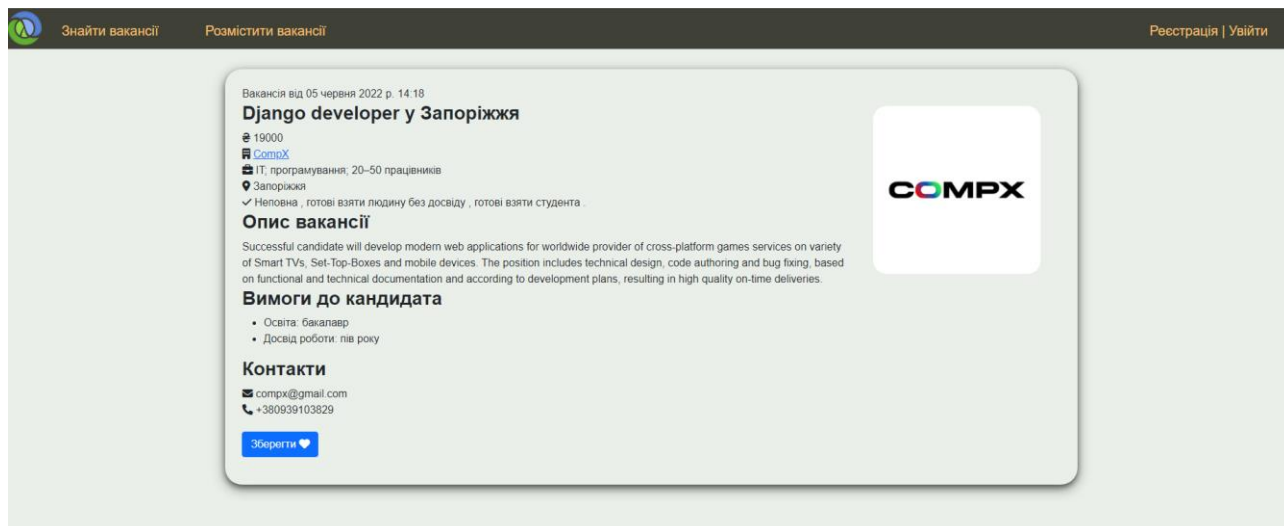


Рисунок 3.13 – Сторінка відображення вакансії

3.4 Висновки до розділу 3

В ході виконання третього розділу було здійснено пошук оптимальної мови програмування для виконання поставленого завдання. В результаті було обрано мову програмування Python, оскільки вона відповідає основним вимогам краще ніж інші мови програмування. Для обраної мови програмування було обрано середовище розробки PyCharm, оскільки воно має вбудований термінал та краще підходить для роботи саме з проектом, а не з конкретним файлом, як це роблять текстові редактори. Також розроблено основні модулі програми та описано їх функціонал з наведеними рисунками.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Призначення програми – пошук роботи та співробітників. Програмне забезпечення надає доступ до створення і пошуку вакансій та кандидатів на робоче місце.

Для повноцінної роботи з системою користувачу необхідно бути авторизованим, обравши при цьому профіль роботодавця або кандидата. Якщо користувач не має облікового запису, система запропонує створити його. Після успішної авторизації користувач потрапляє на профіль, який обрав при вході у систему.

У випадку вибору профіля роботодавця, якщо користувач входить в систему перший раз, він потрапляє на сторінку реєстрації своєї компанії. Після завершення цього етапу, він може редагувати інформацію про свою компанію, додавати, редагувати та видаляти вакансії, а також переглядати, шукати та зберігати кандидатів на робочі місця своєї компанії.

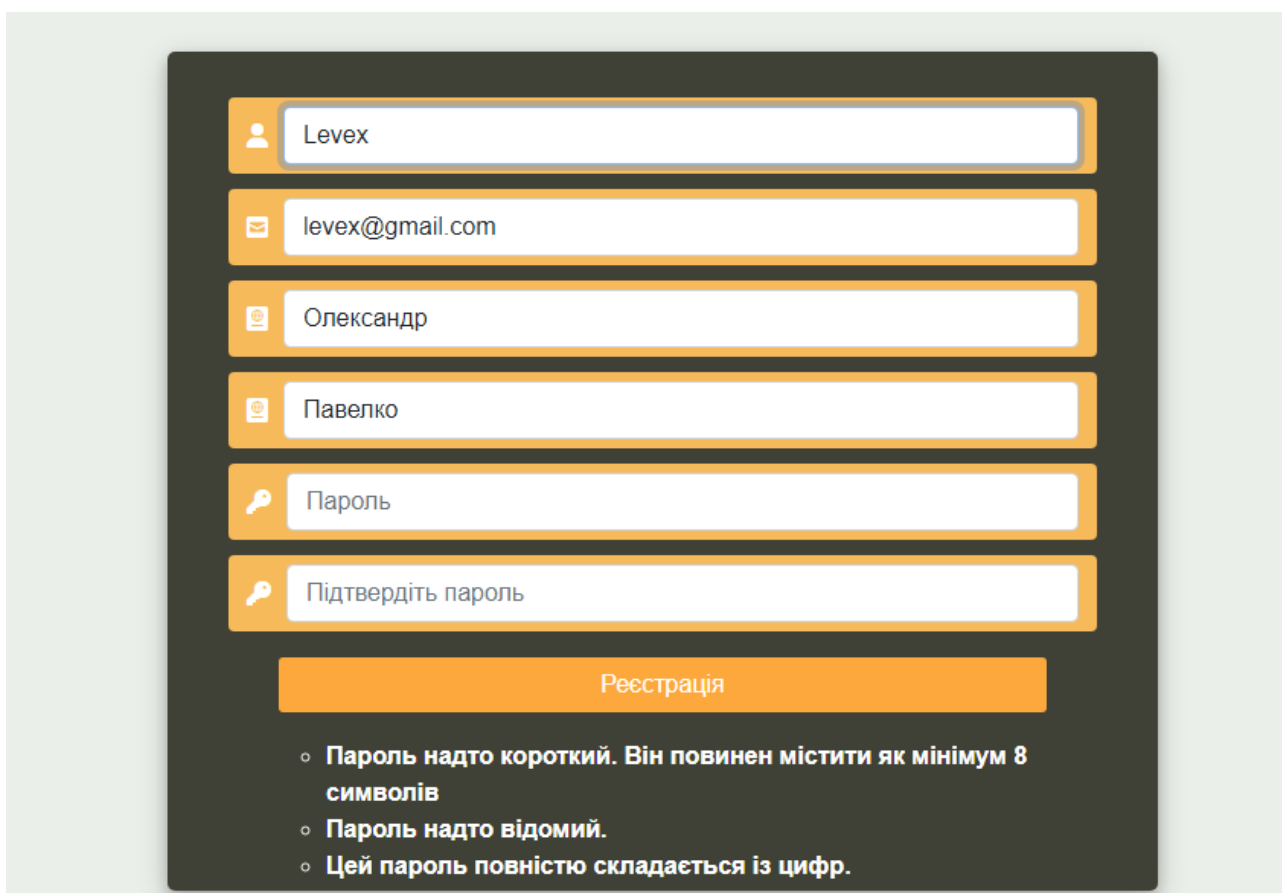
Обравши профіль кандидата, користувач потрапляє на сторінку додавання особистої інформації, такої, як рівень освіти, досвіду. Після завершення цього етапу, користувач зможе переглядати та шукати вакансії, зберігати їх собі.

Функціональні обмеження такі:

- реєстрація нового користувача;
- авторизація користувача;
- додавання, редагування, видалення компанії та контактів до неї;
- додавання, редагування, видалення, перегляд вакансій;
- пошук вакансій за запитом та фільтрами;
- збереження вакансій до списку збережених вакансій користувача;
- збереження кандидатів до списку збережених кандидатів користувача.

4.2 Тестування програми

Для розроблюваного ПЗ було проведене повноцінне тестування: перевірка кнопок, перевірка відповідності знайдених вакансій, перевірка на зрозумілість інтерфейсу, коректність ведення даних та перевірки інформації в базі даних. Перевірка відповідності знайдених вакансій, показала відповідність вакансій фільтрам, відсутність непередбачених помилок, також була перевірена інформація в базі даних. Результат перевірки без помилок. Під час перевірки придатності до використання, виявленню, що ПЗ має зручний та інтуїтивний інтерфейс, має гарно збудований інтерфейс, а саме добре підібрана кольорова гама та не складна логічна структура, тим самим покращує інтегрованість нового користувача з програмою. Були проведені тести на валідацію введеної інформації. Результати валідації зображені на рис. 4.1 – 4.3.



The image shows a registration form with the following fields and values:

- Username: Levex
- Email: levex@gmail.com
- First Name: Олександр
- Last Name: Павелко
- Password: Пароль
- Confirm Password: Підтвердіть пароль

Below the fields is an orange button labeled "Реєстрація". Below the button, there are three validation error messages:

- Пароль надто короткий. Він повинен містити як мінімум 8 символів
- Пароль надто відомий.
- Цей пароль повністю складається із цифр.

Рисунок 4.1 – Валідація реєстрації

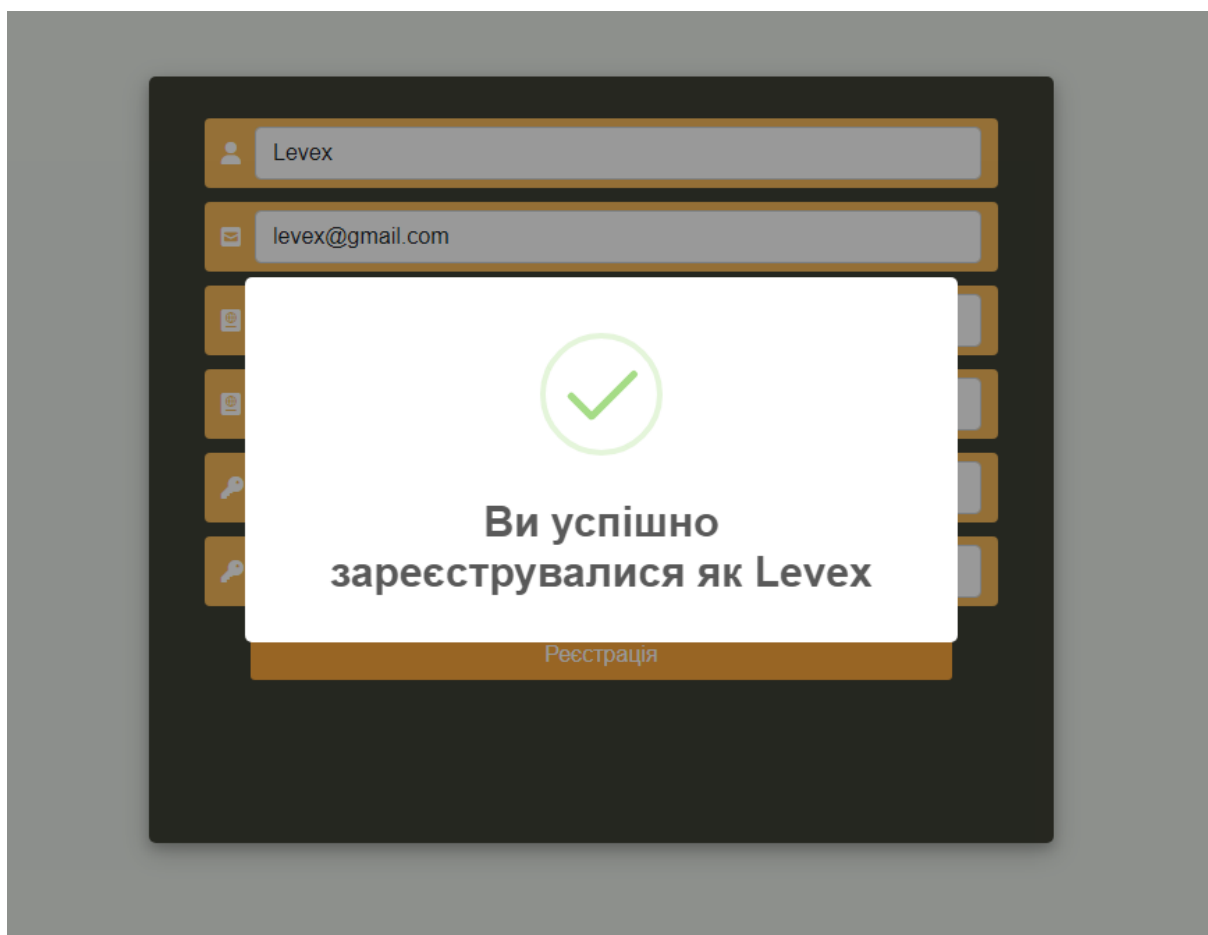


Рисунок 4.2 – Успішна реєстрація

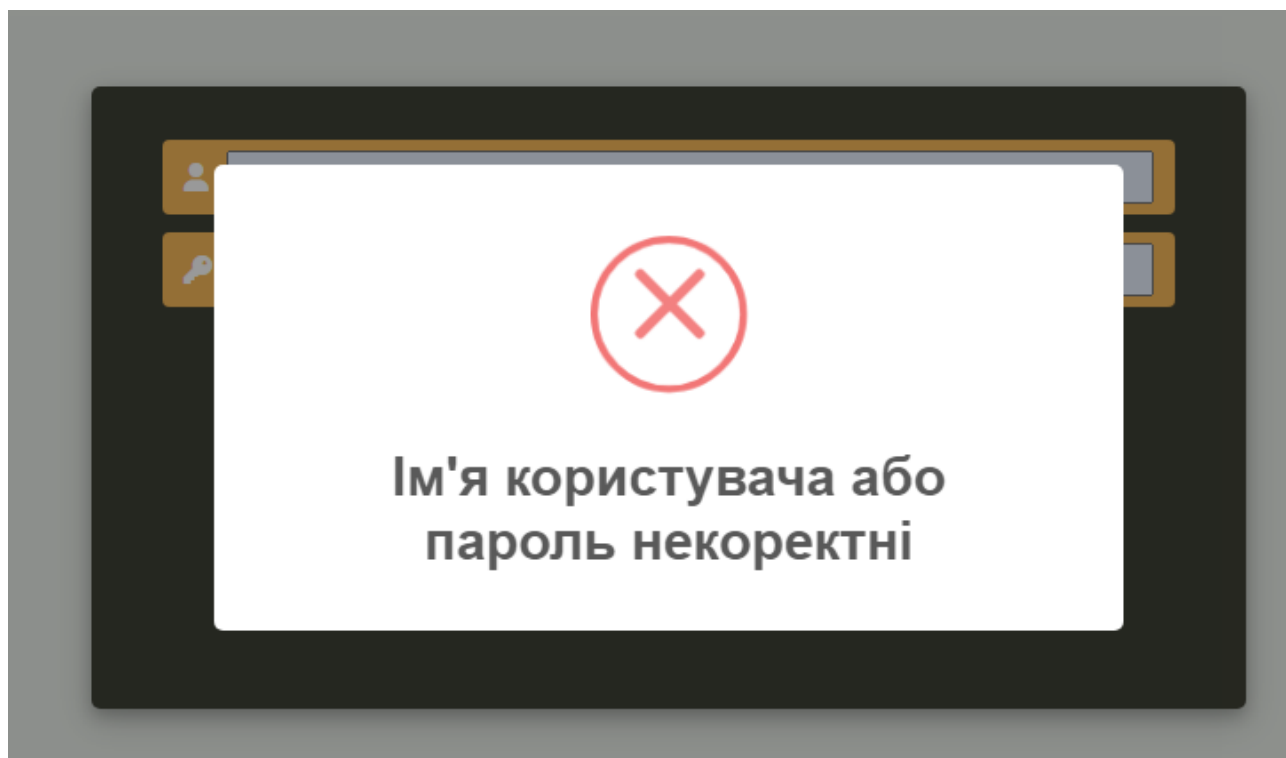


Рисунок 4.3 – Невдала авторизація

4.3 Виконання програми

Для того щоб розпочати роботу з вебзастосунком потрібно спочатку запустити його на вебсервері за відповідною встановленою адресою, за якою потім звертаються користувачі. Після введення у браузері адреси вебсервера, відкривається головка сторінка вебзастосунку (рис. 4.4).

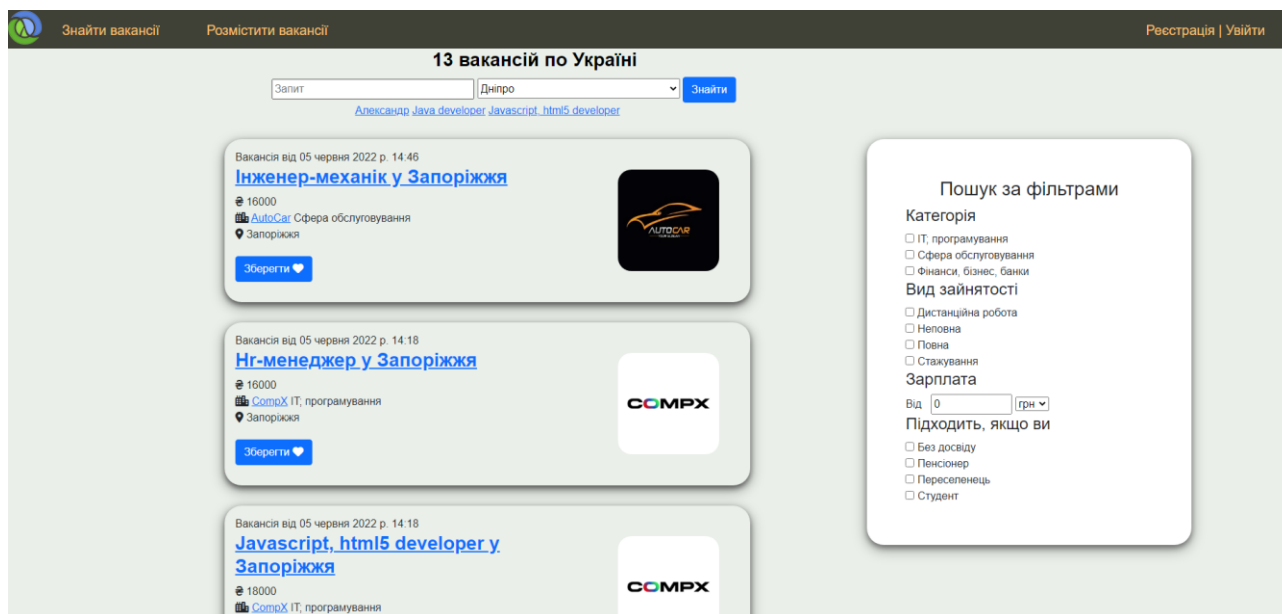


Рисунок 4.4 – Головна сторінка вебзастосунку

Вебсторінка містить чотири пункти меню, розташовані у навігаційному меню зліва направо:

- «Знайти вакансії», що представляє безпосередньо відкриту першу сторінку вебзастосунку;
- «Розмістити вакансії», що дозволяє додати власну компанію, в якій надалі можна додавати вакансії;
- «Увійти», що дозволяє виконати авторизацію облікового запису;
- «Реєстрація». Якщо користувач не має облікового запису, то він може зареєструватися в системі, для чого йому потрібно натиснути на кнопку «Реєстрація» навігаційного меню, після чого відкриється сторінка реєстрації облікового запису (рис. 4.5). Реєстрація може бути виконана тільки для

користувача, адміністративні облікові записи можуть бути створені тільки адміністратором.

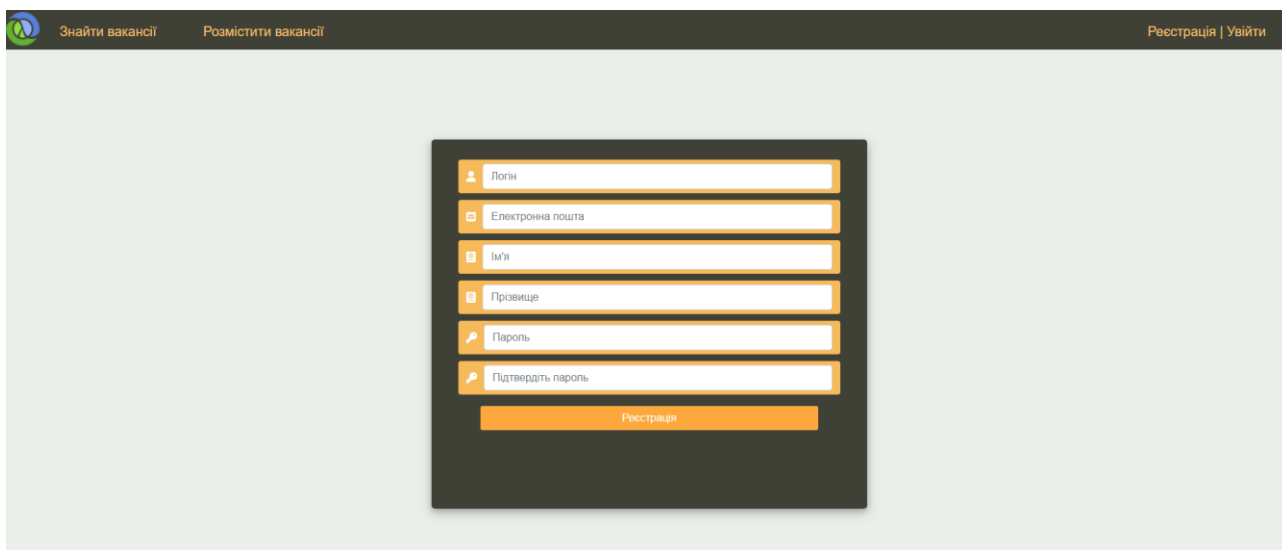


Рисунок 4.5 – Сторінка реєстрації облікового запису вебзастосунку

Якщо натиснути на кнопку «Увійти» головної сторінки, то відкриється сторінка авторизації (рис. 4.6).

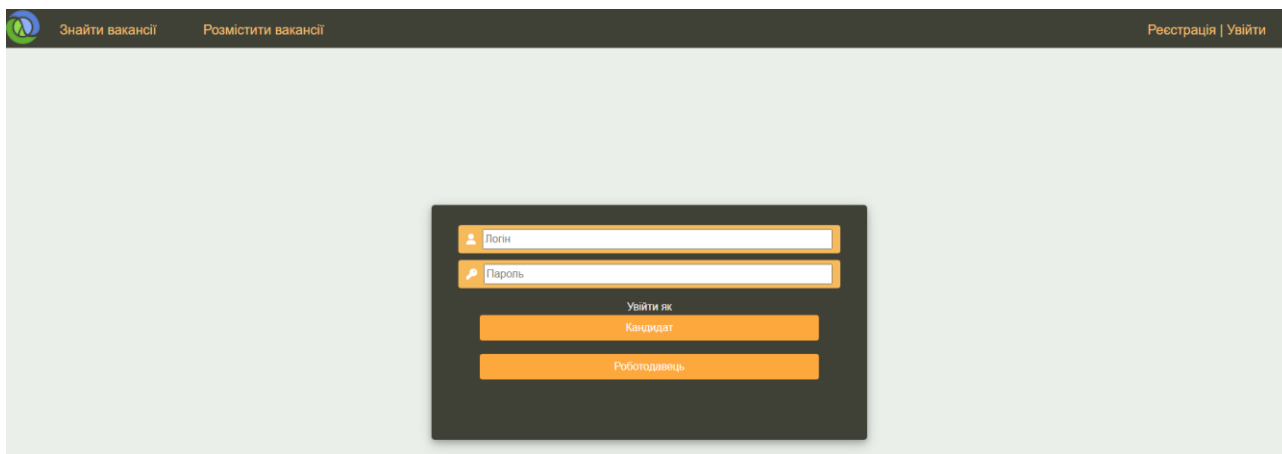


Рисунок 4.6 – Сторінка авторизації облікового запису вебзастосунку

Якщо користувач увійшов як кандидат, натиснувши відповідну кнопку на сторінці авторизації, система визначить чи вказав користувач детальну

інформацію про себе. Якщо цей етап не було виконано, користувач потрапить на сторінку з формою надання інформації про себе (рис. 4.7).

Додати дані кандидата

Номер телефону:

Освіта:

Досвід:

Ваше фото: Файл не выбран

Бажана вакансія:

Місто:

Дніпро
Запоріжжя
Київ
Львів

Рисунок 4.7 – Сторінка додавання даних кандидата

Після завершення цього етапу користувач потрапляє на сторінку пошуку вакансій (рис. 4.8).

13 вакансій по Україні

Запит

[Django.developer Senior js developer Django.developer](#)

Вакансія від 05 червня 2022 р. 14:46
Інженер-механік у Запоріжжя
16000 AutoCar Сфера обслуговування
Запоріжжя

Вакансія від 05 червня 2022 р. 14:18
ІТ-менеджер у Запоріжжя
16000 Comox ІТ, програмування
Запоріжжя

Вакансія від 05 червня 2022 р. 14:18
Javascript, html5 developer у Запоріжжя
18000 Comox ІТ, програмування
Запоріжжя

Денис Петренко, профіль кандидата | Вийти

Пошук за фільтрами

Категорія

- ☐ ІТ, програмування
- ☐ Сфера обслуговування
- ☐ Фінанси, бізнес, банки

Вид зайнятості

- ☐ Дистанційна робота
- ☐ Неповна
- ☐ Повна
- ☐ Стажування

Зарплата

Від грн

Підходить, якщо ви

- ☐ Без досвіду
- ☐ Пенсіонер
- ☐ Переселенець
- ☐ Студент

Рисунок 4.8 – Сторінка пошуку вакансій

На цій сторінці користувач може скористатися пошуком за запитом та фільтрами. Результати пошуку вакансій наведені на рис. 4.9 та рис. 4.10.

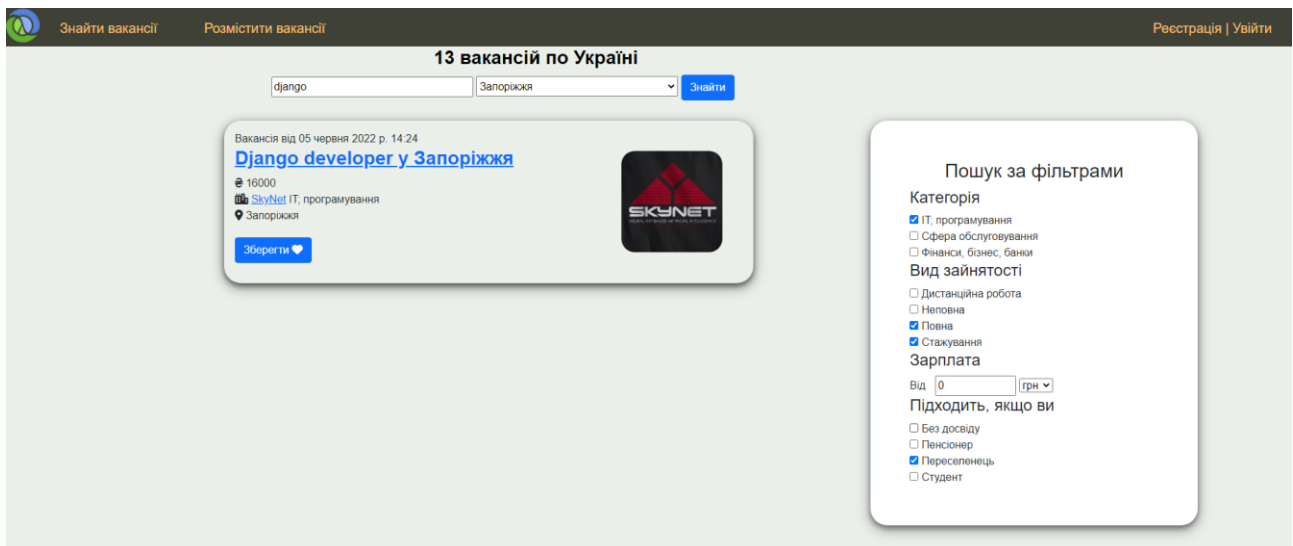


Рисунок 4.9 – Успішний пошук за фільтрами

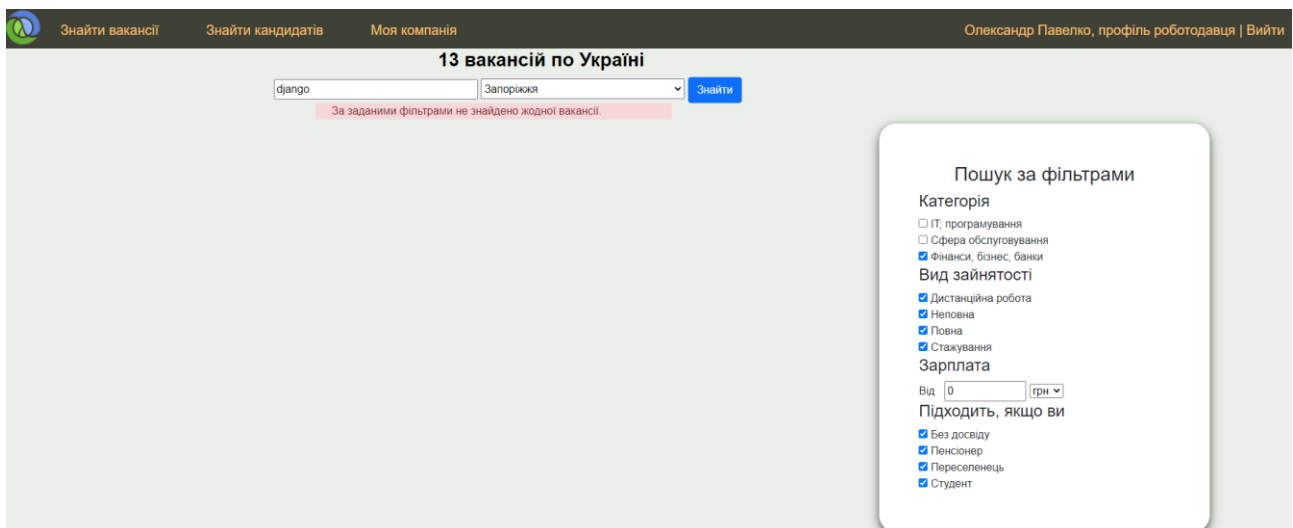


Рисунок 4.10 – Невдалий пошук за фільтрами

Переглядаючи вакансії, користувач може подивитися більш детальну інформацію про конкретну вакансію, натиснувши на посилання всередині неї (рис. 4.11). На сторінці, що відкриється, буде доступна до перегляду інформація про розташування компанії, вимоги до кандидата та контакти компанії. Також, якщо користувач авторизований через профіль кандидата, на сторінці буде доступна кнопка "Зберегти", при натисканні якої вакансія потрапить до списку збережених вакансій кандидата.

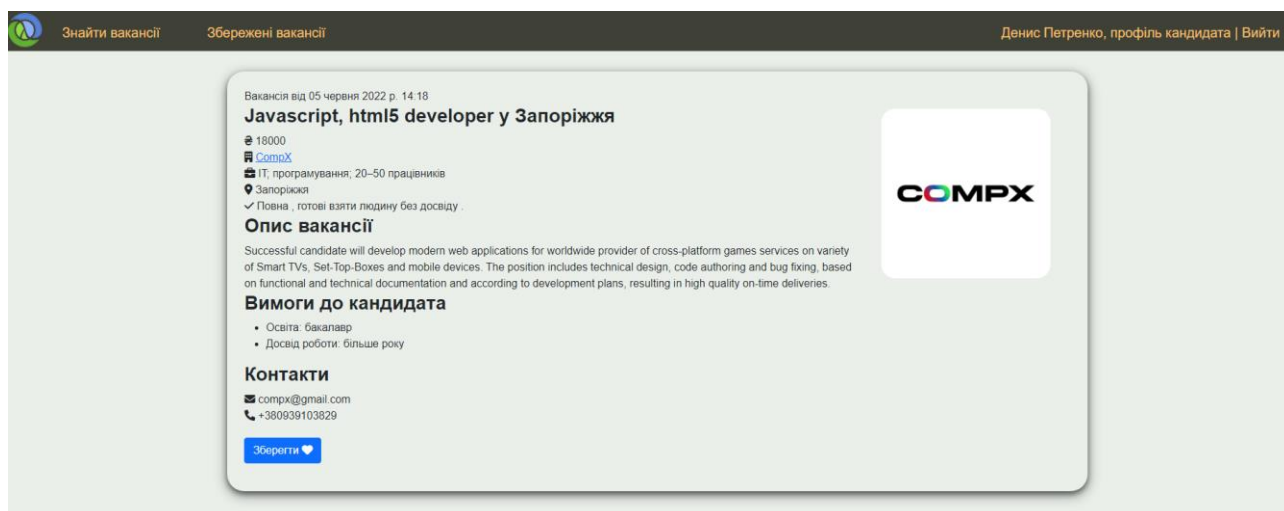


Рисунок 4.11 – Сторінка перегляду вакансії

На сторінці перегляду вакансії вказане посилання на компанію, до якої належить ця вакансія. Якщо користувач перейде за цим посиланням, він потрапить на сторінку перегляду компанії (рис. 4.12). Також на цій сторінці знаходиться кнопка «Зберегти», натиснувши яку, вакансія буде збережена до списку збережених вакансій кандидата (рис. 4.13).

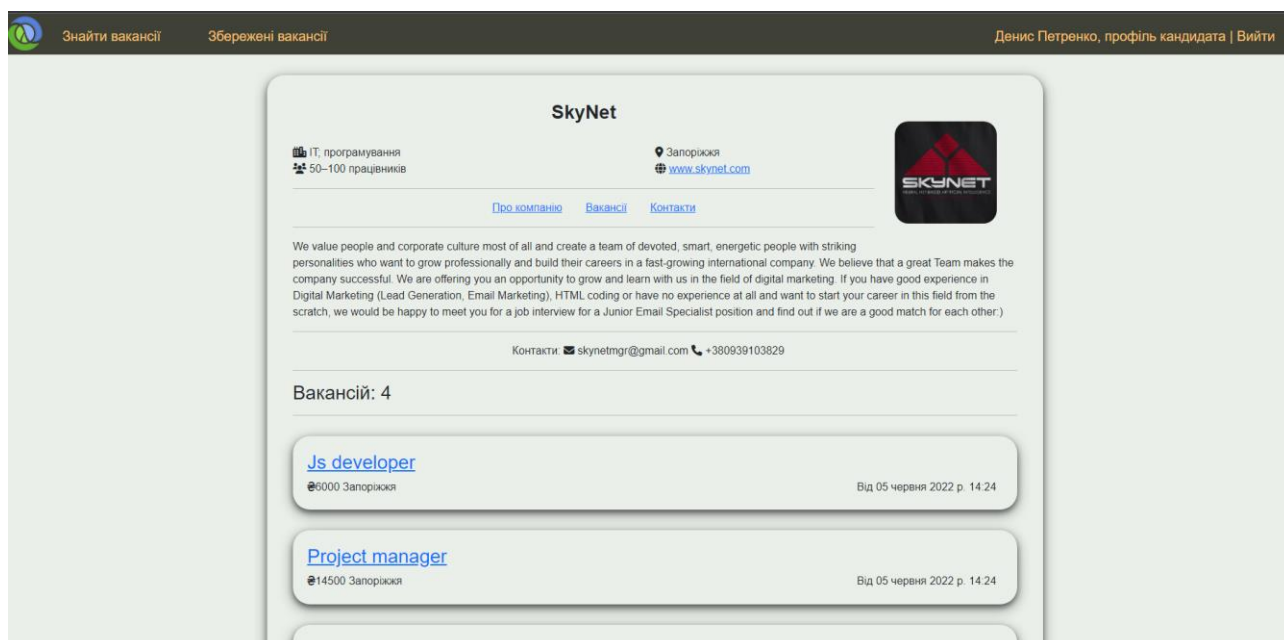


Рисунок 4.12 – Сторінка перегляду компанії

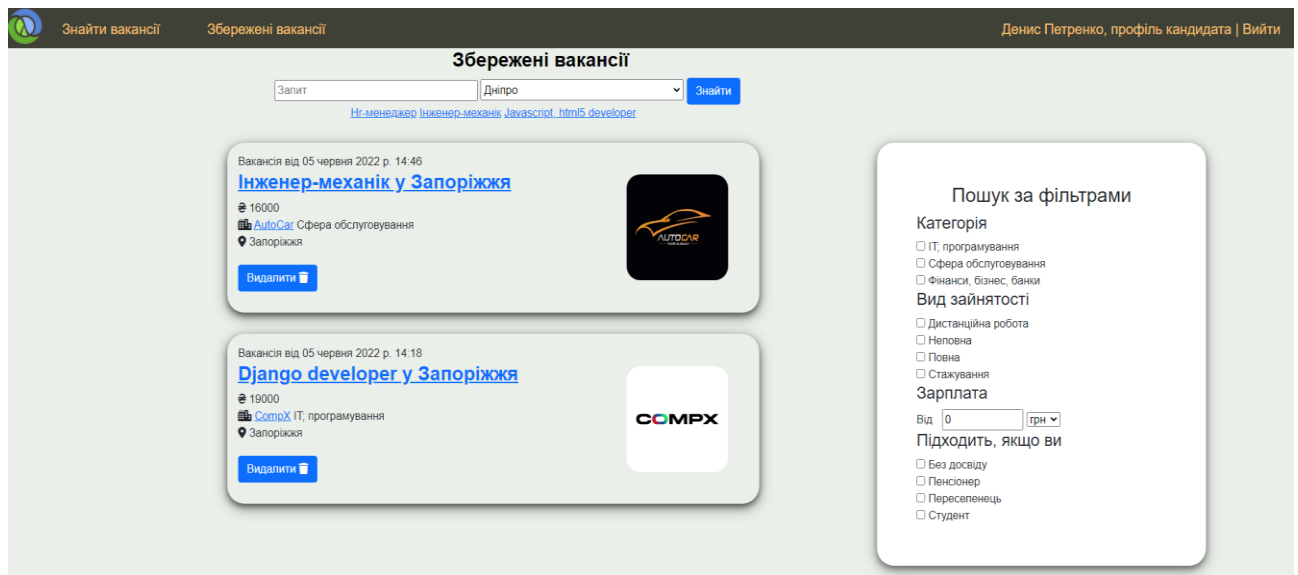


Рисунок 4.13 – Сторінка перегляду списку збережених вакансій кандидата

Якщо користувач увійшов як роботодавець, натиснувши відповідну кнопку на сторінці авторизації, система визначить чи вказав він детальну інформацію про свою компанію. Якщо цей етап не було виконано, користувач потрапить на сторінку з формою надання інформації про свою компанію (рис. 4.14).

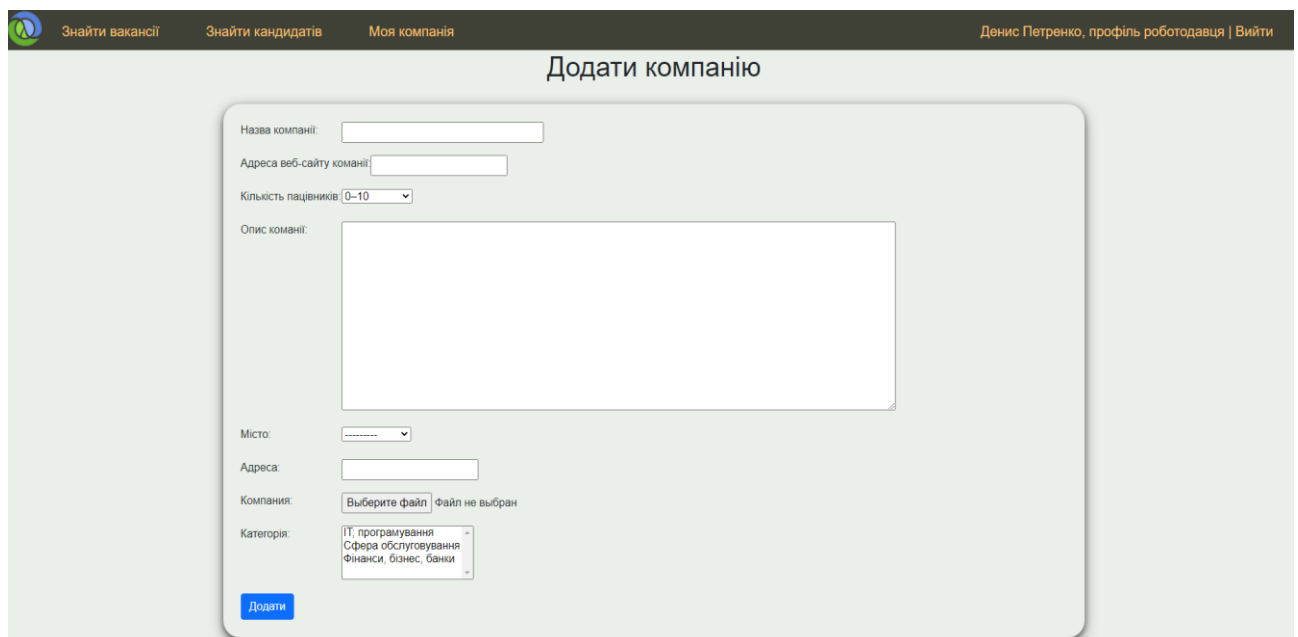


Рисунок 4.14 – Сторінка реєстрації компанії користувача

Після того, як компанію буде зареєстровано, користувачу, який авторизувався як роботодавець, буде доступний функціонал роботи з компанією, вакансіями, контактами та кандидатами (рис. 4.15 – 4.20).

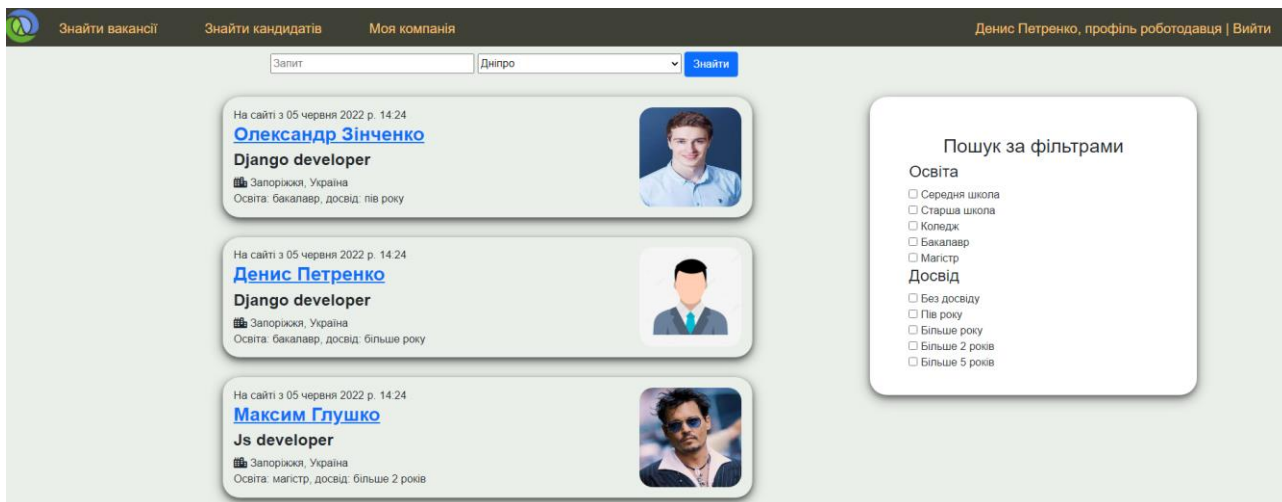


Рисунок 4.15 – Сторінка пошуку кандидатів



Рисунок 4.16 – Сторінка перегляду кандидата

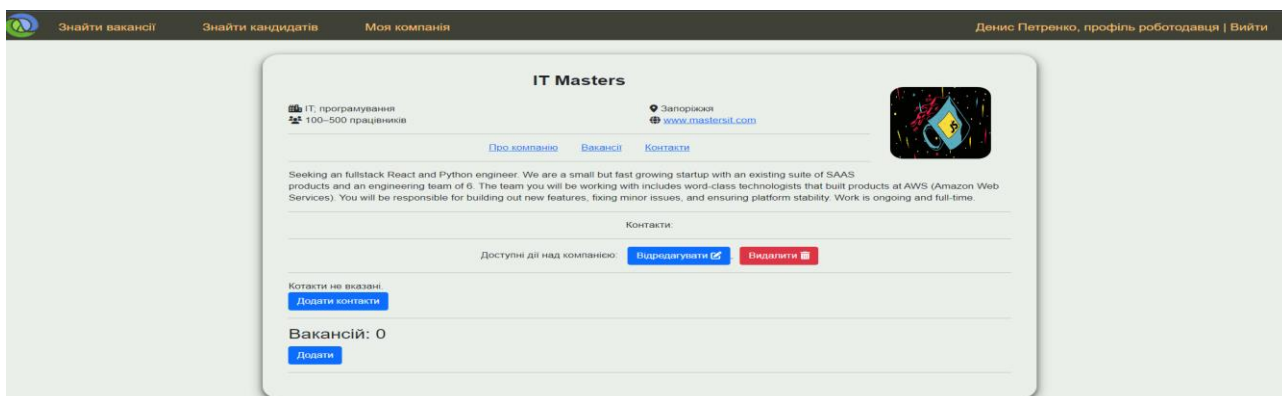
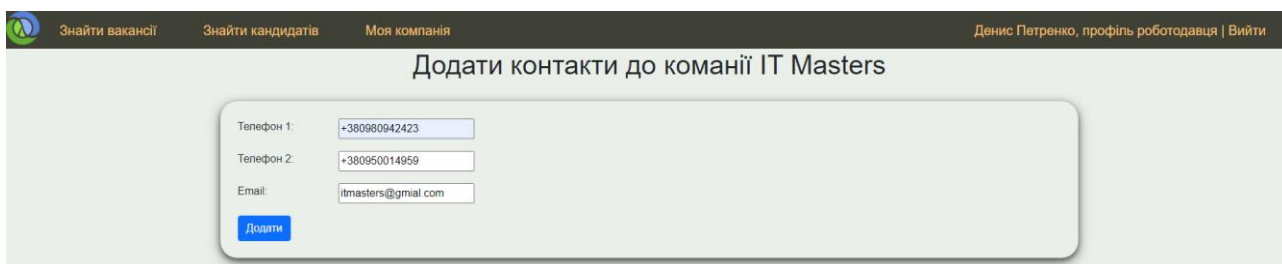


Рисунок 4.17 – Сторінка перегляду компанії користувача



Знайти вакансії Знайти кандидатів Моя компанія Денис Петренко, профіль роботодавця | Вийти

Додати контакти до компанії IT Masters

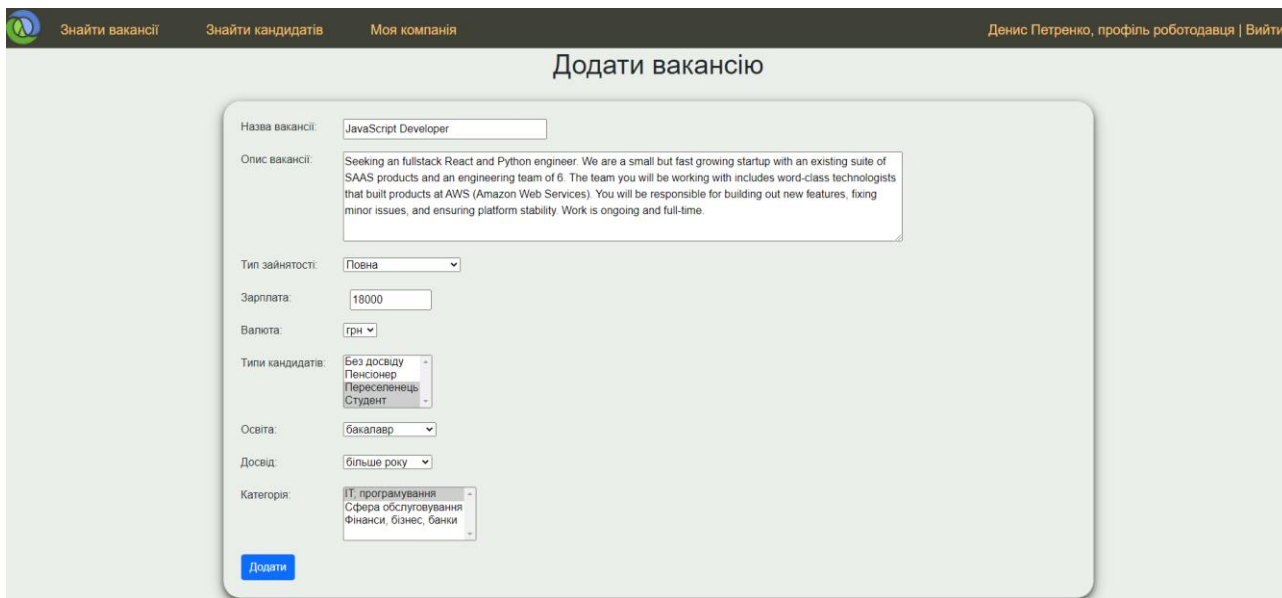
Телефон 1:

Телефон 2:

Email:

[Додати](#)

Рисунок 4.18 – Сторінка додавання контактів компанії користувача



Знайти вакансії Знайти кандидатів Моя компанія Денис Петренко, профіль роботодавця | Вийти

Додати вакансію

Назва вакансії:

Опис вакансії:

Тип зайнятості:

Зарплата:

Валюта:

Типи кандидатів:

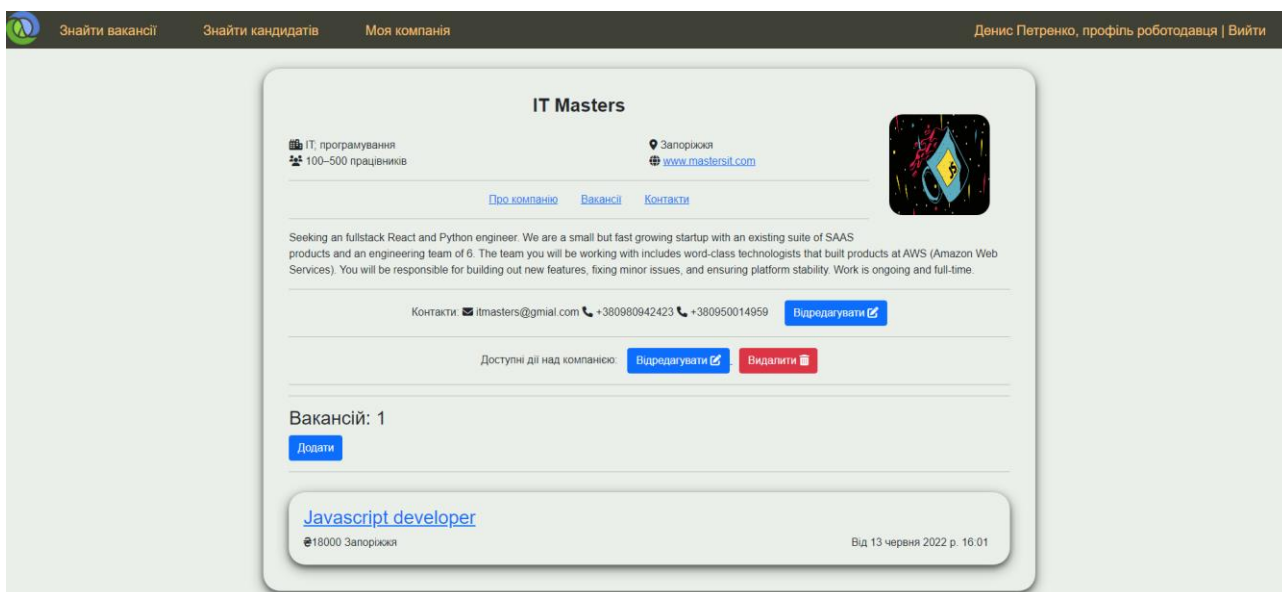
Освіта:

Досвід:

Категорія:

[Додати](#)

Рисунок 4.19 – Сторінка додавання вакансії компанії користувача



Знайти вакансії Знайти кандидатів Моя компанія Денис Петренко, профіль роботодавця | Вийти

IT Masters

IT, програмування
100-500 працівників

Запоріжжя
www.mastersit.com

[Про компанію](#) [Вакансії](#) [Контакти](#)

Seeking an fullstack React and Python engineer. We are a small but fast growing startup with an existing suite of SAAS products and an engineering team of 6. The team you will be working with includes word-class technologists that built products at AWS (Amazon Web Services). You will be responsible for building out new features, fixing minor issues, and ensuring platform stability. Work is ongoing and full-time.

Контакти: itmasters@gmail.com [+380980942423](tel:+380980942423) [+380950014959](tel:+380950014959) [Відредагувати](#)

Доступні дії над компанією: [Відредагувати](#) [Видалити](#)

Вакансій: 1
[Додати](#)

[Javascript developer](#)
18000 Запоріжжя

Від 13 червня 2022 р. 16:01

Рисунок 4.20 – Сторінка перегляду компанії користувача після додавання вакансій та контактів

Для стабільної роботи системи передбачено роботу адміністратора сайту. Адміністратору доступний функціонал перегляду, додавання, редагування та пошуку всіх сутностей (рис. 4.21 – 4.23).

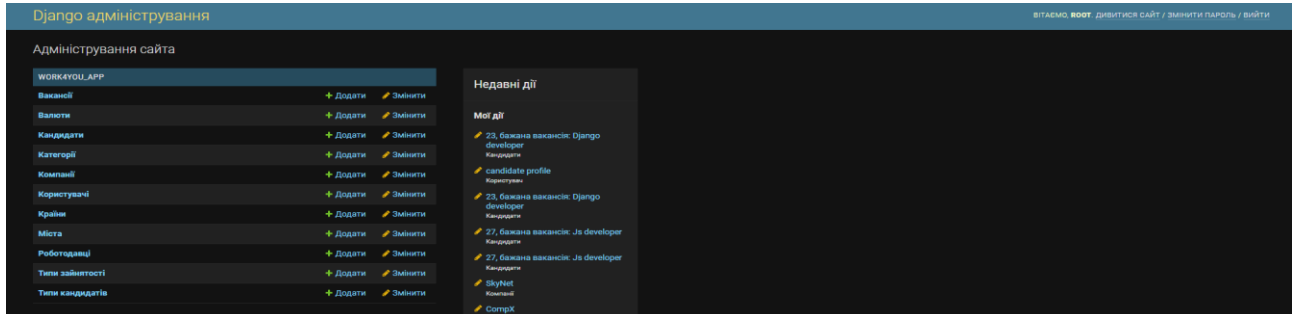


Рисунок 4.21 – Сторінка перегляду всіх сутностей в адмін-панелі

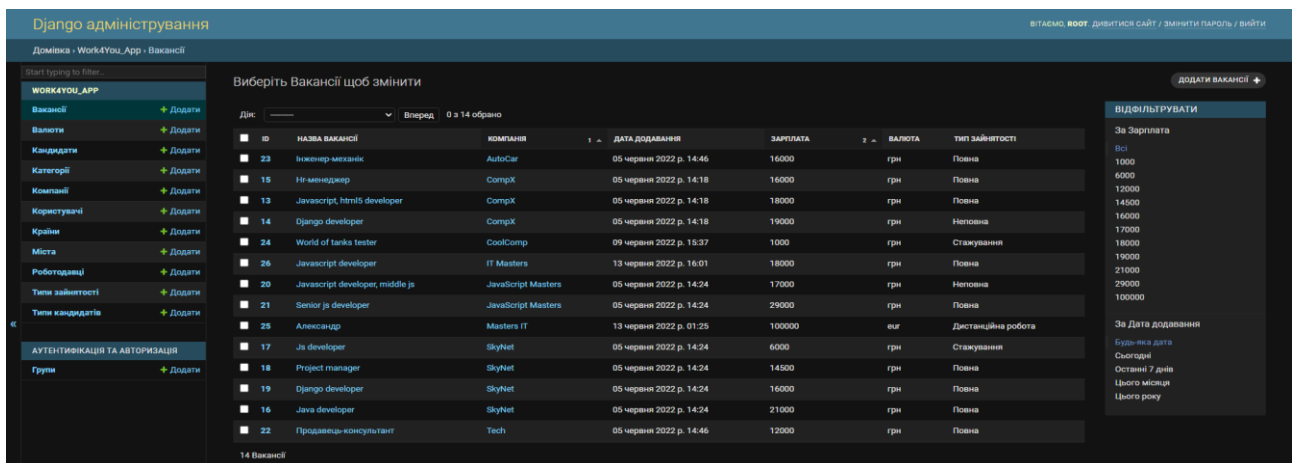


Рисунок 4.22 – Сторінка перегляду всіх вакансій в адмін-панелі

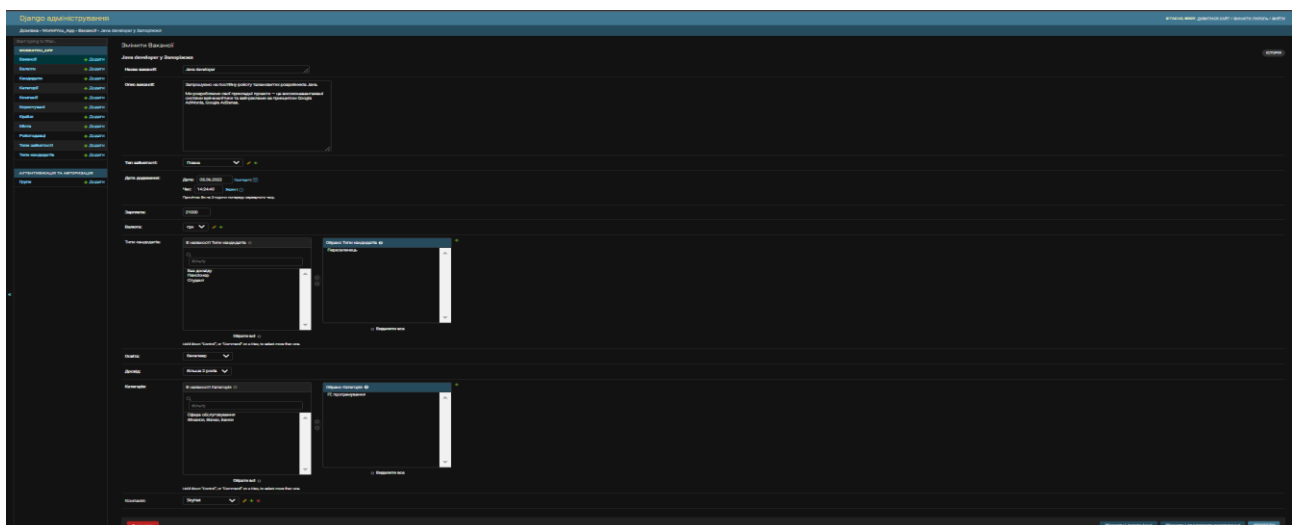


Рисунок 4.23 – Сторінка редагування однієї з вакансій в адмін-панелі

4.4 Тестування в критичних в середовищах

Для тестування було взято операційну систему Windows 7, 8 та 10, різну виділену оперативну пам'ять, певну кількість ядер на процес. Призначення критичного тестування це те, щоб отримати час, за який виконується певна дія, в нашому прикладі, це процес авторизації користувача. Порівняння вищеописаного тесту процесу наведено у табл. 4.1.

Таблиця 4.1 – Тест процесу авторизації

	Windows 7	Windows 8	Windows 10
Оперативна пам'ять 256 мб Процесор 1 ядра	2.4 секунд	3.1 секунд	3.1 секунд
Оперативна пам'ять 512 мб Процесор 1 ядра	2 секунд	2.4 секунд	2.4 секунд
Оперативна пам'ять 256 мб Процесор 2 ядра	2.1 секунд	2.5 секунд	2.7 секунд
Оперативна пам'ять 512 мб Процесор 2 ядра	1.9 секунд	2.3 секунд	2.0 секунд

4.5 Висновки до розділу 4

Описано основні методики для проведення повноцінного тестування програми пошуку роботи "Work4You". Під час проведення тестування програми виявлено, що програма інтуїтивна та має зручний інтерфейс, сайт зручний для користувача, має зручну та логічну структуру. Також були проведенні тестування в екстремальних середовищах, результатом цих тестувань є те, що

програма може працювати і в поганих умовах, але буде гальмувати, а це не відповідає вимогам швидкого аналізу. Після проведених тестувань можна зробити висновок, що загалом ПЗ працює коректно та відповідає поставленим до нього вимогам.

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи було проведено аналіз предметної області та порівняння з існуючими подібними застосунками. Були визначені переваги та недоліки найпопулярніших сервісів для пошуку роботи. На підставі порівнянь різних систем сформульовані функціональні вимоги. У вигляді діаграм були представлені вимоги до ПЗ.

Для реалізації ПЗ була обрана мова програмування Python, фреймворк Django та СУБД SQLite. Середовищем розробки програми було обрано PyCharm. У розробленому програмному забезпеченні реалізовано створення вакансій, їх пошук за запитом та фільтрами. Також була реалізована авторизація користувачів з можливістю вибору профілю. В залежності від вибору доступний відповідний функціонал.

ПЗ було протестовано та перевірено на відображення коректних повідомлень про помилку. Завдання, яке було поставлено до дипломної роботи, було виконано до кінця.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Work.ua: сайт для пошуку роботи [Електронний ресурс]. – Режим доступу: <https://www.work.ua/>
2. Rabota.ua [Електронний ресурс]. – Режим доступу : <https://rabota.ua/>
3. Olx.ua [Електронний ресурс]. – Режим доступу : <https://www.olx.ua/>
4. Порівняння фреймворків [Електронний ресурс]. – Режим доступу : <https://stackshare.io/stackups/django-vs-spring-boot>
5. Архітектура MTV в Django [Електронний ресурс]. – Режим доступу : <https://djbook.ru/ch05s02.html>
6. Введення в Django Django [Електронний ресурс]. – Режим доступу : <https://metanit.com/python/django/1.1.php>
7. Мова Python та її переваги [Електронний ресурс]. – Режим доступу : <https://brainskills.ru/blog/plyusy-i-minusy-python/>
8. Реляційна СУБД SQLite [Електронний ресурс]. – Режим доступу : <https://blog.skillfactory.ru/glossary/sqlite/>
9. Архітектура MTV [Електронний ресурс]. – Режим доступу : <https://towardsdatascience.com/working-structure-of-django-mtv-architecture-a741c8c64082?gi=2ab9b057d291>
10. Різниця між текстовими редакторами та IDE [Електронний ресурс]. – Режим доступу : <https://habr.com/ru/post/306048/>
11. PyCharm IDE та його переваги [Електронний ресурс]. – Режим доступу : <https://habr.com/ru/post/122018/>
12. Текстовий редактор Visual Studio Code [Електронний ресурс]. – Режим доступу : <https://open.zeba.academy/pochemu-vscode-populyaren/>
13. Sublime Text 3, його переваги та недоліки [Електронний ресурс]. – Режим доступу : <https://bizzapps.ru/p/vs-code/>
14. Структура проекту Django [Електронний ресурс]. – Режим доступу : <https://kvckr.me/igi/21.html>
15. Bootstrap: переваги та недоліки [Електронний ресурс]. – Режим доступу : <https://timeweb.com/ru/community/articles/plyusy-i-minusy-bootstrap-1>

16. Векторні іконки FontAwesome [Електронний ресурс]. – Режим доступу : <https://fontawesome.ru/>

17. Векторна графіка: що це таке? [Електронний ресурс]. – Режим доступу : <https://www.informcad.ru/blog/vektornaya-grafika-cto-eto-takoe>

18. Шаблони в Django [Електронний ресурс]. – Режим доступу : <https://django.fun/docs/django/ru/4.0/topics/templates/>

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення для пошуку роботи та співробітників може використовуватися для малого та великого бізнесу, підприємств, які хочуть знайти співробітників.

A.1 Підстави для розробки

Підставою для розробки програмного забезпечення є завдання на дипломну кваліфікаційну роботу бакалавра на тему: “Розробка програмного забезпечення для пошуку роботи”, затверджене наказом № 102 від 27.04.2022 р. по Національному університету “Запорізька політехніка”.

A.2 Призначення розробки

Функціональне призначення: програма дозволяє створювати нові вакансії для користувачів, які шукають роботу.

A.3 Вимоги до програми чи програмному виробу

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати:

- реєстрацію нового користувача;
- авторизацію в програмі;
- роботу адміністратора;
- створення компанії користувача;
- додавання вакансій до компанії, їх перегляд, редагування та видалення, пошук за фільтрами;
- додавання кандидатів до системи, їх перегляд, редагування та видалення, пошук за фільтрами.

А.3.2 Вимоги до надійності

Програмне забезпечення повинно забезпечувати оброблення наступних ситуацій:

- не вдалося авторизувати обліковий запис;
- задано некоректні значення під час реєстрації облікового запису.

А.3.3 Умови експлуатації

Для забезпечення експлуатації програмного забезпечення необхідно встановити наступні програмні бібліотеки:

- Django 3;
- Python версії 3.10;
- браузер Google Chrome, Mozilla Firefox, Opera;
- вебсервер WSGI або Apache.

За відсутності даних бібліотек повноцінне функціонування програмного забезпечення неможливе.

А.3.4 Вимоги до складу та параметрів технічних засобів

Для експлуатації програмного комплексу необхідні наступні технічні засоби:

- процесор – не менше 300 МГц;
- обсяг оперативного запам'ятовуючого пристрою – не менше 2 Гб для повноцінної роботи;
- Svga-монітор і відеокарта з роздільною здатністю не менше 1024x768;
- обсяг вільного простору на жорсткому диску – не менше 1 Гб;
- підключення до мережі Інтернет.

A.3.5 Вимоги до маркування і упакування

Програма може бути записана на оптичному, жорсткому магнітному диску або USB-носії. На упаковці або відповідному пристрої має бути вказано назву роботи «Програмне забезпечення для пошуку роботи».

A.3.6 Вимоги до транспортування і збереження

Вимоги до транспортування та зберігання повністю ідентичні вимогам, пропонованим до носія, на якому зберігається програма, що включає: оптичний, жорсткий магнітний диск або USB-носії.

A.4 Стадії та етапи розробки

Для роботи над дипломною кваліфікаційною роботою бакалавра заплановано виконання наступних етапів:

- аналіз проблеми;
- проєктування програмного забезпечення;
- розробка програмного забезпечення;
- тестування і відлагодження програмного забезпечення;
- оформлення документації.

A.5 Порядок контролю та приймання

Дипломна кваліфікаційна робота бакалавра повинна бути виконана у відповідності з календарним планом. Дипломна кваліфікаційна робота бакалавра має бути прийнята консультантами відповідних розділів та керівником роботи загалом, після чого затверджена завідуючим кафедрою та захищена на засіданні екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Файл models.py

```
import datetime
from django.contrib.auth.models import AbstractUser
from django.db import models

class BaseModel(models.Model):
    objects = models.Manager()

    class Meta:
        abstract = True

class Country(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=100, verbose_name="Країна")

    class Meta:
        verbose_name = "Країни"
        verbose_name_plural = "Країни"
        ordering = ['title', 'id']

    def __str__(self):
        return self.title

class City(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=100, verbose_name="Місто")
    country = models.ForeignKey(to=Country, on_delete=models.CASCADE,
verbose_name="Країна")

    class Meta:
        verbose_name = "Міста"
        verbose_name_plural = "Міста"
        ordering = ['title', 'country']

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

```

class Category(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=100, verbose_name="Назва категорії")

    class Meta:
        verbose_name = "Категорії"
        verbose_name_plural = "Категорії"
        ordering = ['title', 'id']

    def __str__(self):
        return self.title


class Currency(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=100, verbose_name="Валюта")

    class Meta:
        verbose_name = "Валюти"
        verbose_name_plural = "Валюти"
        ordering = ['-title', 'id']

    def __str__(self):
        return self.title


class CandidateType(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=100, verbose_name="Тип кандидата")

    class Meta:
        verbose_name = "Типи кандидатів"
        verbose_name_plural = "Типи кандидатів"
        ordering = ['title', 'id']

    def __str__(self):
        return self.title


class EmploymentType(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=100, verbose_name="Тип зайнятості")

```

```

class Meta:
    verbose_name = "Типи зайнятості"
    verbose_name_plural = "Типи зайнятості"
    ordering = ['title', 'id']

    def __str__(self):
        return self.title


class Contacts(models.Model):
    id = models.AutoField(primary_key=True)
    phone1 = models.CharField(max_length=20, verbose_name="Телефон 1",
null=True, blank=False)
    phone2 = models.CharField(max_length=20, verbose_name="Телефон 2",
null=True, blank=True)
    email = models.CharField(max_length=30, verbose_name="Email", null=True,
blank=False)


class Company(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=100, verbose_name="Назва компанії")
    website_url = models.CharField(max_length=100, verbose_name="Адреса вебсайту
компанії")
    wrkrs_cnt_choices = [('0-10', '0-10'), ('20-50', '20-50'), ('50-100', '50-
100'), ('100-500', '100-500'),
                        ('500-1000+', '500-1000+')]
    workers_count = models.CharField(max_length=20, default="10-20",
verbose_name="Кількість пацієнтів",
                                choices=wrkrs_cnt_choices)
    description = models.TextField(max_length=1500, verbose_name="Опис компанії")
    city = models.ForeignKey(to=City, on_delete=models.CASCADE,
verbose_name="Місто")
    address = models.CharField(max_length=100, verbose_name="Адреса")
    image = models.ImageField(verbose_name="Компанія",
upload_to="photos/%Y/%m/%d/", null=True, blank=True)
    categories = models.ManyToManyField(to=Category, verbose_name="Категорія",
null=True, blank=True)
    contacts = models.OneToOneField(to=Contacts, on_delete=models.CASCADE,
null=True, blank=True, verbose_name="Контакти")

```

```

class Meta:
    verbose_name = "Компанії"
    verbose_name_plural = "Компанії"
    ordering = ['city', 'title']

def __str__(self):
    return self.title

class Vacancy(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=100, verbose_name="Назва вакансії")
    description = models.TextField(max_length=1500, verbose_name="Опис
вакансії")
    employment_type = models.ForeignKey(to=EmploymentType,
on_delete=models.CASCADE, max_length=100,
verbose_name="Тип зайнятості")
    creation_date = models.DateTimeField(default=datetime.datetime.now(),
verbose_name="Дата додавання")
    salary = models.PositiveIntegerField(verbose_name="Зарплата")
    currency = models.ForeignKey(to=Currency, on_delete=models.CASCADE,
max_length=15, verbose_name="Валюта", default="грн")
    candidate_types = models.ManyToManyField(to=CandidateType,
verbose_name="Типи кандидатів", null=True, blank=True)
    edu_lvls = [("середня школа", "середня школа"), ("старша школа", "старша
школа"), ("коледж", "коледж"),
                ("бакалавр", "бакалавр"), ("магістр", "магістр")]
    education_level = models.CharField(max_length=50, choices=edu_lvls,
verbose_name="Освіта")
    exp_lvls = [("без досвіду", "без досвіду"), ("пів року", "пів року"),
("більше року", "більше року"),
                ("більше 2 років", "більше 2 років"), ("більше 5 років", "більше
5 років")]
    experience_level = models.CharField(max_length=50, choices=exp_lvls,
verbose_name="Досвід")
    categories = models.ManyToManyField(to=Category, verbose_name="Категорія")
    company = models.ForeignKey(to=Company, null=True, blank=True,
on_delete=models.PROTECT, verbose_name="Компанія")

    class Meta:
        verbose_name = "Вакансії"
        verbose_name_plural = "Вакансії"

```

```

        ordering = ['company', 'salary']

    def __str__(self):
        return f'{self.title} y {self.company.city}'

    def save(self, *args, **kwargs):
        self.title = self.title.lower().capitalize()
        super().save(*args, **kwargs)

    def get_currency_sign(self):
        if str(self.currency) == 'грн':
            return "fa-solid fa-hryvnia-sign"
        elif str(self.currency) == 'usd':
            return "fa-solid fa-usd"
        elif str(self.currency) == 'eur':
            return "fa-solid fa-eur"

class Candidate(models.Model):
    id = models.AutoField(primary_key=True)
    phone = models.CharField(max_length=30, null=True, verbose_name="Номер телефону")
    creation_date = models.DateTimeField(default=datetime.datetime.now(), verbose_name="Дата реєстрації")

    edu_lvls = [("середня школа", "середня школа"), ("старша школа", "старша школа"), ("коледж", "коледж"),
                ("бакалавр", "бакалавр"), ("магістр", "магістр")]
    education_level = models.CharField(max_length=50, choices=edu_lvls, verbose_name="Освіта")
    exp_lvls = [("без досвіду", "без досвіду"), ("пів року", "пів року"), ("більше року", "більше року"),
                ("більше 2 років", "більше 2 років"), ("більше 5 років", "більше 5 років")]
    experience_level = models.CharField(max_length=50, choices=exp_lvls, verbose_name="Досвід")
    image = models.ImageField(verbose_name="Ваше фото", upload_to="photos/%Y/%m/%d/", null=True, blank=True)
    saved_vacancies = models.ManyToManyField(to=Vacancy, null=True, verbose_name="Збережені вакансії", blank=True)
    desired_vacancy = models.CharField(max_length=100, null=True, blank=True, verbose_name="Бажана вакансія")

```

```

city = models.ForeignKey(to=City, on_delete=models.CASCADE,
verbose_name="Місто", null=True, blank=True)

class Meta:
    verbose_name = "Кандидати"
    verbose_name_plural = "Кандидати"
    ordering = ['creation_date', 'id']

def __str__(self):
    return f'{self.id}, бажана вакансія: {self.desired_vacancy}'

class Employer(models.Model):
    id = models.AutoField(primary_key=True)
    creation_date = models.DateTimeField(default=datetime.datetime.now(),
verbose_name="Дата реєстрації")
    saved_candidates = models.ManyToManyField(to=Candidate, null=True,
verbose_name="Збережені кандидати", blank=True)
    company = models.ForeignKey(to=Company, on_delete=models.CASCADE, null=True,
verbose_name="Ваша компанія",
                                blank=True)

    class Meta:
        verbose_name = "Роботодавці"
        verbose_name_plural = "Роботодавці"
        ordering = ['creation_date', 'id']

    def __str__(self):
        return f'{self.id}, {self.company.title}'

class User(AbstractUser):
    is_candidate = models.BooleanField(default=False, verbose_name="Кандидат")
    is_employer = models.BooleanField(default=False,
verbose_name="Роботодавець")
    candidate = models.OneToOneField(to=Candidate, on_delete=models.CASCADE,
null=True, blank=True)
    employer = models.OneToOneField(to=Employer, on_delete=models.CASCADE,
null=True, blank=True)

    def __str__(self):
        if self.is_candidate:

```

```

        return f'candidate profile'
    else:
        return f'employer profile'

```

Б.2 Файл forms.py

```

from django import forms
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from .models import *

```

```

class CandidateForm(forms.ModelForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

    class Meta:
        model = Candidate
        fields = '__all__'
        exclude = ['user', 'creation_date', 'saved_vacancies']
        widgets = {
            'title': forms.TextInput(attrs={'class': 'form-input'}),
            'description': forms.Textarea(attrs={'cols': 100, 'rows': 20})
        }

```

```

class VacancyForm(forms.ModelForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

    class Meta:
        model = Vacancy
        fields = '__all__'
        exclude = ['creation_date', 'company']
        widgets = {
            'title': forms.TextInput(attrs={'class': 'form-input'}),
            'description': forms.Textarea(attrs={'cols': 100, 'rows': 20})
        }

```

```

class CompanyForm(forms.ModelForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

```

```

class Meta:
    model = Company
    fields = '__all__'
    exclude = ['contacts']
    widgets = {
        'title': forms.TextInput(attrs={'class': 'form-input'}),
        'description': forms.Textarea(attrs={'cols': 100, 'rows': 20})
    }

class ContactsForm(forms.ModelForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

    class Meta:
        model = Contacts
        fields = '__all__'
        widgets = {
            'title': forms.TextInput(attrs={'class': 'form-input'}),
            'description': forms.Textarea(attrs={'cols': 100, 'rows': 20})
        }

class RegisterUserForm(UserCreationForm):
    username = forms.CharField(label='Логін',
    widget=forms.TextInput(attrs={'class': 'form-input'}))
    first_name = forms.CharField(label='Ім\''я',
    widget=forms.TextInput(attrs={'class': 'form-input'}))
    last_name = forms.CharField(label='Прізвище',
    widget=forms.TextInput(attrs={'class': 'form-input'}))
    email = forms.EmailField(label='Email',
    widget=forms.EmailInput(attrs={'class': 'form-input'}))
    password1 = forms.CharField(label='Пароль',
    widget=forms.PasswordInput(attrs={'class': 'form-input'}))
    password2 = forms.CharField(label='Повтор пароля',
    widget=forms.PasswordInput(attrs={'class': 'form-input'}))

    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email', 'password1',
        'password2')

```

```
class LoginUserForm(AuthenticationForm):
    username = forms.CharField(label='Логін',
widget=forms.TextInput(attrs={'class': 'form-input'}))
    password = forms.CharField(label='Пароль',
widget=forms.PasswordInput(attrs={'class': 'form-input'}))
```

Б.3 Файл apps.py

```
from django.apps import AppConfig

class Work4YouAppConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'work4you_app'
```

Б.4 Файл admin.py

```
from django.contrib import admin
from django.forms import Textarea
from .models import *

formfield_overrides = {
    models.CharField: {'widget': Textarea(attrs={'rows': 1, 'cols': 50})},
    models.TextField: {'widget': Textarea(attrs={'rows': 15, 'cols': 60})},
}

class VacancyAdmin(admin.ModelAdmin):
    list_display = ('id', 'title', 'company', 'creation_date', 'salary',
'currency', 'employment_type')
    list_display_links = ('id', 'title', 'company')
    list_filter = ('salary', 'creation_date')
    formfield_overrides = formfield_overrides
    filter_horizontal = ('categories', 'candidate_types')

class CompanyAdmin(admin.ModelAdmin):
    list_display = ('id', 'title', 'website_url', 'description',
```

```

'workers_count', 'city', 'address')
    list_display_links = ('title', 'website_url', 'address')
    formfield_overrides = formfield_overrides
    filter_horizontal = ('categories',)

class CategoryAdmin(admin.ModelAdmin):
    list_display = ('id', 'title')
    list_display_links = ('id', 'title',)
    formfield_overrides = formfield_overrides

admin.site.register(User)
admin.site.register(Country)
admin.site.register(City)
admin.site.register(Currency)
admin.site.register(CandidateType)
admin.site.register(EmploymentType)
admin.site.register(Category, CategoryAdmin)
admin.site.register(Vacancy, VacancyAdmin)
admin.site.register(Company, CompanyAdmin)
admin.site.register(Candidate)
admin.site.register(Employer)

```

Б.5 Файл views.py

```

import random
import sweetify as sweetify
from django.contrib import messages
from django.contrib.auth import authenticate, login, logout
from django.shortcuts import render, redirect
from work4you_app.forms import *
from work4you_app.models import *

def get_candidates(request):
    page = {
        'candidates': User.objects.all(),
        'exp_lvls': map(lambda lvl: str(lvl[0]).capitalize(),
Candidate.exp_lvls),
        'edu_lvls': map(lambda lvl: str(lvl[0]).capitalize(),

```

```

Candidate.edu_lvls),
    'cities': City.objects.all()
}
return render(request, 'work4you_app/user/candidates.html', page)

def get_vacancies(request):
    count = Vacancy.objects.count()
    top3 = random.sample(list(Vacancy.objects.all()), 3)
    page = {
        'title': f'{count} вакансій по Україні',
        'vacancies': Vacancy.objects.all(),
        'top3': top3,
        'categories': Category.objects.all(),
        'employment_types': EmploymentType.objects.all(),
        'currencies': Currency.objects.all(),
        'candidate_types': CandidateType.objects.all(),
        'cities': City.objects.all(),
    }
    return render(request, "work4you_app/vacancy/vacancies.html", page)

def get_vacancy(request, vacancy_id):
    vacancy = Vacancy.objects.get(id=vacancy_id)
    page = {
        'vacancy_details': vacancy,
        'title': vacancy
    }
    return render(request, "work4you_app/vacancy/vacancy.html", page)

def find_vacancies(request):
    query = request.GET.get('query').lower()
    city = City.objects.get(title=request.GET.get('city'))
    cities = [city]
    salary = request.GET.get('salary')
    employment_types = list(map(lambda t: EmploymentType.objects.get(title=t),
request.GET.getlist('employment_types')))
    candidate_types = list(map(lambda ct: CandidateType.objects.get(title=ct),
request.GET.getlist('candidate_types')))
    categories = list(map(lambda c: Category.objects.get(title=c),
request.GET.getlist('categories')))

```

```

currency = Currency.objects.get(title=request.GET.get('user_currency'))
if city.title == 'Вся країна':
    cities = City.objects.all()
if not employment_types:
    employment_types = EmploymentType.objects.all()
if not categories:
    categories = Category.objects.all()
if not candidate_types:
    candidate_types = CandidateType.objects.all()
vacancies = Vacancy.objects.filter(
    categories__in=categories, salary__gte=salary, currency=currency,
    employment_type__in=employment_types,
candidate_types__in=candidate_types,
    company__city__in=cities, title__contains=query).distinct()
count = Vacancy.objects.count()
page = {
    'title': f'{count} вакансій по Україні',
    'vacancies': vacancies,
    'categories': Category.objects.all(),
    'employment_types': EmploymentType.objects.all(),
    'currencies': Currency.objects.all(),
    'candidate_types': CandidateType.objects.all(),
    'cities': City.objects.all(),
    'selected_query': query,
    'selected_city': city,
    'selected_salary': salary,
    'selected_employment_types': employment_types,
    'selected_candidate_types': candidate_types,
    'selected_categories': categories,
    'selected_currency': currency
}
return render(request, "work4you_app/vacancy/vacancies.html", page)

def get_company(request, company_id):
    page = {
        'title': Company.objects.get(id=company_id).title,
        'company': Company.objects.get(id=company_id),
        'vacancies':
Vacancy.objects.filter(company=Company.objects.get(id=company_id))
    }
    return render(request, "work4you_app/company/company.html", page)

```

```

def get_companies(request):
    page = {
        'companies': Company.objects.all()
    }
    return render(request, "work4you_app/company/company.html", page)

def get_company_vacancies(request, company_id):
    page = {
        'vacancies':
Vacancy.objects.filter(company=Company.objects.get(id=company_id))
    }
    return render(request, "work4you_app/company/company.html", page)

def get_candidate(request, candidate_id):
    user = User.objects.get(candidate_id=candidate_id)
    page = {
        'user': user
    }
    return render(request, 'work4you_app/user/candidate.html', page)

def get_saved_vacancies(request, candidate_id):
    candidate = Candidate.objects.get(id=candidate_id)
    vacancies = candidate.saved_vacancies.all()
    top3 = sorted(Vacancy.objects.all()[:3], key=lambda x: random.random())
    page = {
        'title': 'Збережені вакансії',
        'vacancies': vacancies,
        'top3': top3,
        'categories': Category.objects.all(),
        'employment_types': EmploymentType.objects.all(),
        'currencies': Currency.objects.all(),
        'candidate_types': CandidateType.objects.all(),
        'cities': City.objects.all(),
        'candidate': candidate
    }
    return render(request, 'work4you_app/vacancy/vacancies.html', page)

```

```

def save_vacancy(request, candidate_id, vacancy_id):
    candidate = Candidate.objects.get(id=candidate_id)
    if request.method == 'GET':
        vacancy = Vacancy.objects.get(id=vacancy_id)
        candidate.saved_vacancies.add(vacancy)
        candidate.save()

    return redirect(request.META.get('HTTP_REFERER'))

def add_candidate(request):
    candidate_form = CandidateForm()
    if request.method == 'POST':
        candidate_form = CandidateForm(request.POST)
        if candidate_form.is_valid():
            candidate = candidate_form.save()
            user = request.user
            user.candidate = candidate
            user.candidate.image = request.FILES['image']
            candidate.save()
            user.save()

            return redirect(f'/candidate/{candidate.id}')
    page = {
        'title': 'Додати данні кандидата',
        'form': candidate_form,
    }
    return render(request, "work4you_app/form.html", page)

def add_company(request):
    company_form = CompanyForm()
    if request.method == 'POST':
        company_form = CompanyForm(request.POST)
        if company_form.is_valid():
            company = company_form.save()
            company.save()
            user = request.user
            user.employer.company = company
            user.employer.company.image = request.FILES['image']
            user.employer.company.save()

```

```

        user.employer.save()
        user.save()
        messages.success(request, 'Ви успішно зареєстрували компанію ' +
company.title)
        return redirect(f'/company/{company.id}')

```

```

page = {
    'title': 'Додати компанію',
    'form': company_form,
}
return render(request, "work4you_app/form.html", page)

```

```

def add_company_vacancy(request, company_id):
    vacancy_form = VacancyForm()
    if request.method == 'POST':
        vacancy_form = VacancyForm(request.POST)
        if vacancy_form.is_valid():
            vacancy = vacancy_form.save()
            vacancy.company = Company.objects.get(id=company_id)
            vacancy.save()
            return redirect(f'/company/{company_id}')
    page = {
        'title': 'Додати вакансію',
        'form': vacancy_form
    }
    return render(request, "work4you_app/form.html", page)

```

```

def add_contacts(request, company_id):
    contacts_form = ContactsForm()
    company = Company.objects.get(id=company_id)
    if request.method == 'POST':
        contacts_form = ContactsForm(request.POST)
        if contacts_form.is_valid():
            contacts = contacts_form.save()
            contacts.save()
            company.contacts = contacts
            company.save()
            return redirect(f'/company/{company_id}')
    page = {
        'title': f'Додати контакти до компанії {company.title}',

```

```

        'form': contacts_form
    }
    return render(request, "work4you_app/form.html", page)

def registerPage(request):
    sweetify.DEFAULT_OPTS = {
        'showConfirmButton': True,
        'timer': 0,
        'allowOutsideClick': True,
        'confirmButtonText': 'OK',
    }

    form = RegisterUserForm()
    if request.method == 'POST':
        form = RegisterUserForm(request.POST)
        if form.is_valid():
            user = form.save()
            username = form.cleaned_data.get('username')
            candidate = Candidate.objects.create(user=user)
            candidate.save()
            employer = Employer.objects.create(user=user)
            employer.save()
            user.candidate = candidate
            user.employer = employer
            user.save()
            sweetify.success(request, f'Ви успішно зареєструвалися як {username}')

    page = {'form': form}
    return render(request, 'work4you_app/user/auth/register.html', page)

def loginPage(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            if request.POST.get('candidate') is not None:
                user.is_candidate = True

```

```

        user.is_employer = False
        user.save()
        if user.candidate.desired_vacancy is None:
            return redirect('/candidate/add')
        return redirect('/vacancies')
    else:
        user.is_employer = True
        user.is_candidate = False
        user.save()
        if user.employer.company is None:
            return redirect('/company/add')
        return redirect(f'/company/{user.employer.company.id}')
    else:
        sweetify.error(request, 'Ім'я користувача або пароль некоректні')
page = {}
return render(request, 'work4you_app/user/auth/login.html', page)

```

```

def logoutUser(request):
    logout(request)
    return redirect('login')

```

Б.6 Файл urls.py

```

from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path

from work4you import settings
from work4you_app.views import *

urlpatterns = [
    path('admin/', admin.site.urls),

    path('register/', registerPage, name="register"),
    path('login/', loginPage, name="login"),
    path('logout/', logoutUser, name="logout"),

    path('', get_vacancies),
    path('vacancies', get_vacancies),
    path('vacancy/<int:vacancy_id>', get_vacancy),

```

```

    path('vacancies/company/<int:company_id>', get_company_vacancies),
    path('vacancies/find', find_vacancies),

    path('company/<int:company_id>', get_company),
    path('company/add', add_company),
    path('company/<int:company_id>/vacancy/add', add_company_vacancy),

    path('candidates', get_candidates),
    path('candidate/<int:candidate_id>', get_candidate),
    path('candidate/add', add_candidate),
    path('candidate/<int:candidate_id>/vacancy/<int:vacancy_id>/save',
save_vacancy),
    path('candidate/<int:candidate_id>/vacanices/saved', get_saved_vacancies),

    path('company/<int:company_id>/contacts/add', add_contacts),

] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

Б.7 Файл style.css

```

* {
    font-family: "Google Sans", arial, sans-serif;
}

body {
    background-color: #eae9e9;
}

label {
    font-weight: normal;
}

input[type='number'] {
    width: 120px;
    margin-left: 10px;
}

.header {

```

```
        background: #3F4137;
        height: 60px;
    }

    .logo {
        margin-top: 4px;
    }

    ul.mainmenu {
        text-align: center;
        list-style: none;
        margin: 0;
        padding: 0;
        height: 60px;
        color: #fdc073;
        font-size: 20px;
        overflow: hidden;
    }

    ul.mainmenu li {
        float: left;

        margin: 18px 40px 0 30px;
    }

    ul.mainmenu li.logo {
        margin: 0;
    }

    ul.mainmenu li.last {
        float: right;
    }

    ul.mainmenu li a {
        color: #fdc073;
        text-decoration: none;
    }

    ul.mainmenu li a:hover {
        color: #FDA83D;
    }
```

```
.panelitems {
    text-align: center;
}

ul.langitem {
    list-style: none;
    display: inline-block;
    margin: 30px;
    padding: 0;
    max-width: 300px
}

ul.langitem li {
    margin: 10px 0 0 0;
    width: 100%;
}

ul.langitem li.image {
    margin: 0;
    text-align: center;
    width: 100%;
    height: 250px;
    overflow: hidden;
}

ul.langitem li.descr {
    color: #777;
    height: 170px;
    overflow: hidden;
}

ul.langitem li a {
    color: #fdc073;
    text-decoration: none;
    font-size: 20px;
}

ul.langitem li a:hover {
    color: #FDA83D;
}

.vacancy {
```

```

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.5), 0 6px 20px 0 rgba(0, 0, 0,
0.39);
    border-radius: 25px;
    padding: 2%;
    overflow: auto;
    text-overflow: ellipsis;
    margin-top: 30px;
}

.company {
    width: 60%;
    margin-top: 30px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.5), 0 6px 20px 0 rgba(0, 0, 0,
0.39);
    border-radius: 25px;
    padding: 2%;
    overflow: auto;
    text-overflow: ellipsis;
}

.search_bar {
    margin-left: 70px;
    color: black;
}

.filters {
    margin-top: 30px;
    padding: 3%;
    background: white;
    border-radius: 25px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.4), 0 6px 20px 0 rgba(0, 0, 0,
0.36);
}

#selected_city, #query {
    width: 300px;
    height: 30px;
}

hr {
    border-color: black;

```

```
}

.company_img {
    margin: 30px;
    float: right;
    width: 150px;
    height: 150px;
    border-radius: 20px;
}

.company_img-lg {
    margin: 30px;
    float: right;
    width: 250px;
    height: 250px;
    border-radius: 20px;
}

#company_urls {
    text-align: center;
}

#company_urls a {
    margin-left: 30px;
}

.buttons_form {
    margin-right: 20px;
    margin-top: 20px;
}

.form-input {
    width: 300px;
    font-size: 16px;
}

.form-label {
    display: inline-block;
    min-width: 150px;
    vertical-align: top;
}
```

```

.form-error ul {
    list-style: none;
    padding: 0;
    color: #CC0000;
}

.form-button {
    min-width: 200px;
    font-size: 16px;
}

.auth-urls {
    float: right;
}

.user_card {
    padding: 20px 40px;
    color: white;
    background-color: #3f4136;
    width: 650px;
    height: 550px;
    position: absolute;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    margin: auto;
    justify-content: center;

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0,
0.19);
    -webkit-box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0,
0, 0.19);
    -moz-box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0,
0.19);
    border-radius: 5px;
}

.login_input * {
    margin-top: 20px;
    width: 200px;
}

```

```

.form_container {
    margin-top: 20px;
}

#form-title {
    color: #fff;
}

.login_btn {
    width: 100%;
    background-color: #FDA83D;
    /*background: #3f4136;*/
    color: white;
}

.login_btn:focus {
    box-shadow: none !important;
    outline: 0px !important;
}

.login_container {
    padding: 0 2rem;
}

.input-group-text {
    background: #f7ba5b;
    color: white ;
    border: 0 ;
    margin-top: 10px;
}

.input-group-text input {
    margin-left: 10px;
    width: 100%;
}

.input_user,
.input_pass:focus {
    box-shadow: none !important;
    outline: 0px !important;
}

```

```
.input-field {
    height: 50px;
    margin: 30px;
}
```

```
.input-text {
    width: 300px;
}
```

Б.8 Файл base_header_generic.html

```
{% load static %}
<head>
    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@9"></script>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOMLASjC"
crossorigin="anonymous">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">
    <link rel="stylesheet" href="{% static "style.css" %}">
    <link rel="icon" href="{% static "images/logo.png" %}">
    <title>{{ title }}</title>
</head>
```

Б.9 Файл base_navigation_generic.html

```
{% load static %}
<div class="header">
    <ul id="mainmenu" class="mainmenu">
        <li class="logo"><a href="/"></a></li>
        <li><a href="/vacancies">Знайти вакансії </a></li>
        {% if request.user.is_authenticated %}
```

```

        {% if user.is_candidate %}
            <li><a href="/candidate/{{ user.candidate.id
}}/vacanices/saved">Збережені вакансії </a></li>
            <li class="last">{{ user.first_name }} {{ user.last_name }},
профіль кандидата
            {% else %}
            <li><a href="/candidates">Знайти кандидатів </a></li>
            <li><a href="/company/{{ user.employer.company.id }}">Моя
компанія </a></li>
            <li class="last">{{ user.first_name }} {{ user.last_name }},
профіль роботодавця
            {% endif %}
        | <a href="{% url 'logout' %}">Вийти</a></li>

    {% else %}
        <li><a href="/login">Розмістити вакансії</a></li>
        <li class="last"><a href="/register">Реєстрація</a> | <a
href="/login">Увійти</a>
        </li>
    {% endif %}

</ul>
</div>

```

Б.10 Файл company.html

```

{% include "work4you_app/includes/base_header_generic.html" %}
{% include "work4you_app/includes/base_navigation_generic.html" %}

<!DOCTYPE html>
<html lang="en">
<body>
<div class="container company">
    {% if user.employer.company.image %}
        
    {% endif %}
    <div style="text-align: center"><h3><strong>{{ company.title
}}</strong></h3></div>
    <div style="margin-top: 30px">
        <div style="float:left;">

```

```

        <div><i class="fa-solid fa-city"></i>
            {% for category in company.categories.all %}
                {{ category.title }}{% if company.categories.all.count !=
forloop.counter %};{% endif %}
            {% endfor %}
        </div>
        <div><i class="fa fa-users"></i> {{ company.workers_count }}
працівників</div>
    </div>
    <div style="margin-left: 50%">
        <div><i class="fa-solid fa-location-dot"></i> {{ company.city }}
</div>
        <div><i class="fa fa-globe"></i> <a href="//{{ company.website_url
}}/"
                                                    target="_blank">{{
company.website_url }}</a></div>
    </div>
</div>
<div>
    <hr>
    <div id="company_urls">
        <a href="#description">Про компанію</a>
        <a href="#vacancies">Вакансії</a>
        <a href="#contacts">Контакти</a>
    </div>
    <hr>
</div>
<div id="description">{{ company.description }}</div>
<div>
    <hr>
    <div style="text-align: center" id="contacts">
        Контакти:
        {% if company.contacts.email %}
            <i class="fa fa-envelope"></i>
            {{ company.contacts.email }}
        {% endif %}
        {% if company.contacts.phone1 %}
            <i class="fa fa-phone"></i>
            {{ company.contacts.phone1 }}
        {% endif %}
        {% if company.contacts.phone2 %}
            <i class="fa fa-phone"></i>

```

```

        {{ company.contacts.phone2 }}
    {% endif %}
    {% if company.contacts.email and company.contacts.phone1 %}
        <a href="company/contacts/edit/{{ company.contacts.id }}"
style="margin-left: 20px">
            <button type="submit" class="btn btn-primary">Відредагувати
<i
                class="fa-solid fa-pen-to-square"></i>
            </button>
        </a>
    {% endif %}
</div>
{% if user.is_employer %}
    <hr>
    <div style="text-align: center">
        Доступні дії над компанією:
        <a href="/company/edit/{{ company.id }}" style="margin-left: 10px">
            <button type="button" class="btn btn-primary">Відредагувати <i
class="fa-solid fa-pen-to-square"></i></button>
        </a>
        <a href="/company/delete/{{ company.id }}" style="margin-left: 10px">
            <button type="button" class="btn btn-danger">Видалити <i class="fa-
solid fa-trash-can"></i></button>
        </a>
    </div>
    <hr>
{% endif %}
    {% if company.contacts is None %}
        <form method="get" action="/company/{{ company.id }}/contacts/add">
            <div>Контакти не вказані.</div>
            <button type="submit" class="btn btn-primary">Додати
контакти</button>
        </form>
    {% endif %}
    <hr>
    <h3>Вакансій: {{ vacancies.all.count }}</h3>
    {% if user.is_employer %}
        <form method="get" action="/company/{{ company.id }}/vacancy/add">
            <button type="submit" class="btn btn-primary">Додати</button>
        </form>
    {% endif %}
    <hr>

```

```

</div>
<div id="vacancies">
    {% for vacancy in vacancies %}
        <div class="vacancy">
            <h3><a href="/vacancy/{{ vacancy.id }}">{{ vacancy.title
}}</a></h3>

            <div>
                <i class="{{ vacancy.get_currency_sign }}"></i>{{
vacancy.salary }} {{ vacancy.company.city }}
                <span style="float: right; margin-right: 10px">Від {{
vacancy.creation_date }}</span>
            </div>

        </div>
    {% endfor %}
</div>
</div>
</body>
</html>

```

Б.11 Файл login.html

```

{% include "work4you_app/includes/base_header_generic.html" %}
{% include "work4you_app/includes/base_navigation_generic.html" %}
{% load sweetify %}
{% sweetify %}

<!doctype html>
<html lang="en">
<body>
    {% if messages %}

    {% endif %}
    <div class="user_card" style="height: 350px">
        <h1>{{ title }}</h1>
        <form method="POST">
            {% csrf_token %}
            <div class="input-group-text">
                <i class="fas fa-user"></i><input type="text" placeholder="Лорін"
class="input-text" name="username">
            </div>

```

```

    <div class="input-group-text">
        <i class="fas fa-key"></i><input type="password"
placeholder="Пароль" class="input-text" name="password">
    </div>

    <div class="justify-content-center mt-3 login_container">
        <div style="text-align: center">Увійти як</div>
        <input class="btn login_btn" type="submit" value="Кандидат"
name="candidate">
        <input class="btn login_btn" type="submit" value="Роботодавець"
name="employer" style="margin-top: 20px">
    </div>
    <ul class="messages">
        {% for message in messages %}
            <li>
                {% if message.tags %}
                    class="alert {{ message.tags }} d-flex justify-content-
center"
                {% endif %}>{{ message }}</li>
            {% endfor %}
        </ul>
    </form>

    {{ form.errors }}

</div>
</body>
</html>

```

Б.12 Файл register.html

```

{% include "work4you_app/includes/base_header_generic.html" %}
{% include "work4you_app/includes/base_navigation_generic.html" %}

<!doctype html>
<html lang="en">
<body>

<div class="user_card">

```

```

<h1>{{ title }}</h1>
<form method="POST" action="">
    {% csrf_token %}
    <div class="input-group-text"><i class="fas fa-user"></i>{{
form.username }}</div>
    <div class="input-group-text"><i class="fas fa-envelope-square"></i>{{
form.email }}</div>
    <div class="input-group-text"><i class="fa-solid fa-passport"></i>{{
form.first_name }}</div>
    <div class="input-group-text"><i class="fa-solid fa-passport"></i>{{
form.last_name }}</div>
    <div class="input-group-text"><i class="fas fa-key"></i>{{
form.password1 }}</div>
    <div class="input-group-text"><i class="fas fa-key"></i>{{
form.password2 }}</div>
    <div class="d-flex justify-content-center mt-3 login_container">
        <input class="btn login_btn" type="submit" value="Регистрация">
    </div>
</form>

<ol>
    {% if form.errors %}
        <ul>
            {% for field in form %}
                {% for error in field.errors %}

                    <li><strong>{{ error|escape }}</strong></li>

                {% endfor %}
            {% endfor %}
        </ul>
    {% endif %}
</ol>

</div>
{% load sweetify %}
{% sweetify %}
<script>
    var form_fields = document.getElementsByTagName('input')
    form_fields[1].placeholder = 'Логін';
    form_fields[2].placeholder = 'Електронна пошта';

```

```

form_fields[3].placeholder = 'Ім\''я';
form_fields[4].placeholder = 'Прізвище';
form_fields[5].placeholder = 'Пароль';
form_fields[6].placeholder = 'Підтвердіть пароль';

for (var field in form_fields) {
    form_fields[field].className += ' form-control'
}
</script>
</body>
</html>

```

Б.13 Файл candidate.html

```

{% include "work4you_app/includes/base_header_generic.html" %}
{% include "work4you_app/includes/base_navigation_generic.html" %}

<!DOCTYPE html>
<html lang="en">

<body>
<div class="row">
    <div class="col-md-2"></div>
    <div class="col-md-8 vacancy">
        {% if user.candidate.image %}
            
        {% endif %}
        <div>На сайті з {{ user.candidate.creation_date }}</div>
        <h3><strong>{{ user.first_name }} {{ user.last_name }}</strong></h3>
        <i class="fa-solid fa-city"></i>
        {{ user.candidate.city }}, {{ user.candidate.city.country }}
        {% if user.candidate.desired_vacancy %}
            <h4 style="margin-top: 10px"><strong>Бажанана вакансія:</strong> {{
user.candidate.desired_vacancy }}</h4>
        {% endif %}
        <div>
            <h3><strong>Опис кандидата</strong></h3>
            <ul>
                <li>Освіта: {{ user.candidate.education_level }}</li>
                <li>Досвід роботи: {{ user.candidate.experience_level }}</li>
            </ul>
        </div>
    </div>
</div>

```

```

</ul>

<h3><strong>Контакты</strong></h3>
<div>
    <i class="fa fa-envelope"></i>
    {{ user.email }}
</div>
<div>{% if user.candidate.phone %}
    <i class="fa fa-phone"></i>
    {{ user.candidate.phone }}
{% endif %}
</div>
<div style="margin-top: 30px">
    <a href="/save/vacancy/{{ user.candidate.id }}"
class="buttons_form">
        <button type="button" class="btn btn-primary">Зберігти <i
class="fa-solid fa-heart"></i></button>
    </a>
</div>
</div>
</div>
</body>
</html>

```

Б.14 Файл candidates.html

```

{% include "work4you_app/includes/base_header_generic.html" %}
{% include "work4you_app/includes/base_navigation_generic.html" %}

<!DOCTYPE html>
<html lang="en">
<body>
<div class="row">
    <div class="col-md-2"></div>
    <div class="col-md-5 search_bar">
        <h3 style="text-align: center"><strong>{{ title }}</strong></h3>
        <input type="text" id="query" name="query" placeholder="Запит" value="{{
selected_query }}">
        <select id="selected_city" name="city">
            {% for city in cities %}

```

```

        {% if selected_city.title == city.title %}
            <option selected>{{ selected_city }}</option>
        {% else %}
            <option>{{ city }}</option>
        {% endif %}
    {% endfor %}
</select>
<button type="submit" class="btn btn-primary">Знайти</button>
<label for="selected_city"> </label>
</div>
</div>
<form method="get" action="/find/vacancies">
    <div class="row vacancies">
        <div class="col-md-2"></div>
        <div class="col-md-5">
            {% if candidates.count > 0 %}
                {% for user in candidates %}
                    {% if user.candidate.desired_vacancy %}
                        <div class="vacancy">
                            {% if user.candidate.image %}
                                
                            {% endif %}
                            <div>На сайті з {{ user.candidate.creation_date
}}</div>

                            <h3>
                                <strong>
                                    <a href="/candidate/{{ user.candidate.id
}}">
                                        {{ user.first_name }} {{ user.last_name
}}
                                    </a>
                                </strong>
                            </h3>
                            <h4><strong>{{ user.candidate.desired_vacancy }}
</strong></h4>

                            <i class="fa-solid fa-city"></i>
                            {{ user.candidate.city }}, {{
user.candidate.city.country }}
                            <div>Освіта: {{ user.candidate.education_level }},
досвід: {{ user.candidate.experience_level }}</div>

```

```

        </div>
        {% endif %}
    {% endfor %}
    {% else %}
        <div class="col-md-3"></div>
        <div class="col-md-8 container alert-danger">За заданими
        фільтрами не знайдено жодної вакансії.</div>
    {% endif %}
</div>
<div class="col-md-1"></div>
<div class="col-md-3 filters" style="height: 450px">
    <center><h3>Пошук за фільтрами</h3></center>
    <h4>Освіта</h4>
    {% for edu_lvl in edu_lvls %}
        <div>
            <label for="edu_lvl_{{ edu_lvl.title }}">
                {% if edu_lvl in edu_lvls %}
                    <input type="checkbox" checked value="{{ edu_lvl }}"
                        id="edu_lvls_{{ edu_lvl }}" name="edu_lvls">
                {% else %}
                    <input type="checkbox" value="{{ edu_lvl }}"
id="edu_lvl_{{ edu_lvl }}" name="edu_lvls">
                {% endif %}
                {{ edu_lvl }}
            </label>
        </div>
    {% endfor %}
    <h4>Досвід</h4>
    {% for exp_lvl in exp_lvls %}
        <div>
            <label for="exp_lvl_{{ exp_lvl }}">
                {% if exp_lvl in edu_lvls %}
                    <input type="checkbox" checked value="{{ exp_lvl
}} id="exp_lvl_{{ exp_lvl }}"
                        name="exp_lvls">
                {% else %}
                    <input type="checkbox" value="{{ exp_lvl }}"
id="exp_lvl_{{ exp_lvl }}"
                        name="exp_lvls">
                {% endif %}
                {{ exp_lvl }}
            </label>
        </div>
    {% endfor %}

```

```

        </label>
    </div>
    {% endfor %}
</div>
</div>
</form>
</body>
</html>

```

Б.15 Файл vacancies.html

```

{% include "work4you_app/includes/base_header_generic.html" %}
{% include "work4you_app/includes/base_navigation_generic.html" %}

<!DOCTYPE html>
<html lang="en">
<body>
<form method="get" action="/vacancies/find">
    <div class="row">
        <div class="col-md-2"></div>
        <div class="col-md-5 search_bar">
            <h3 style="text-align: center"><strong>{{ title }}</strong></h3>
            <input type="text" id="query" name="query" placeholder="Запит"
value="{{ selected_query }}">
            <select id="selected_city" name="city">
                {% for city in cities %}
                    {% if selected_city.title == city.title %}
                        <option selected>{{ selected_city }}</option>
                    {% else %}
                        <option>{{ city }}</option>
                    {% endif %}
                {% endfor %}
            </select>
            <button type="submit" class="btn btn-primary">Знайти</button>
            <label for="selected_city"> </label>
        </div>
    </div>
    <div class="row" style="text-align: center; ">
        <div class="col-md-2"></div>
        <div class="col-md-5">
            {% for top_vacancy in top3 %}

```

```

        <a href="/vacancy/{{ top_vacancy.id }}">{{ top_vacancy.title
    }}</a>

        {% endfor %}
    </div>
</div>
<div class="row vacancies">
    <div class="col-md-2"></div>
    <div class="col-md-5">
        {% if vacancies.count > 0 %}
            {% for vacancy in vacancies %}
                <div class="vacancy">
                    {% if vacancy.company.image %}
                        
                    {% endif %}
                    <div>Вакансія від {{ vacancy.creation_date }}</div>
                    <h3><strong><a href="/vacancy/{{ vacancy.id }}">{{
vacancy }}</a></strong></h3>
                    <div><i class="{{ vacancy.get_currency_sign }}"></i> {{
vacancy.salary }}</div>
                    <div>
                        <i class="fa-solid fa-city"></i>
                        <a href="/company/{{ vacancy.company.id }}">{{
vacancy.company }}</a>
                    {% for category in vacancy.categories.all %}
                        {{ category.title }}{% if
vacancy.categories.all.count != forloop.counter %};
                    {% endif %}
                    {% endfor %}
                </div>
                <div><i class="fa-solid fa-location-dot"></i> {{
vacancy.company.city }}</div>
                <div style="display: flex">
                    {#
                    <a
href="/send/resume/{{ vacancy.company.id }}" class="buttons_form">#}
                    {#
                    <button
type="button" class="btn btn-primary">Відгукнутися <i#}
                    {#
                    class="fa-
solid fa-hand"></i></button>#}
                    {#
                    </a>#}
                    {% if user.candidate %}
                        <a href="/candidate/{{ user.candidate.id

```

```

}}/vacancy/{{ vacancy.id }}/save"

        class="buttons_form">
        {% else %}
            <a href="/login" class="buttons_form">
        {% endif %}
        <button type="button" class="btn btn-primary">
            {% if title == 'Збережені вакансії' %}Видалити
<i
            class="fa-solid fa-trash"></i>{% else %}
Зберегти <i
            class="fa-solid fa-heart"></i> {% endif
%}

        </button>
        </a>
    </div>
</div>
{% endfor %}
{% else %}
    <div class="col-md-3"></div>
    <div class="col-md-8 container alert-danger">За заданими
фільтрами не знайдено жодної вакансії.</div>
    {% endif %}
</div>
<div class="col-md-1"></div>
<div class="col-md-3 filters" style="height: 600px">
    <center><h3>Пошук за фільтрами</h3></center>
    <h4>Категорія</h4>
    <div>
        {% for category in categories %}
            <div>
                <label for="category_{{ category.title }}">
                    {% if category in selected_categories %}
                        <input type="checkbox" checked value="{{
category.title }}"
                            id="category_{{ category.title }}"
                            name="categories">
                    {% else %}
                        <input type="checkbox" value="{{ category.title
}}" id="category_{{ category.title }}"
                            name="categories">
                    {% endif %}
                    {{ category.title }}

```

```

        </label>
    </div>
    {% endfor %}
</div>
<div>
    <h4>Вид зайнятості</h4>
    {% for emp_type in employment_types %}
        <div>
            <label for="type_{{ emp_type }}">
                {% if emp_type in selected_employment_types %}
                    <input type="checkbox" checked value="{{
emp_type }}" id="type_{{ emp_type }}"
                                name="employment_types">
                {% else %}
                    <input type="checkbox" value="{{ emp_type }}"
id="type_{{ emp_type }}"
                                name="employment_types">
                {% endif %}
                {{ emp_type }}
            </label>
        </div>
    {% endfor %}
    <div><h4>Зарплата</h4></div>
    <label for="salary">Від</label>
    <input type="number" maxlength="6" id="salary" name="salary"
value="0">

    <select id="user_currency" name="user_currency">
        {% if selected_currency %}
            <option selected>{{ selected_currency }}</option> {%
endif %}

        {% for cur in currencies %}
            <option>{{ cur }}</option>
        {% endfor %}
    </select>
</div>
<div>
    <div><h4>Підходить, якщо ви</h4></div>
    {% for candidate_type in candidate_types %}
        <div>
            {% if candidate_type in selected_candidate_types %}
                <input type="checkbox" checked value="{{
candidate_type }}"

```

```

        id="candidate_{{ candidate_type }}"
        name="candidate_types">
    {% else %}
        <input type="checkbox"
            value="{{ candidate_type }}" id="candidate_{{
candidate_type }}"

            name="candidate_types">
    {% endif %}
    <label for="candidate_{{ candidate_type }}">{{
candidate_type }}</label>
    </div>
{% endfor %}
</div>
</div>
</div>
</form>
</body>
</html>

```

Б.16 Файл vacancy.html

```

{% include "work4you_app/includes/base_header_generic.html" %}
{% include "work4you_app/includes/base_navigation_generic.html" %}

<!DOCTYPE html>
<html lang="en">
<body>
<div class="row">
    <div class="col-md-2"></div>
    <div class="col-md-8">
        {% if vacancy_details %}
            <div class="container vacancy">
                {% if vacancy_details.company.image %}
                    
                {% endif %}
                <div>Вакансія від {{ vacancy_details.creation_date }}</div>
                <h3><strong>{{ vacancy_details }}</strong></h3>
                <div><i class="{{ vacancy_details.get_currency_sign }}"></i> {{
vacancy_details.salary }}</div>
                <div>

```

```

        <i class="fa-solid fa-building"></i>
        <a href="/company/{{ vacancy_details.company.id }}">{{
vacancy_details.company }}</a>
        <div>
            <i class="fa-solid fa-briefcase"></i>
            {% for category in vacancy_details.categories.all %}
                {{ category.title }};
            {% endfor %} {{ vacancy_details.company.workers_count }}
працівників
        </div>
        <i class="fa-solid fa-location-dot"></i> {{
vacancy_details.company.city }}
    </div>
    <div>
        <i class="fa fa-check" aria-hidden="true"></i>
        {{ vacancy_details.employment_type }}
        {% for candidate_type in vacancy_details.candidate_types.all
%}

            {% if candidate_type.title == 'Переселенець' %}, готові
взяти переселенця{% endif %}

            {% if candidate_type.title == 'Студент' %}, готові взяти
студента{% endif %}

            {% if candidate_type.title == 'Пенсіонер' %}, готові
взяти пенсіонера{% endif %}

            {% if candidate_type.title == 'Без досвіду' %}, готові
взяти людину без досвіду{% endif %}

        {% endfor %}.
    </div>
    <div>
        <h3><strong>Опис вакансії</strong></h3>
        {{ vacancy_details.description }}
        <h3><strong>Вимоги до кандидата</strong></h3>
        <ul>
            <li>Освіта: {{ vacancy_details.education_level }}</li>
            <li>Досвід роботи: {{ vacancy_details.experience_level
}}</li>

        </ul>
    </div>
    <div id="contacts">
        <h3><strong>Контакти</strong></h3>
        <div>{% if vacancy_details.company.contacts.email %}
            <i class="fa fa-envelope"></i>

```

```

        {{ vacancy_details.company.contacts.email }}
    {% endif %}
</div>
<div>{% if vacancy_details.company.contacts.phone1 %}
    <i class="fa fa-phone"></i>
    {{ vacancy_details.company.contacts.phone1 }}
{% endif %}</div>
<div>{% if vacancy_details.company.contacts.phone2 %}
    <i class="fa fa-phone"></i>
    {{ vacancy_details.company.contacts.phone2 }}
{% endif %}</div>
</div>
<div style="display: flex">
    {% if user.is_employer and vacancy_details.company.id ==
user.employer.company.id %}
        <form action="vacancy/edit/{{ vacancy_details.id }}"
class="buttons_form">
            <button type="submit" class="btn btn-
primary">Відредагувати <i class="fa-solid fa-pen-to-square"></i>
            </button>
        </form>
        <form action="vacancy/delete/{{ vacancy_details.id }}"
class="buttons_form">
            <button type="submit" class="btn btn-
danger">Видалити <i class="fa-solid fa-trash-can"></i>
            </button>
        </form>
    {% else %}
        <form action="candidate/{{ user.candidate.id
}}/vacancy/<int:vacancy_id>/save"
class="buttons_form">
            <button type="submit" class="btn btn-
primary">Зберегти <i class="fa-solid fa-heart"></i>
            </button>
        </form>
    {% endif %}
</div>
</div>
{% endif %}
</div>
</div>

```

```
</body>
</html>
```

Б.17 Файл form.html

```
<!DOCTYPE html>
<html lang="en">
<body>
{% include "work4you_app/includes/base_header_generic.html" %}
{% include "work4you_app/includes/base_navigation_generic.html" %}
<div class="col-md-8 mx-auto">
    <h1 style="text-align: center"> {{ title }}</h1>
    <form method="post" class="vacancy" enctype="multipart/form-data">
        {% csrf_token %}
        <div class="form-error">{{ form.non_field_errors }}</div>
        {% for f in form %}
            <p><label class="form-label" for="{{ f.id_for_label }}">{{ f.label
}}: </label>{{ f }}</p>
            {% endfor %}
            <button type="submit" class="btn btn-primary">Додати</button>
        </form>
    </div>
</body>
</html>
```

ДОДАТОК В
Слайди презентації

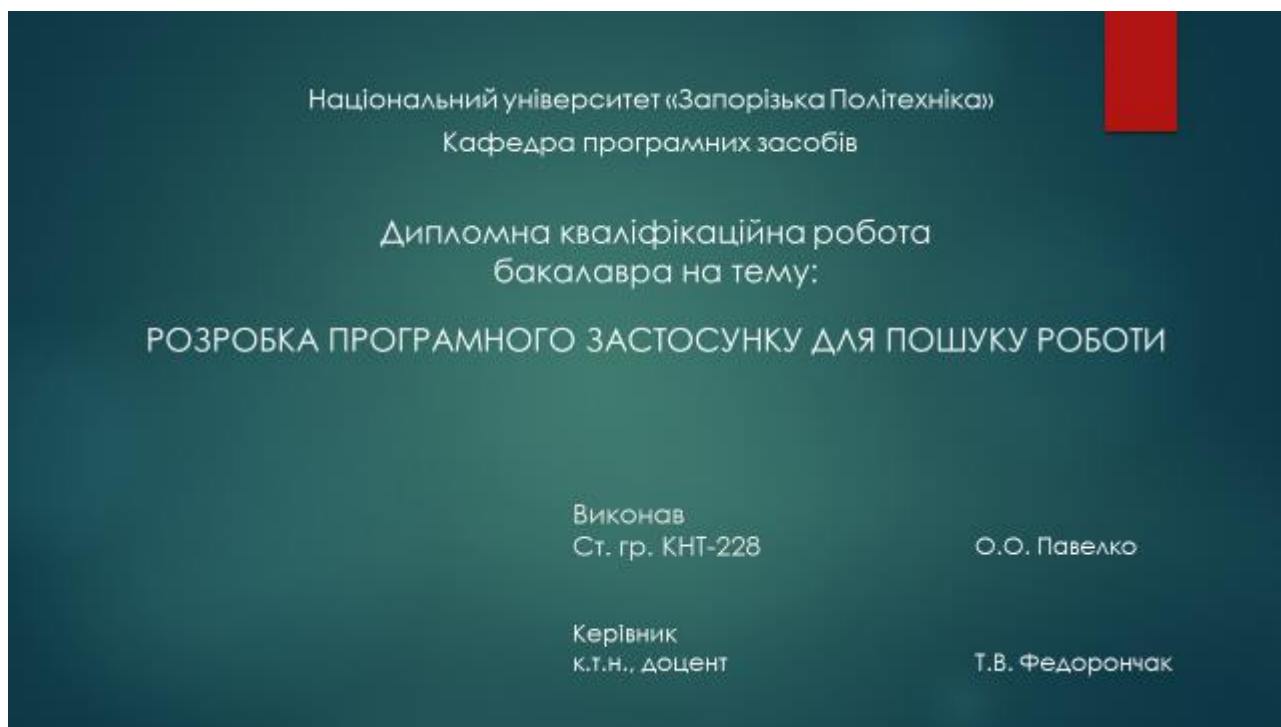


Рисунок В.1 – Слайд 1

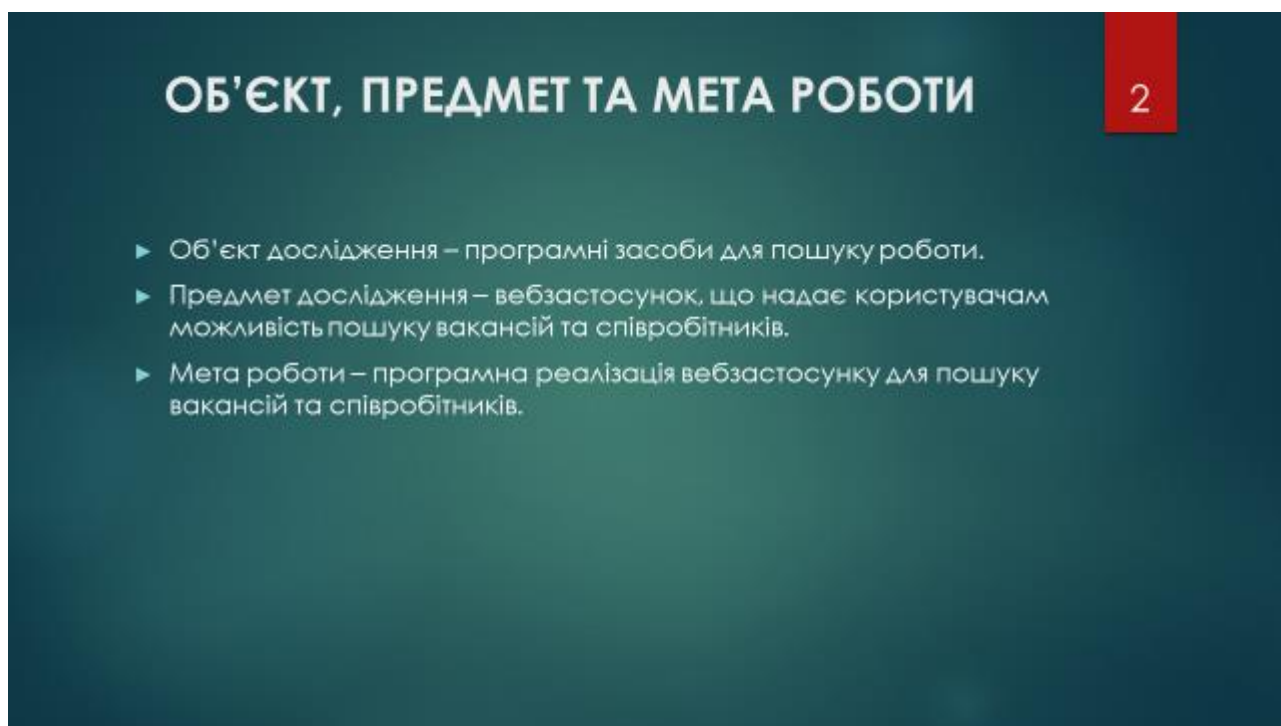


Рисунок В.2 – Слайд 2

Сервіси-аналоги Work.ua, Olx.ua, Rabota.ua

3

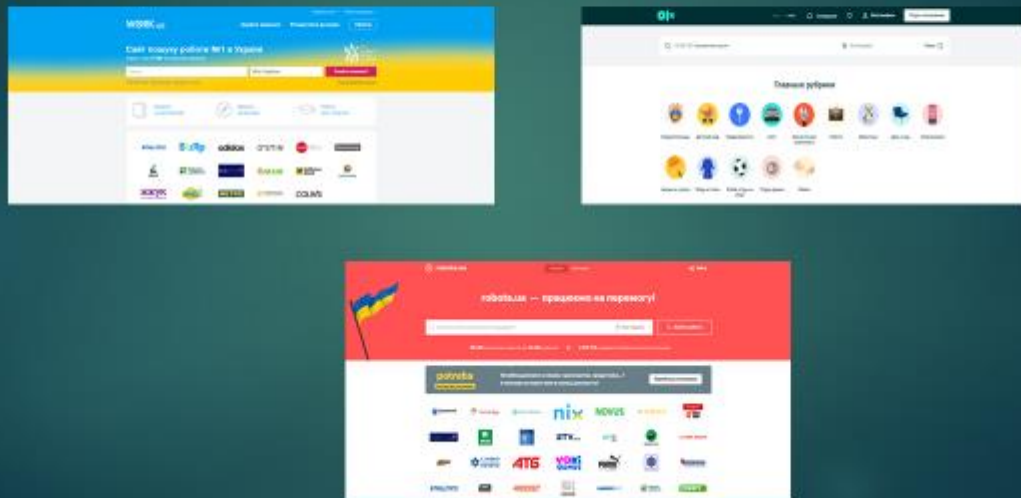


Рисунок В.3 – Слайд 3

Порівняння аналогів

4

Характеристика	«Work.ua»	«Rabota.ua»	«Olx.ua»
Виконання основного функціоналу	+	+	+
Багатоплатформність	-	+	+
Зручний інтерфейс	-	+	+
Платність	+	+	-
Підтримка української мови	+	+	+
Пошук вакансій за фільтрами	+	+	-
Вбудований конструктор резюме	-	+	-

Рисунок В.4 – Слайд 4

Порівняння фреймвориків для створення вебзастосунку

5

Характеристика	«Django»	«Spring»
Область застосування	Веб розробка, штучний інтелект	Веб розробка, системне адміністрування
Мова програмування	Python	Java
Шаблон проектування	Model-Template-View	Model-View-Controller
Швидкість роботи	Більша	Менша
Авторизація користувачів	Вбудована	Необхідно реалізовувати самостійно
Панель адміністратора	Вбудована	Необхідно реалізовувати самостійно
Локалізація текстових ресурсів	Вбудована	Необхідно реалізовувати самостійно

Рисунок В.5 – Слайд 5

Порівняння СУБД для створення вебзастосунку

6

Характеристика	SQLite	MySQL	PostgreSQL
Тип бази даних	Файлова	Серверна	Серверна
Операційна система	Безсерверна	Linux, Windows, OS X, Solaris, FreeBSD	FreeBSD, Linux, Windows, OS X, Solaris, Unix
Підтримувані мови	ActionScript, Python, C, C#, C++, PHP, Java, JavaScript	C, C++, Delphi, Perl, Java, PHP, Lua, .NET, Node.js	.NET, C, C++, Delphi, Java, Python, PHP, Perl
Видання	Community з функцією PRO підтримки	Community free standard та Enterprise edition	Community з функцією PRO допомоги

Рисунок В.6 – Слайд 6

7



7

8



8



8



8

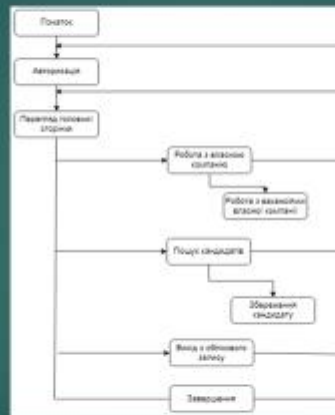
8

Функціональні схеми роботи програми

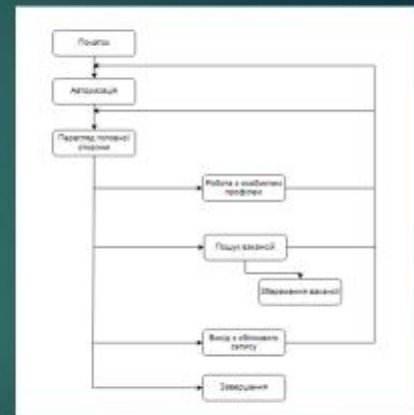
9



Функціональна схема програми роботи адміністратора



Функціональна схема програми роботи роботодавця



Функціональна схема програми роботи кандидата

Рисунок В.9 – Слайд 9

Структура проекту

10

work4you	update gitignore	7 hours ago
work4you_app	add authentication, user roles, entity pages, change styles	7 hours ago
gitignore	update gitignore	7 hours ago
image.jpg	create models, project files	15 days ago

Структура кореневої папки проекту «work4you»

work4you	create models, project files
work4you_app	add authentication, user roles, entity pages, change styles
gitignore	update gitignore
image.jpg	create models, project files

Структура класів пакету «work4you»

work4you	add authentication, user roles, entity pages, change styles
work4you_app	add authentication, user roles, entity pages, change styles
work4you_app	add authentication, user roles, entity pages, change styles
work4you_app	add authentication, user roles, entity pages, change styles
work4you_app	add authentication, user roles, entity pages, change styles
work4you_app	add authentication, user roles, entity pages, change styles
work4you_app	add authentication, user roles, entity pages, change styles
work4you_app	add authentication, user roles, entity pages, change styles

Структура класів пакету «work4you_app»

Рисунок В.10 – Слайд 10

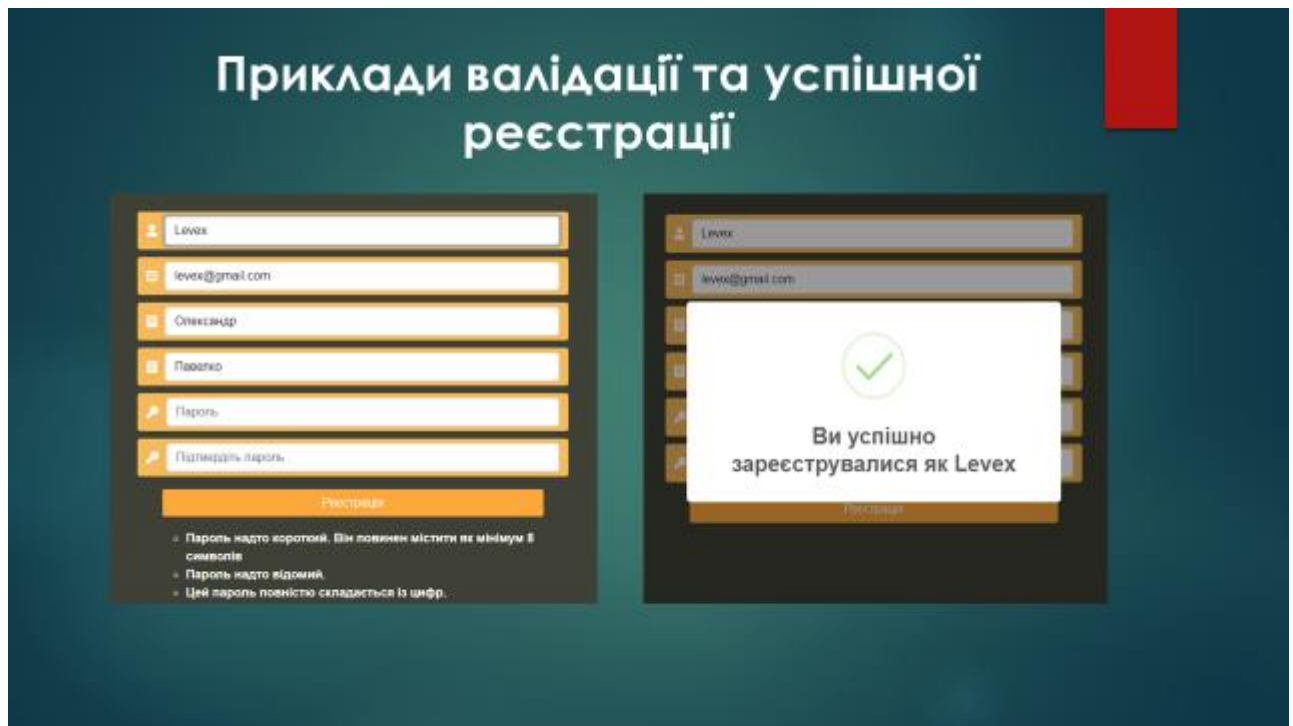


Рисунок В.11 – Слайд 11

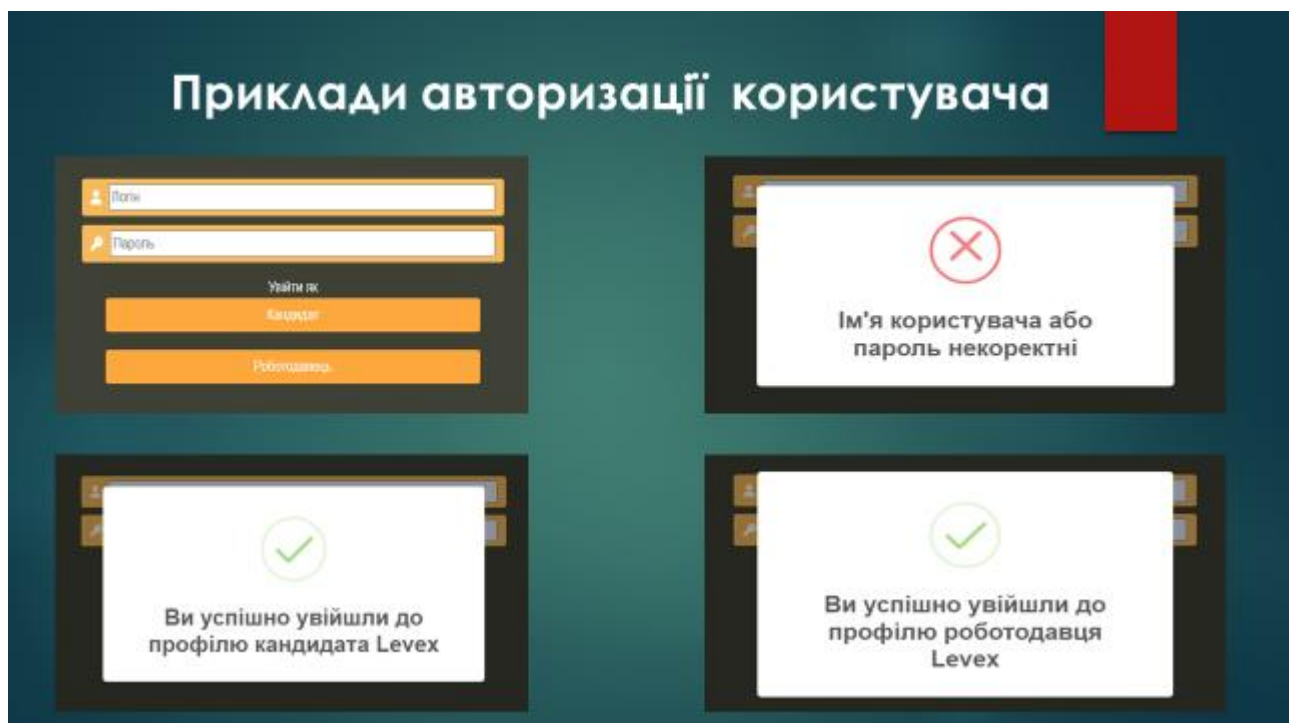


Рисунок В.12 – Слайд 12

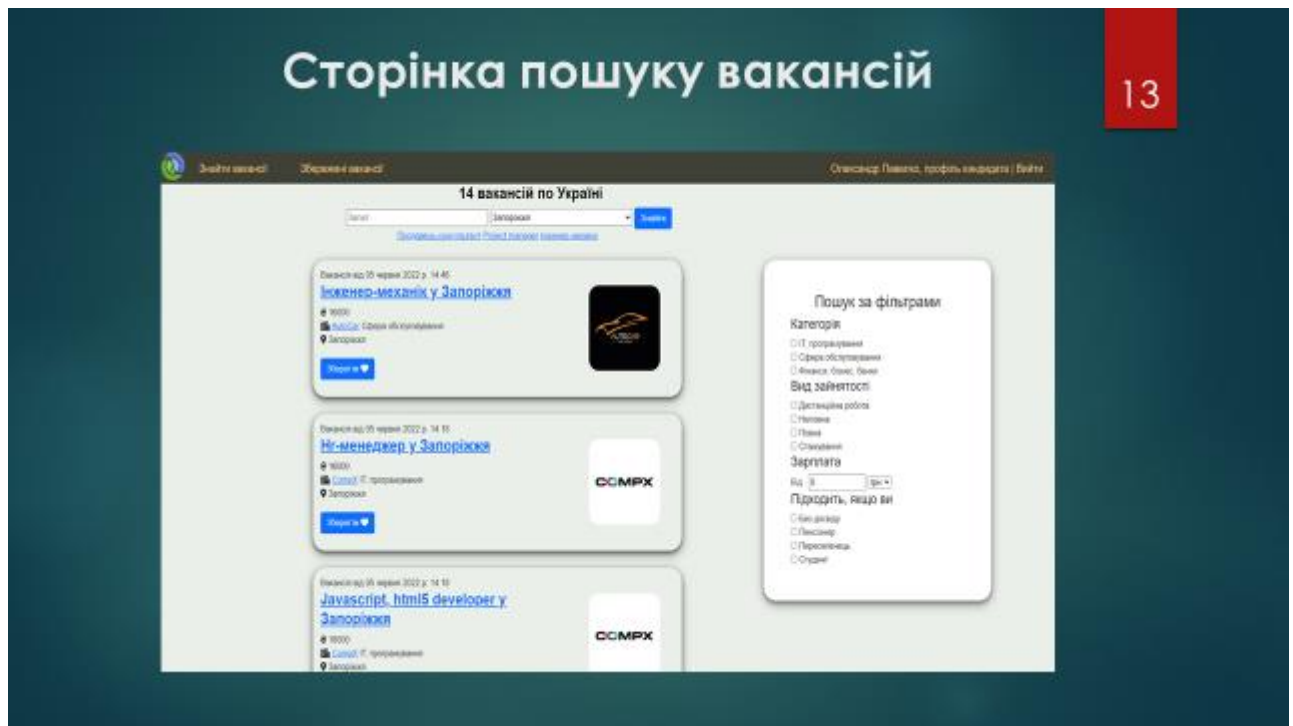


Рисунок В.13 – Слайд 13

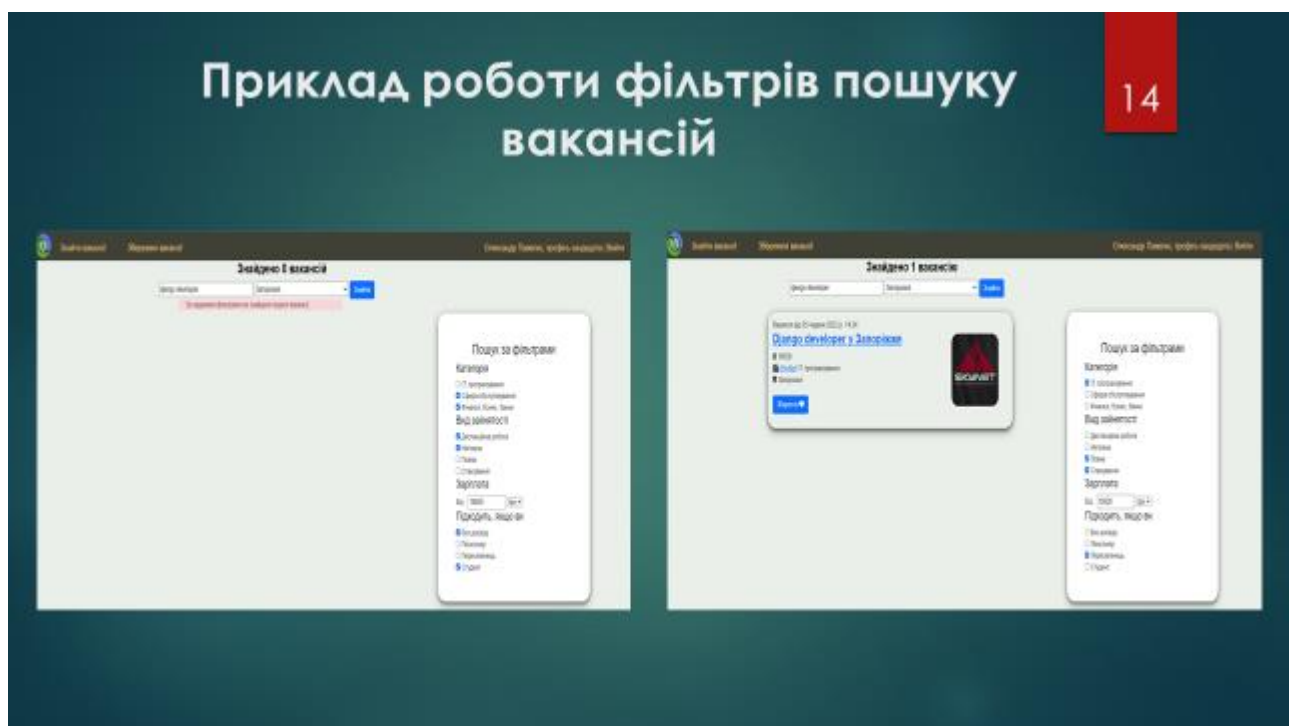


Рисунок В.14 – Слайд 14

Сторінки перегляду вакансії та списку збережених вакансій користувача

15



Рисунок В.15 – Слайд 15

Сторінка перегляду компанії роботодавця

16



Рисунок В.16 – Слайд 16

Сторінка додавання компанії роботодавця

17

Додати компанію

Назва компанії:

Адреса (включно з поштою):

Вебсайт:

Місто:

Тип компанії:

Опис:

Вибрати адресу:

Вибрати тип компанії:

Зберегти

Рисунок В.17 – Слайд 17

Сторінка перегляду компанії роботодавця після заповнення даними

18

IT Masters

IT-програмування
150-500 працівників

Детальний профіль роботи

Зберегти

Вакансії: 1

Зберегти

javascript developer

15000 Зарплата

Від 10 червня 2021 о 18:01

Рисунок В.18 – Слайд 18

Сторінка пошуку та перегляду кандидатів

19

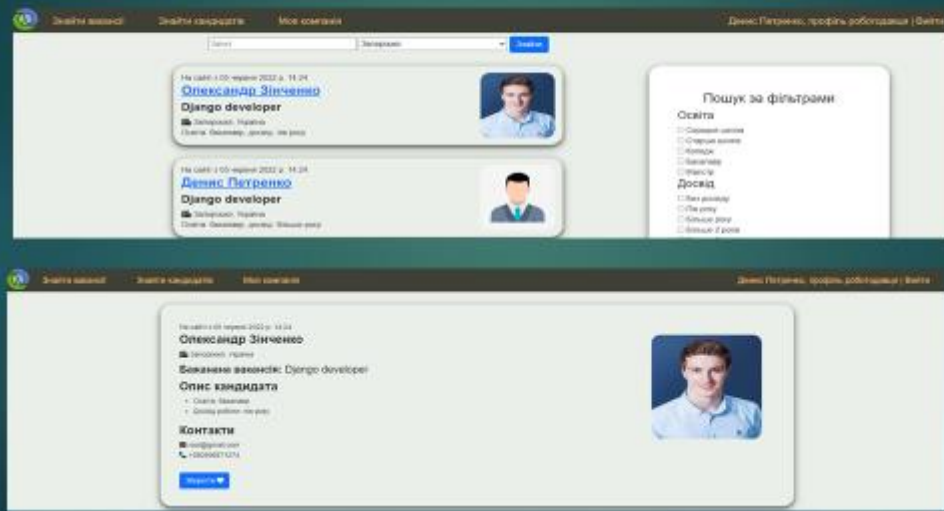


Рисунок В.19 – Слайд 19

Робота з панеллю адміністратора

20

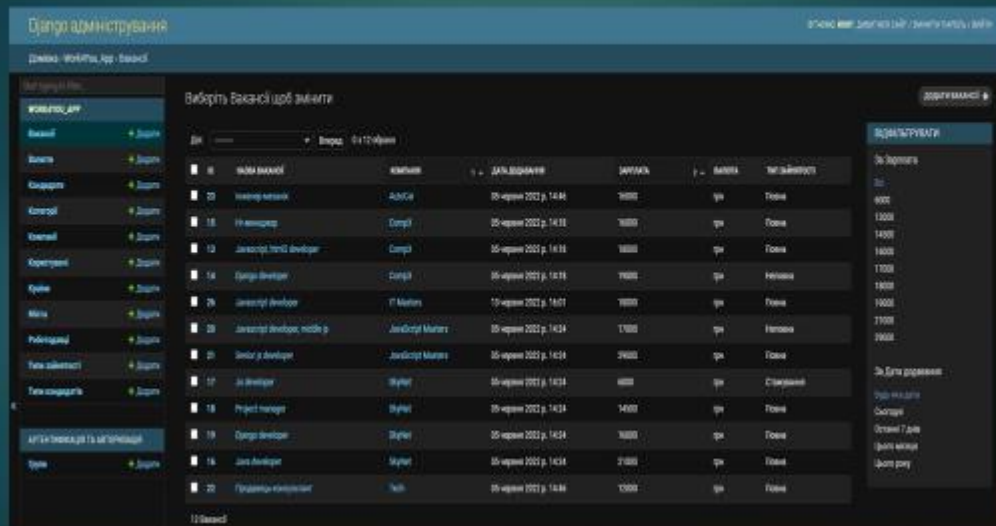


Рисунок В.20 – Слайд 20

Тест процесу авторизації

21

	Windows 7	Windows 8	Windows 10
Оперативна пам'ять 256 мб Процесор 1 ядро	2.4 секунд	3.1 секунд	3.1 секунд
Оперативна пам'ять 512 мб Процесор 1 ядро	2 секунд	2.4 секунд	2.4 секунд
Оперативна пам'ять 256 мб Процесор 2 ядра	2.1 секунд	2.5 секунд	2.7 секунд
Оперативна пам'ять 512 мб Процесор 2 ядра	1.9 секунд	2.3 секунд	2.0 секунд

Рисунок В.21 – Слайд 21

ВИСНОВКИ

22

- ▶ Під час виконання дипломної кваліфікаційної роботи бакалавра було розроблено вебзастосунок для пошуку вакансій та співробітників «Work4You».
- ▶ Після проведення існуючих аналогів було створено вебзастосунок, який, на думку автора, включає тільки переваги.
- ▶ Програму «Work4You» можна використовувати для малого та великого бізнесу, підприємств, які хочуть знайти співробітників.
- ▶ Реалізоване програмне забезпечення відповідає всім вимогам, поставленим в технічному завданні до проекту. Отримана програма є стійкою до змін і передбачає можливості свого подальшого розширення.

Рисунок В.22 – Слайд 22

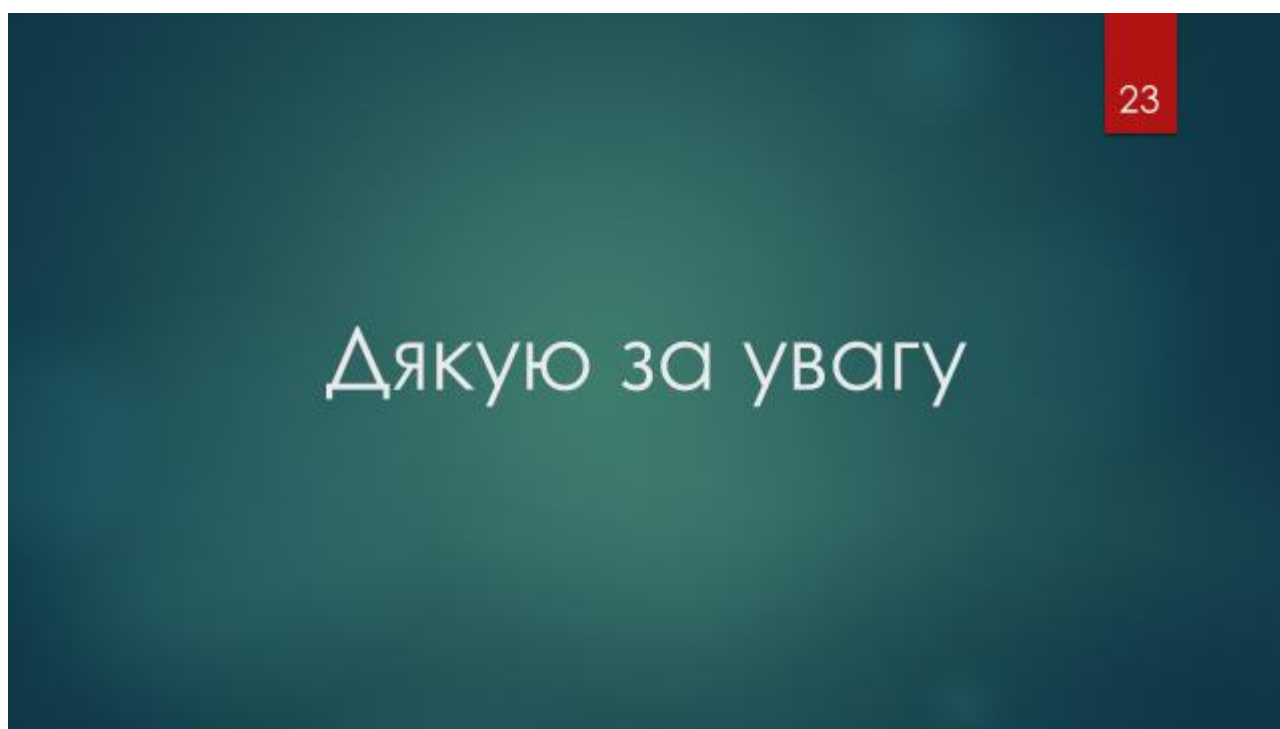


Рисунок В.23 – Слайд 23