

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОШУКУ ТА

(назва теми)

ПУБЛІКАЦІЇ ВАКАНСІЙ ДЛЯ ПРАЦЕВЛАШТУВАННЯ

SOFTWARE FOR SEARCHING AND POSTING JOB

VACANCIES

Виконав(ла): студент(ка) 4 курсу, групи КНТ-119

Спеціальності 121 Інженерія програмного

забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ГОРОБЕЦЬ В.І.

(ПРІЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРІЗВИЩЕ та ініціали)

Рецензент ТІМЕНКО А.В.

(ПРІЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ГОРОБЦЯ Владислава Ігоровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення пошуку та публікації вакансій для працевлаштування

Software for Searching and Posting Job Vacancies

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

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 року

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

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

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 04 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Владислав ГОРОБЕЦЬ
(ПРІЗВИЩЕ та ініціали)

Керівник проєкту (роботи)

(підпис) Валерій ЛЬОВКІН
(ПРІЗВИЩЕ та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
109 с., 37 рис., 5 табл., 3 дод., 24 джерел.

ВАКАНСІЯ, РЕЗЮМЕ, РОБОТОДАВЕЦЬ, ТЕСТУВАННЯ, AJAX,
BOOTSTRAP, POSTGRESQL, DJANGO, HTML, CSS, MVT, PYTHON.

Об'єкт роботи – процес пошуку та публікації вакансій для працевлаштування. Предмет роботи – програмне забезпечення пошуку та публікації вакансій для працевлаштування. Мета роботи – розробити програмне забезпечення для пошуку та публікації вакансій для працевлаштування з удосконаленням процесу відбору кандидатів через виконання первинного тестування користувачів.

Матеріали, методи та технічні засоби: мови програмування Python, JavaScript, HTML, CSS, вебфреймворк Django, шаблон проєктування Model-View-Template, AJAX, база даних PostgreSQL.

Результати. Розробку програмного забезпечення вебзастосунку пошуку та публікації вакансій для працевлаштування з можливістю первинного тестування користувачів виконано за допомогою Django з визначенням структури даних, вимог до програми, розробленням структури і алгоритму роботи програми на основі цих вимог, реалізацією відповідних одиниць. Програма дозволяє переглядати вакансії, сортувати їх за категоріями, а для створення резюме, вакансій, проходження первинного тестування користувач має зареєструватись.

Висновки. Розроблений програмний засіб дозволяє знайти користувачу необхідну вакансію, сприяючи швидкості та зручності даного процесу, а роботодавцю – співробітника з потрібною кваліфікацією.

Галузь використання. Ринок праці. Програмний засіб може використовуватися в якості вебзастосунку для пошуку та публікації вакансій для працевлаштування, а також для підбору співробітників.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 109 pages, 37 figures, 5 tables, 3 appendixes, 24 sources.

VACANCY, RESUME, EMPLOYER, TESTING, AJAX, BOOTSTRAP, POSTGRESQL, DJANGO, HTML, CSS, MVT, PYTHON.

Object of work is the process of searching and publishing job vacancies. Subject of work is software for searching and publishing job vacancies. Aim of the paper - to develop software for searching and publishing job vacancies with the improvement of the candidate selection process through the implementation of initial user testing.

Materials, methods and technical tools: programming languages Python, JavaScript, HTML, CSS, Django web framework, Model-View-Template design template, AJAX, PostgreSQL database.

Results. The development of software for a web application for searching and publishing job vacancies with the possibility of initial user testing was performed by using Django with the definition of the data structure, program requirements, development of the structure and algorithm of the program based on these requirements, and implementation of the corresponding units. The program allows you to view vacancies, sort them by category, and to create a resume, vacancies, and pass the initial testing, the user have to sign up.

Conclusions. The developed software tool allows a user to find the necessary vacancy, contributing to the speed and convenience of this process, and tan employer to find an employee with the necessary qualifications.

Scope of use. The labor market. The software tool can be used as a web application for searching and publishing jobvacancies , as well as for recruiting employees.

ЗМІСТ

С.

Скорочення та умовні позначки	8
Вступ	9
1 Аналіз предметної області	11
1.1 Постановка проблеми	11
1.2 Огляд існуючих програмних аналогів	12
1.3 Порівняння існуючих аналогів	16
1.4 Висновки за розділом 1	17
2 Проєктування програми	18
2.1 Вибір засобів для проєктування і реалізації	18
2.2 Визначення функціональних вимог до програми	20
2.3 Вибір та обґрунтування системи керування базою даних	24
2.4 Проєктування БД	25
2.5 Визначення структури вебсистеми	32
2.6 Структура вебсистеми	33
2.7 Висновки за розділом 2	35
3 Розробка програми	36
3.1 Алгоритм функціонування програми	36
3.2 Опис реалізованих класів моделей	37
3.3 Опис реалізованих класів представлень	40
3.4 Висновки за розділом 3	44
4 Експлуатація та тестування програми	45
4.1 Призначення програми	45
4.2 Умови виконання програми	45
4.3 Експлуатація програми	45
4.4 Тестування програми	54
4.5 Висновки за розділом 4	56
Висновки	58
Перелік джерел посилань	59

Додаток А Технічне завдання.....	61
Додаток Б Текст програми.....	66
Додаток В Слайди презентації	101

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

MVC – model-view-controller;

MVT – model-view-template;

URL – uniform resource locator;

БД – база даних;

ПЗ – програмне забезпечення;

СКБД – система управління базами даних.

ВСТУП

В наш час актуальність пошуку роботи не лише не зменшується, але навіть стає все більшою. Різні обставини змушують людей шукати нові місця роботи, але зараз, завдяки Інтернету, цей процес стає набагато простішим та швидшим.

Інтернет забезпечує можливість шукати роботу без витрати часу на перегляд оголошень в газетах та простій у чергах. Можна просто знайти відповідну вакансію та відправити лист зі своїм резюме. І все це можна зробити, не виходячи з дому та не втрачаючи свій час на дорогу.

Однак, серед великої кількості сайтів з працевлаштування, дуже важливо знайти той, який надає якісні та професійні послуги. Так само роботодавці повинні мати можливість швидко та ефективно знайти найкращих працівників. А цей процес займає значну кількість часу та зусиль, і його необхідно вдосконалювати.

З появою технологій штучного інтелекту, відбір працівників стає ще більш складним завданням. Автоматично згенеровані пропозиції можуть створювати непередбачувані складнощі, спотворюючи реальний образ кандидатів та ускладнюючи відбір серед численних анкет та резюме.

Це додатково ускладнює процес найму та може перешкоджати досягненню оптимального результату.

Для цього пропонується використовувати первинне тестування, яке дозволить ефективно відсіювати кандидатів, які не відповідають вимогам вакансії. Це дасть можливість роботодавцям сконцентрувати свій час та зусилля на кандидатах, які вже підійшли за початковими вимогами, та зменшить загальний час, потрібний на процес найму.

Такий підхід не тільки допоможе зменшити навантаження на роботодавців, але й дозволить підвищити якість найму та забезпечити підприємство новими співробітниками, які відповідають вимогам кожної посади. Отже, використання первинного тестування є важливим кроком у процесі найму нових працівників.

Звідси і необхідність розробки програмного засобу (ПЗ) для пошуку та публікації вакансій, яке забезпечує швидкий та ефективний пошук відповідних вакансій для працевлаштування та дозволяє публікувати власні вакансії для приваблення кандидатів. Це може бути корисним для різних галузей бізнесу та допоможе зменшити час та зусилля, необхідні для знаходження та приваблення талантів.

В результаті таке ПЗ дозволить роботодавцям знайти найкращих фахівців, а кандидатам на працевлаштування – найкращі робочі місця для їхнього професійного розвитку. Тож результати цієї роботи можуть бути взаємно корисним для обох груп учасників ринку праці.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка проблеми

Будь-яка компанія, що починає свою діяльність або швидко розвивається, потребує досвідчених співробітників, які сприятимуть подальшому зростанню та просуванню компанії на ринку. На перший погляд, сучасний ринок праці не відчуває нестачі в кваліфікованих претендентах, які шукають роботу, проте на практиці знайти відповідного за всіма параметрами співробітника виявляється не так просто.

Як правило, пошук кандидатів на посаду в середній або великій компанії здійснює менеджер з персоналу, який залучає всі доступні канали: газети, Інтернет, радіо, телебачення, власну базу даних (БД) компанії, особисті знайомства, агенції з найму працівників. Враховуючи зокрема дані в роботі [1], слід зрозуміти, що найчастіше вибір того чи іншого джерела залежить від розмірів компанії, а також коштів та часу, які вона готова витратити на пошук співробітника.

Практично всі компанії відчувають потребу у підборі персоналу. Для одних це одиниці співробітників, для інших – десятки та сотні.

Залежно від чисельності персоналу, всі компанії вирішують проблему підбору індивідуально.

Для допомоги використовується Інтернет із його широкими можливостями. Зареєструвавшись на одному або кількох сайтах, роботодавець отримує можливість помістити свою вакансію, оформити передплату на регулярне отримання нових резюме, а також самостійно здійснювати пошук кандидатів на посаду в великому банку резюме. Коли робота зроблена і потрібно відібрати лише необхідних кандидатів, не витрачаючи багато зусиль, на допомогу приходить первинне тестування, у якому відразу стане видно професіоналізм претендента та його особисті якості (психологічне тестування).

У будь-якому випадку слід пам'ятати, що підбір персоналу – не рутина, а захоплююче заняття, яке потребує уваги та творчого підходу.

1.2 Огляд існуючих програмних аналогів

Залучення кваліфікованих та некваліфікованих кадрів має одну загальну початкову стадію – оповіщення потенційної аудиторії про відкриту вакансію. Цей огляд направлений на аналіз ресурсів-аналогів в Україні щодо пошуку та публікації вакансій [2].

Однією із платформ для пошуку роботи є Robota.ua [3] (рис. 1.1). Це популярний український портал, відомий зокрема і за рахунок цілком очевидної як для вебресурсу про працевлаштування назвою.

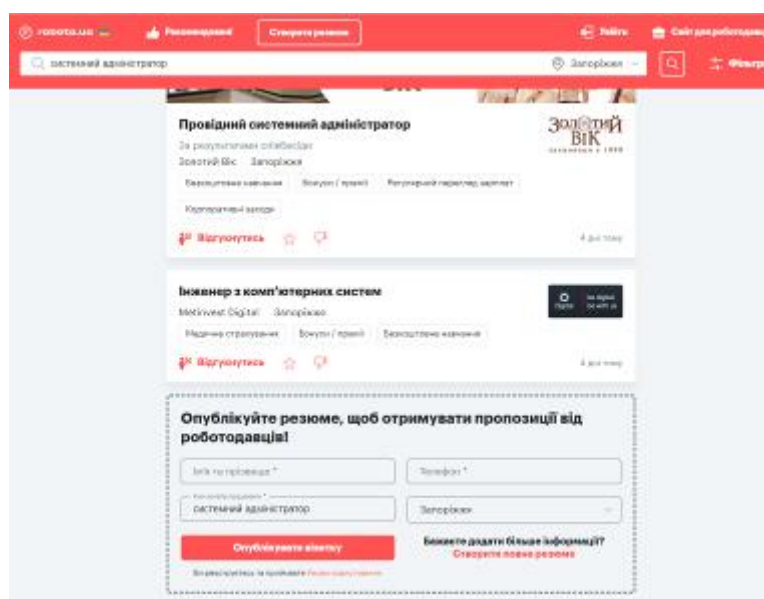


Рисунок 1.1 – Robota.ua

Вебзастосунок «Robota.ua» має таку функціональність:

- можливість реєстрації / авторизації користувача;
- можливість перегляду вакансій;
- можливість додавання вакансій роботодавцем.

До недоліків даного вебресурсу потрібно віднести:

- нав'язлива реклама;
- відсутність реєстрації / авторизації через сервіси GitHub, GitLab;
- багато платних функцій;

- відсутність первинного тестування кандидатів на вакансію.

Наступний портал з працевлаштування в Україні – це Work.ua [4] (рис. 1.2).

До позитивних сторін сайту можна віднести наступне:

- інтуїтивно зрозумілий сервіс;
- можливість додавання вакансій;
- можливість перегляду вакансій;
- можливість реєстрації / авторизації користувача.

До недоліків даного вебресурсу потрібно віднести:

- громіздке складання резюме, багато вигаданих правил;
- великі обмеження безкоштовного облікового запису;
- незручна система фільтрації (без блокувань компаній / вакансій, задатами / зарплатою);
- нав'язлива реклама;
- відсутність реєстрації / авторизації через сервіси GitHub, GitLab;
- відсутність первинного тестування кандидатів на вакансію.

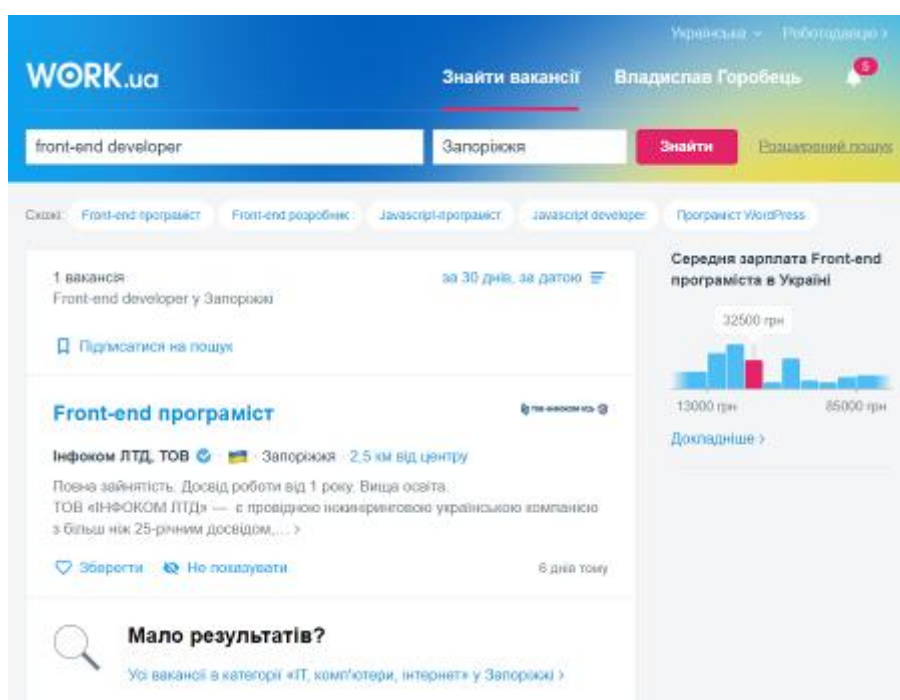


Рисунок 1.2 – Work.ua

Jobs.ua – вебпортал з пошуку роботи [5] (рис. 1.3). За його допомогою можна залучити і кваліфікованих працівників, і некваліфікованих. Є вакансії з пошуку прибиральниць, менеджера з продажу, спеціаліста з розробки ПЗ, сантехніка тощо.

До плюсів можна віднести:

- доступна функціональність дозволяє зручно публікувати оголошення;
- можливість додавання вакансій;
- можливість перегляду вакансій;
- можливість реєстрації / авторизації користувача.

До мінусів:

- досить багато додаткових неважливих функцій;
- створення оголошень ускладнене в порівнянні з аналогічними порталами;
- багато додаткових розділів («Тренінги», «Освіта»), які загалом не потрібні авторам оголошень;
- відволікаюча реклама;
- відсутність реєстрації / авторизації через GitHub, GitLab;
- відсутність первинного тестування кандидатів на вакансію.

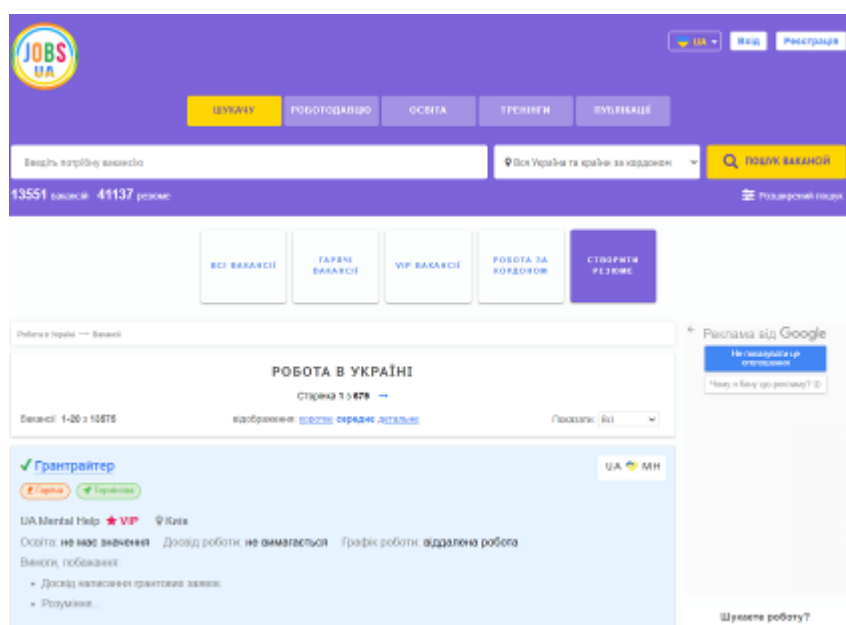


Рисунок 1.3 – Jobs.ua

Trud.ua – портал з пошуку персоналу в Україні [6] (рис. 1.4).

Позитивні сторони сайту:

- можливість додавання вакансій;
- можливість перегляду вакансій;
- можливість реєстрації / авторизації користувача.

Негативні сторони сайту:

- відсутність реєстрації / авторизації через сервіси GitHub, GitLab;
- багато реклами;
- відсутність первинного тестування кандидатів на вакансію;
- по-справжньому ефективна робота з даним порталом можлива лише у разі придбання платних функцій;
- робота з сайтом є не досить зручною.

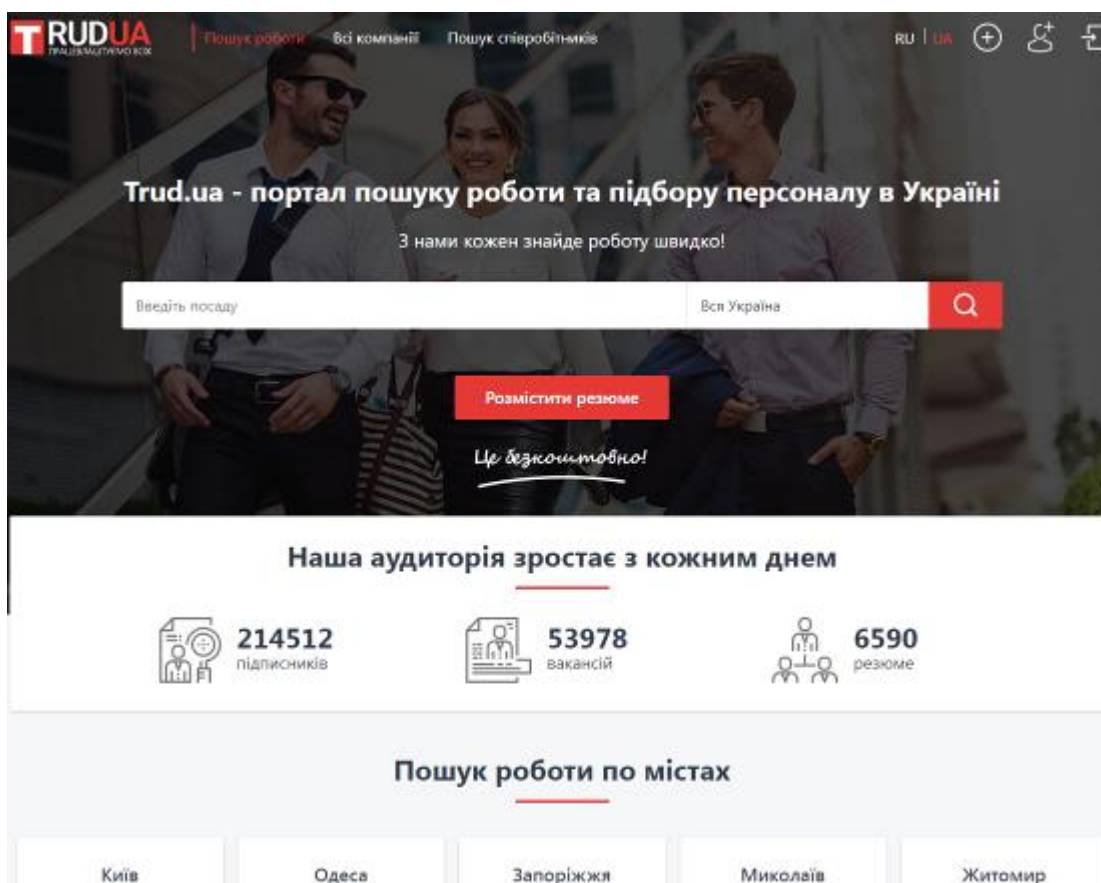


Рисунок 1.4 – Trud.ua

1.3 Порівняння існуючих аналогів

За результатами огляду було виконано порівняння основних існуючих засобів з тією програмою, яку необхідно в роботі розробити (табл. 1.1). В якості особливостей розробки визначена категорія ресурсу, наявність первинного тестування, наявність рекламних оголошень, платні функції, наявність реєстрації / авторизації через GitHub [7], GitLab, додавання / редагування / видалення вакансії.

Таблиця 1.1 – Порівняння програмних аналогів з виконуваною розробкою

Платформа	Robota.ua	Work.ua	Jobs.ua	Trud.ua	YettiJob
Категорія сайту	Пошук роботи	Пошук роботи	Пошук роботи	Пошук роботи	Пошук роботи
Реєстрація користувачів	+	+	—	—	+
Первинне тестування кандидатів	—	—	—	—	+
Наявність рекламних оголошень	+	+	+	+	—
Реєстрація / авторизація через GitHub, GitLab	—	—	—	—	+
Платні функції	+	+	+	+	—
Додавання / редагування / видалення вакансії	+	+	+	+	+

1.4 Висновки за розділом 1

Перший розділ направлений на аналіз предметної області в питанні пошуку та публікації вакансій для працевлаштування. Було детально розглянуто кілька платформ для працевлаштування, а саме: Robota.ua, Jobs.ua, Work.ua, Trud.ua. Докладно досліджено плюси та мінуси кожного вебресурсу. Дані українські сайти не об'єднані на єдиній платформі. Така роздробленість ринку праці ускладнює завдання кандидатів на пошуки роботи та роботодавців, які хочуть знайти відповідних працівників.

Проведений аналіз продемонстрував, що частина цих вебзастосунків має деякі недоліки, а саме:

- відсутнє первинне тестування кандидатів на вакансії;
- відсутня реєстрація / авторизація через GitHub, GitLab;
- ефективна робота можлива лише у разі придбання платних функцій;
- досить багато додаткових функцій, розділів, а це робить ресурс досить громоздким;
- велика кількість рекламного контенту.

Завданнями дипломної роботи є:

- аналіз предметної області;
- проєктування програми;
- розробка програми;
- тестування програми.

2 ПРОЄКТУВАННЯ ПРОГРАМИ

2.1 Вибір засобів для проєктування і реалізації

Ключовим питанням, що вирішувалось у даному підрозділі, було питання вибору мови програмування. Враховуючи необхідність веброзробки, було використано три альтернативи, які включали мови Python3, PHP [8] і Go.

Проведене порівняння (табл. 2.1) вказало на те, що три мови є достатньо якісними для розробки цієї програми, проте обрано було мову Python3 [9]. Python3 є однією з найбільш популярних мов програмування у світі, а також однією з найбільш підходящих для роботи з широким спектром завдань і проєктів. Вона має потужні фреймворки, такі як Django [10] і Flask, які допомагають розробляти високоякісні вебдодатки. Дана мова є потужним інструментом для автоматизації рутинних завдань, таких як обробка тексту, збір даних та інші. Крім того, Python3 має простий і легкий у вивченні синтаксис, що робить його адекватним вибором для початківців і експертів у програмуванні. Всі ці фактори роблять Python3 вибором для роботи з проєктом, що розробляється.

Таблиця 2.1 – Порівняння мов програмування для створення ПЗ

Критерій	Python3	PHP	Go
1	2	3	4
Тип мови	Мова загального призначення	Мова для веброзробки	Мова загального призначення
Підтримка Unicode	+	+/-	+
Підтримка об'єктно-орієнтованого програмування	+	+/-	+/-
Підтримка функціонального програмування	+	+/-	+/-

Кінець таблиці 2.1

1	2	3	4
Підтримка асинхронного програмування	+	+/-	+
Наявність фреймворків для створення вебзастосунків	+	+	+
Наявність бібліотек для роботи з зображеннями	+	+/-	-
Функції першого класу	+	+	-
Анонімні функції	+	+	-
Замикання	+	+	-
Garbage Collector	+	+/-	+

Нижче наведена таблиця 2.2, що порівнює основні характеристики та особливості фреймворків Django [11] та Flask.

Таблиця 2.2 – Порівняння фреймворків Django та Flask

Критерій	Django	Flask
Мова програмування	Python	Python
Безпека	+	+/-
Адміністративний інтерфейс	+	-
Роутінг	+	+/-
ORM	+	-
Кешування	+	-

Вибір зроблено на користь вебфреймворку Django.

2.2 Визначення функціональних вимог до програми

Повний перелік функціональних вимог [12] до програми включає:

- реєстрація;
- вибір ролі на сайті;
- реєстрація за допомогою сторонніх вебдодатків;
- авторизація;
- редагування власного кабінету;
- видалення акаунту;
- додавання зображення аккаунта;
- додавання освіти користувачем;
- видалення освіти користувачем;
- редагування освіти користувачем;
- додавання мов користувачем;
- редагування мов користувачем;
- видалення мов користувачем;
- додавання сертифікатів користувачем;
- редагування сертифікатів користувачем;
- видалення сертифікатів користувачем;
- додавання досвіду роботи користувачем;
- редагування досвіду роботи користувачем;
- видалення досвіду роботи користувачем;
- додавання компанії роботодавцем;
- редагування компанії роботодавцем;
- видалення компанії роботодавцем;
- додавання логотипу компанії роботодавцем;
- додавання відеопрезентації компанії роботодавцем;
- додавання вакансії роботодавцем;
- редагування вакансії роботодавцем;
- видалення вакансії роботодавцем;

- перегляд вакансії користувачем;
- додавання резюме користувачем;
- редагування резюме користувачем;
- видалення резюме користувачем;
- перегляд резюме роботодавцем;
- пошук вакансій користувачем;
- пошук резюме роботодавцем;
- перегляд відеопрезентації користувачем;
- створення тесту для вакансії роботодавцем;
- редагування тесту для вакансії роботодавцем;
- видалення тесту для вакансії роботодавцем;
- створення питань для тесту роботодавцем;
- редагування питань для тесту роботодавцем;
- видалення питань для тесту роботодавцем;
- перегляд результатів тестування користувачів роботодавцем;
- проходження користувачем тестування;
- додавання резюме в «Список бажаного» роботодавцем;
- перегляд резюме у «Списку бажаного» роботодавцем;
- видалення резюме зі «Списку бажаного» роботодавцем;
- додавання вакансії в «Список бажаного» користувачем;
- перегляд вакансії у «Списку бажаного» користувачем;
- видалення вакансії зі «Списку бажаного» користувачем;
- створення кімнат роботодавцем;
- видалення кімнат роботодавцем;
- редагування кімнат роботодавцем;
- занесення користувачів до кімнат роботодавцем;
- перегляд кімнат користувачем.

На основі цих вимог було визначено перелік сценаріїв роботи для роботодавця (рис. 2.1).

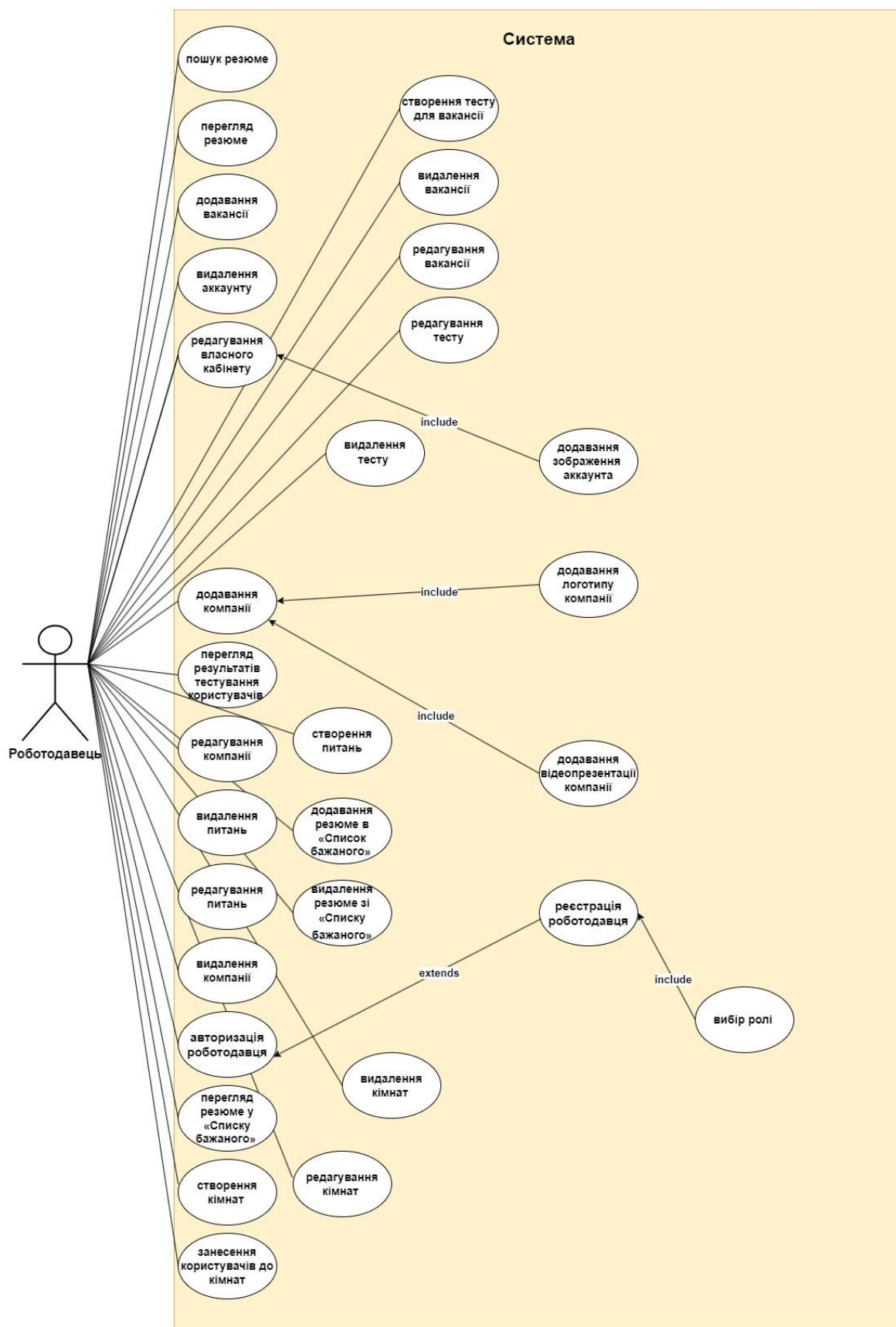


Рисунок 2.1 – Діаграма прецедентів роботодавця

Також на основі вищевказаних функціональних вимог було визначено перелік сценаріїв роботи для користувача (рис. 2.2).

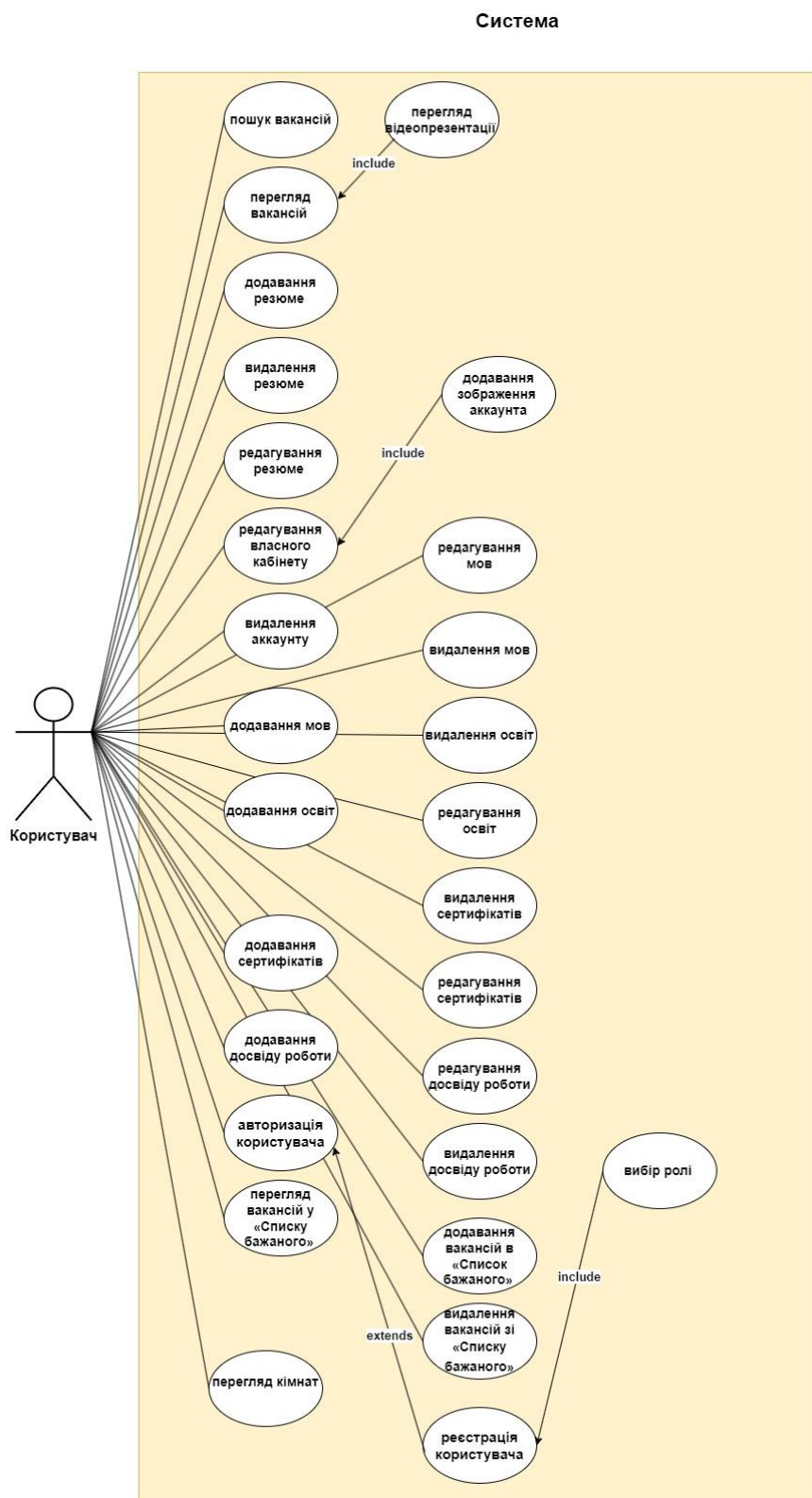


Рисунок 2.2 – Діаграма прецедентів користувача

2.3 Вибір та обґрунтування системи керування базою даних

PostgreSQL – це вільно поширювана система керування базою даних (СКБД), яка відома своєю стабільністю, надійністю, функціональністю та масштабованістю. PostgreSQL є однією з найбільш популярних реляційних СКБД в світі [13].

Також PostgreSQL знаходить своє використання в багатьох галузях, таких як наука, освіта, фінанси, телекомунікації, логістика, медицина та інші.

Причини вибору PostgreSQL для дипломної кваліфікаційної роботи бакалавра:

- надійність: PostgreSQL відомий своєю надійністю і стабільністю. Він використовує транзакційну модель, що дозволяє забезпечувати цілісність даних та уникнення втрати даних у разі відмови системи;
- функціональність: PostgreSQL має багато функціональностей, що дозволяє ефективно керувати БД. PostgreSQL підтримує різні типи даних, включаючи географічні дані, що робить його ідеальним вибором для розробки проєктів, пов'язаних з геолокацією. PostgreSQL також має багато вбудованих функцій, таких як тригери, процедури та виклики функцій, що дозволяє створювати більш складні та потужні запити до БД;
- розширюваність: PostgreSQL відповідає розширюваності та масштабованості, що дозволяє легко розширювати проєкт, додавати нові функції та збільшувати об'єм даних;
- безпека: PostgreSQL має вбудовані механізми безпеки, такі як автентифікація користувачів, шифрування даних та обмеження доступу до даних. Це дозволяє зберігати дані в безпечності та забезпечувати конфіденційність [14].

Нижче наведена таблиця 2.3, яка порівнює основні характеристики трьох популярних реляційних БД: MySQL, SQLite та PostgreSQL.

Таблиця 2.3 – Порівняння реляційних БД для створення ПЗ

Характеристика	MySQL	SQLite	PostgreSQL
Тип ліцензії	Відкрита	Вільна	Відкрита
Тип БД	Клієнт-сервер	Вбудована	Клієнт-сервер
Транзакції	+/-	+/-	+
Розмір БД	До 32 Тб	До 140 Тб	Необмежений
Підтримка тригерів	+	+/-	+
Підтримка процедур	+	—	+
Повнотекстовий пошук	+	+	+
Підтримка реплікації	+	—	+
Підтримка JSON	+/-	+	+
Підтримка масивів	—	—	+

2.4 Проєктування БД

Для створення БД було використано СКБД PostgreSQL та створено таблиці за основними сутностями.

Таблиця Category включає поля:

- id (integer) – первинний ключ;
- title (varchar) – назва категорії;
- slug (varchar) – унікальний рядок ідентифікатор, який з'являється в uniform resource locator (URL) адресі сайту;
- published (boolean) – прапорець, що вказує, чи опубліковано катигорії;
- created (datetime) – дата створення катигорії;

- updated (datetime) – дата оновлення категорії;
- about (text) – додаткова інформація;
- user_id (bigint) – автор запису (зв'язок один-до-багатьох з «User»).

Таблиця Technology включає поля:

- id (integer) – первинний ключ;
- title (varchar) – назва технології;
- slug (varchar) – унікальний рядок ідентифікатор, який з'являється в

URL;

- published (boolean) – прапорець, що вказує, чи опубліковано технології;

- created (datetime) – дата створення технології;
- updated (datetime) – дата оновлення технології;
- about (text) – додаткова інформація;
- user_id (bigint) – автор запису (зв'язок один-до-багатьох з «User»).

Таблиця Advert включає поля:

- id (integer) – первинний ключ;
- slug (varchar) – унікальний рядок ідентифікатор, який з'являється в

URL-адресі сайту;

- published (boolean) – прапорець, що вказує, чи опубліковано оголошення;

- created (datetime) – дата створення оголошення;
- updated (datetime) – дата оновлення оголошення;
- about (text) – додаткова інформація;
- salary (decimal) – заробітня платня;
- currency (varchar) – тип валюти;
- type (varchar) – тип зайнятості;
- title (varchar) – назва оголошення;
- description (varchar) – докладний опис оголошення;
- advert_type (varchar) – тип оголошення;

- category_id (bigint) – категорія оголошення (зв'язок один-до-багатьох з таблицею «Category»);
- user_id (bigint) – автор оголошення (зв'язок один-до-багатьох з таблицею «User»).

Таблиця Wishlist включає поля:

- id (integer) – первинний ключ;
- user_id (bigint) – автор запису (зв'язок один-до-багатьох з таблицею «User»).

Таблиця Room включає поля:

- id (integer) – первинний ключ;
- title (varchar) – назва кімнати;
- user_id (bigint) – автор кімнати (зв'язок один-до-багатьох з таблицею «User»).

Таблиця Quiz включає поля:

- id (integer) – первинний ключ;
- title (varchar) – назва тесту;
- description (varchar) – докладний опис тесту;
- number_of_questions (integer) – кількість питань в тесті;
- time (integer) – обмеження в часі для проходження тестування, в секундах;
- user_id (bigint) – автор тесту (зв'язок один-до-багатьох з таблицею «User»);
- advert_id (bigint) – оголошення, для якого створено тестування (зв'язок один-до-багатьох з таблицею «Advert»).

Таблиця Question включає поля:

- id (integer) – первинний ключ;
- content (varchar) – дані про питання;
- quiz_id (bigint) – тест, для якого створено питання (зв'язок один-до-багатьох з таблицею «Quiz»).

Таблиця Answer включає поля:

- id (integer) – первинний ключ;
- content (varchar) – дані про відповідь;
- correct (boolean) – прапорець, що вказує, чи правильна відповідь;
- question_id (bigint) – питання, для якого створено відповідь (зв'язок один-до-багатьох з таблицею «Question»).

Таблиця MarksOfUser включає поля:

- id (integer) – первинний ключ;
- score (real) – оцінка за проходження тестування;
- quiz_id (bigint) – тест, для якого виставлено оцінку (зв'язок один-до-багатьох з таблицею «Quiz»);
- user_id (bigint) – користувач, який пройшов тестування (зв'язок один-до-багатьох з таблицею «User»).

Таблиця Language включає поля :

- id (integer) – первинний ключ;
- published (boolean) – прапорець, що вказує, чи опубліковано запис;
- created (datetime) – дата створення запису;
- updated (datetime) – дата оновлення запису;
- about (text) – додаткова інформація;
- locale (varchar) – вибір мови, якою володіє користувач;
- level (varchar) – рівень володіння мовою;
- certificate (varchar) – назва сертифікату, якщо він є;
- interview (boolean) – прапорець, що вказує, чи може користувач проходити співбесіду цією мовою;
- user_id (bigint) – автор запису (зв'язок один-до-багатьох з таблицею «User»).

Таблиця Course включає поля :

- id (integer) – первинний ключ;
- published (boolean) – прапорець, що вказує, чи опубліковано запис;
- created (datetime) – дата створення запису;
- updated (datetime) – дата оновлення запису;

- about (text) – додаткова інформація;
- course (varchar) – назва курсу;
- completed (date) – дата закінчення курсу;
- user_id (bigint) – автор запису (зв'язок один-до багатьох з таблицею «User»).

Таблиця Education:

- id (integer) – первинний ключ;
- published (boolean) – прапорець, що вказує, чи опубліковано запис;
- created (datetime) – дата створення запису;
- updated (datetime) – дата оновлення запису;
- about (text) – додаткова інформація;
- institution (varchar) – назва навчального закладу;
- specialty (varchar) – назва спеціальності;
- completed (date) – дата закінчення навчального закладу;
- user_id (bigint) – автор запису (зв'язок один-до багатьох з таблицею «User»).

Таблиця WorkExperience включає поля:

- id (integer) – первинний ключ;
- published (boolean) – прапорець, що вказує, чи опубліковано запис;
- created (datetime) – дата створення запису;
- updated (datetime) – дата оновлення запису;
- about (text) – додаткова інформація;
- position (varchar) – посада користувача;
- company (varchar) – назва компанії;
- started (date) – початок робочої діяльності;
- finished (date) – кінець робочої діяльності;
- user_id (bigint) – автор запису (зв'язок один-до багатьох з таблицею «User»).

Таблиця Company включає поля:

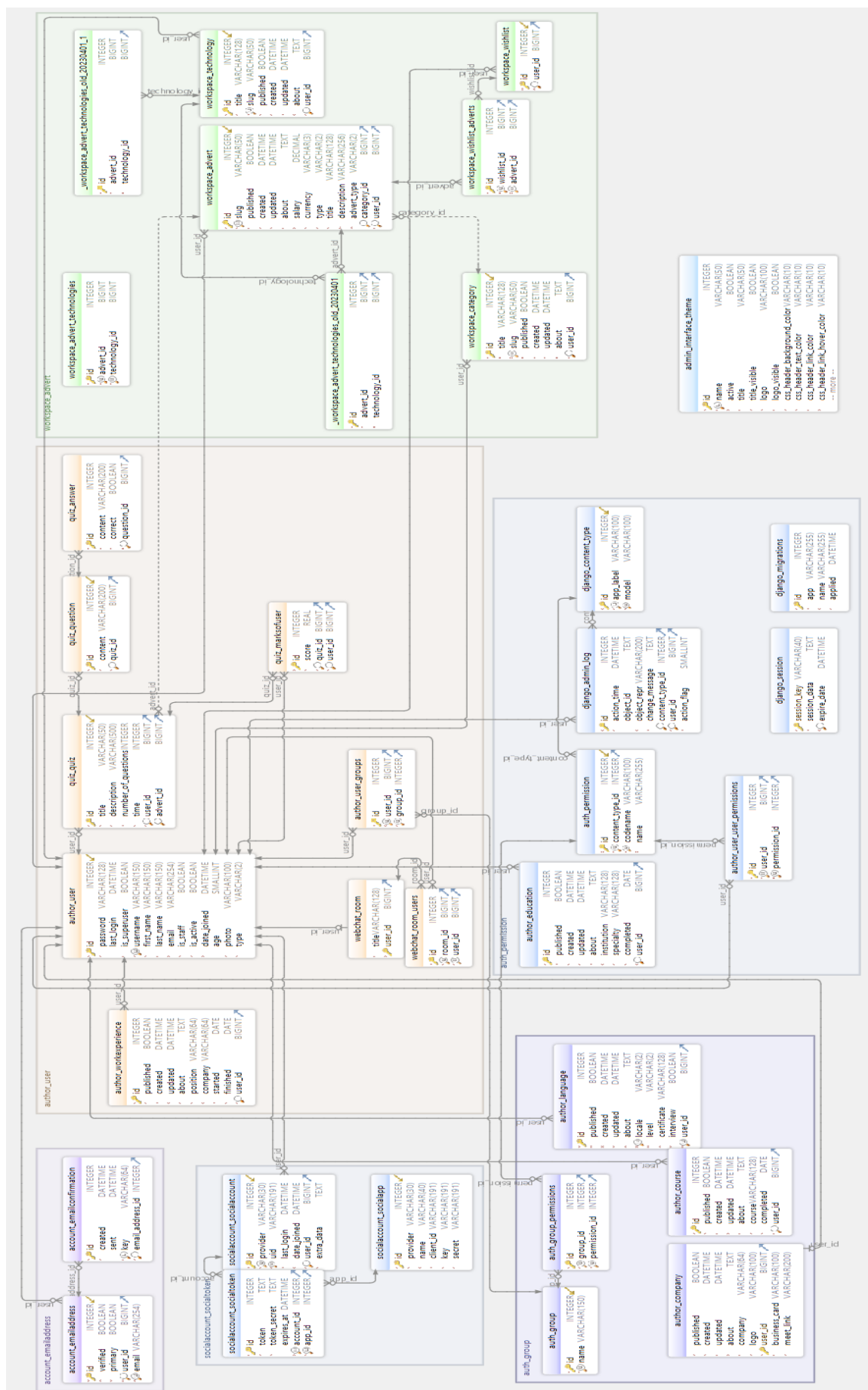
- id (integer) – первинний ключ;

- published (boolean) – прапорець, що вказує, чи опубліковано запис;
- created (datetime) – дата створення запису;
- updated (datetime) – дата оновлення запису;
- about (text) – додаткова інформація;
- company (varchar) – назва компанії;
- logo (varchar) – адреса до файлу логотипа компанії на сервері;
- user_id (bigint) – автор запису (зв'язок один-до багатьох з таблицею «User»).
- business_card (varchar) – адреса до файлу відеопрезентації компанії на сервері;
- meet_link (varchar) – посилання на відеоконференцію.

Таблиця User включає поля:

- id (integer) – первинний ключ;
- password (varchar) – пароль користувача;
- last_login (datetime) – останній візит на сайті;
- is_superuser (boolean) – прапорець, чи є користувач адміністратором;
- username (varchar) – ім'я користувача на сайті (псевдонім);
- first_name (varchar) – ім'я користувача;
- last_name (varchar) – прізвище користувача;
- email (varchar) – електронна пошта користувача;
- is_staff (boolean) – прапорець, чи є користувач персоналом сайту;
- is_active (boolean) – прапорець, чи активований акаунт;
- age (smallint) – вік користувача;
- photo (varchar) – фото профіля користувача;
- type (varchar) – роль користувача на сайті.

Схема БД представлена нижче (рис. 2.3).



2.5 Визначення структури вебсистеми

Django дотримується шаблону Model-View-Controller (MVC) [15], але зберігає власні угоди. Немає окремого контролера, і повний додаток заснований на представленні моделі та шаблоні – це називається додатком Model-View-Template (MVT) (рис. 2.4).

Шаблон проєктування MVT – це архітектурний підхід до розробки ПЗ, що використовується фреймворком Django [16]. Розділяє проєкт на три основних компоненти: модель (Model), представлення (View) та шаблон (Template). Архітектура MVT перетворює розробку складних додатків на набагато більш керований процес. Це дозволяє кільком розробникам одночасно працювати над додатком.

Модель в MVT відповідає за управління даними додатку, які зберігаються та обробляються. Модель містить бізнес-логіку та функціональність, пов'язану з даними. Вона забезпечує доступ до даних та виконання всіх операцій з ними.

Представлення в MVT відображає дані, які зберігаються в моделі, на екрані користувача. Вона не залежить від інших компонентів додатку та має власний набір даних, не пов'язаних з моделлю. Представлення може відображатися у вигляді графічного інтерфейсу користувача або в форматі сторінки вебсайту.

Шаблон в MVT – це рівень уявлення, який повністю обробляє частину інтерфейсу користувача. Подання використовується для виконання бізнес-логіки та взаємодії з моделлю для перенесення даних та відображення шаблону.

Шаблон проєктування MVT дозволяє забезпечити розподілення відповідальності між компонентами додатку, що сприяє зниженню залежності між ними та полегшує розвиток та підтримку ПЗ. Крім того, він забезпечує легку зміну та модифікацію додатку в майбутньому, так як різні компоненти можуть бути змінені незалежно один від одного.



Рисунок 2.4 – Шаблон проєктування MVT

2.6 Структура вебсистеми

При проєктуванні проєкту було використано фреймворк Django, що визначило основну структуру застосунку (рис. 2.5).

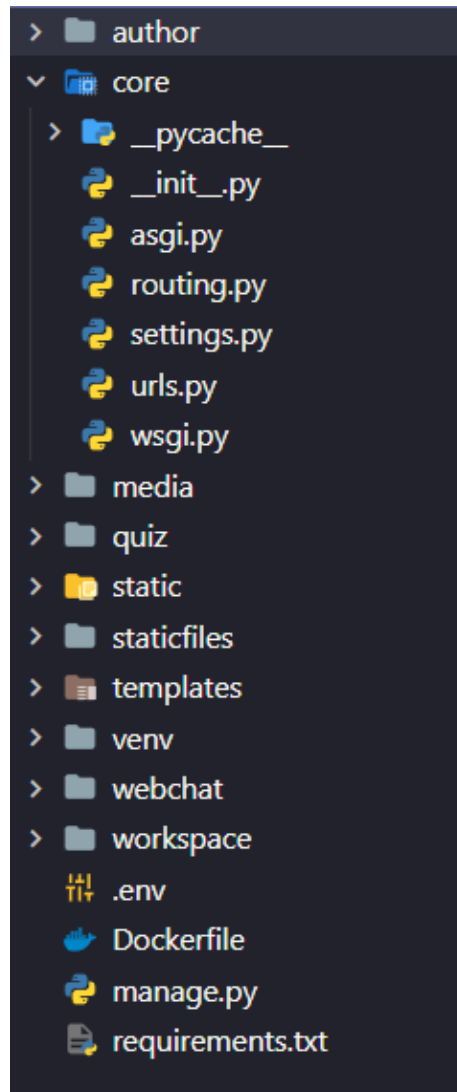


Рисунок 2.5 – Структура проєкту

Структура проєкту складається з:

- requirements.txt – файл з усіма залежностями;
- manage.py – скрипт для керування додатком;
- Dockerfile – скрипт для побудови образу;
- .env – змінні оточення;
- workspace – окремий пакет для роботи з оголошеннями;
- webchat – окремий пакет для роботи з кімнатами;
- venv – віртуальне оточення Python;
- templates – глобальні шаблони проєкту;
- staticfiles – папка для збирання всіх статичних файлів;

- static – глобальні статичні файли проєкту;
- quiz – окремий пакет для тестування;
- media – папка для користувацьких файлів;
- core – пакет з серцевим кодом проєкту;
- core / settings.py – файл з налаштуваннями;
- core / urls.py – файл для роутінгу проєкту;
- author – окремий пакет для роботи з користувачами.

2.7 Висновки за розділом 2

Під час проєктування та розробки даного проєкту було використано низку потужних інструментів та технологій, що дозволило досягти оптимального результату. Важливою складовою проєкту стала мова програмування Python. Python – кросплатформна та мультинаправлена мова. Вона підходить для величезної кількості цілей в веброботі та в обробці зображень. Для додатків Python практично немає обмежень. Мова дозволяє створювати ефективний та легко зрозумілий код [17]-[19], має широкий вибір бібліотек та надає можливість тестувати певні частини коду окремо.

Крім того, був використаний вебфреймворк Django [20], який дозволяє швидко та зручно розробляти вебдодатки. Для покращення структури проєкту використовувався шаблон проєктування MVT, що забезпечує чітке розділення функціональних компонентів системи.

Також було створено діаграми прецедентів для користувача та роботодавця, що дозволило краще зрозуміти функціональні вимоги до системи та її компонентів. Для забезпечення ефективної роботи системи була розроблена та реалізована БД з урахуванням всіх вимог до її структури та функціональності.

У даному розділі показана структура проєкту, що розробляється. Загалом, завдяки використанню потужних інструментів та технологій, а також чіткому проєктуванню та розробці, проєкт був успішно реалізований та готовий до використання.

3 РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програми

Порядок роботи з функціями програми користувача та роботодавця показаний на рис. 3.1.

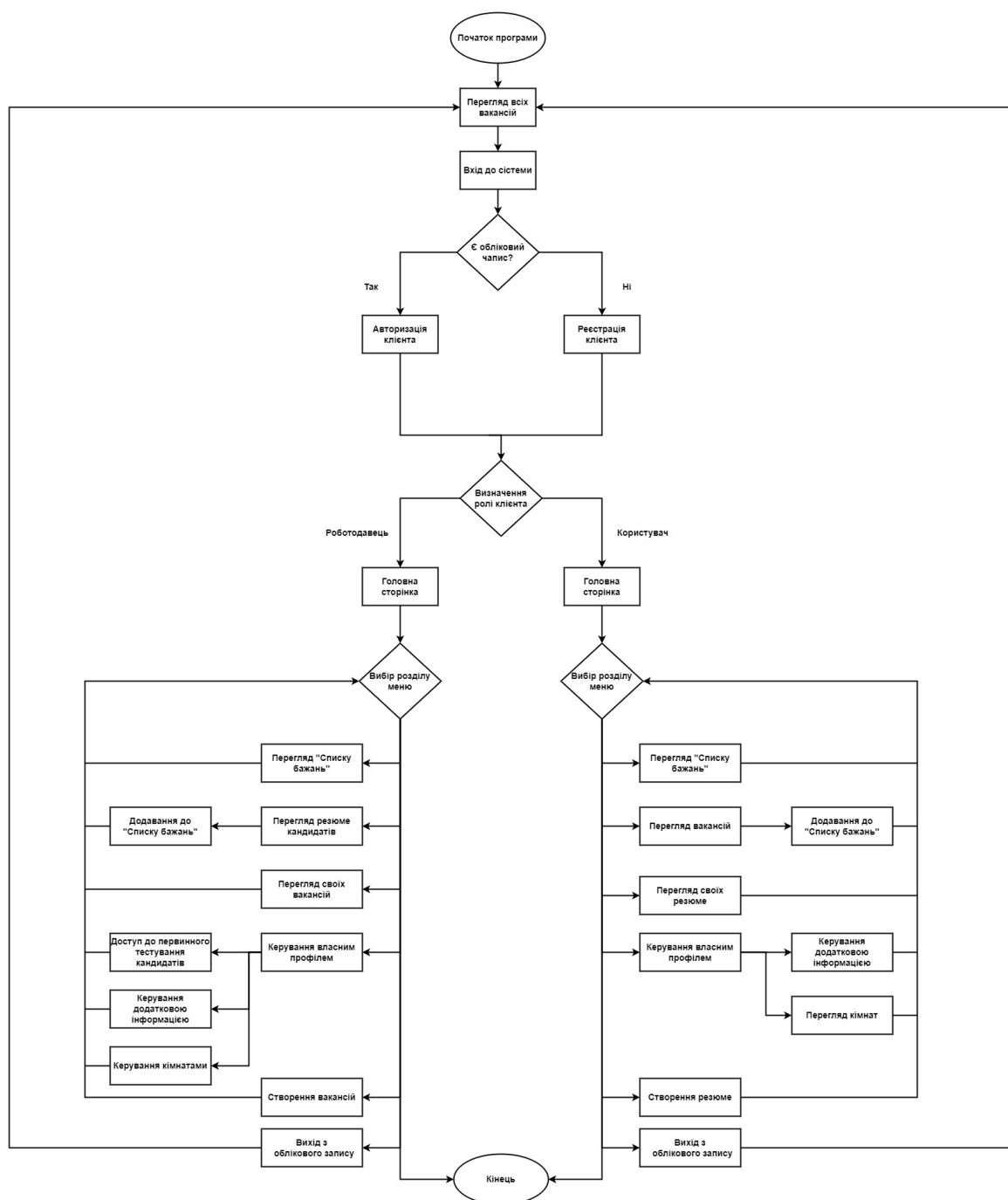


Рисунок 3.1 – Схема функціонування програми

3.2 Опис реалізованих класів моделей

Клас Quiz є підкласом класу Model з фреймворку Django, містить поля, що відображають властивості самого тесту: title (назва тесту), description (опис), user (зовнішній ключ, що посилається на користувача, який створив тест), advert (зовнішній ключ, що посилається на оголошення, пов'язане з тестом), number_of_questions (кількість питань у тесті) та time (тривалість тесту в секундах). Метод get_questions() повертає всі питання, які належать до даного тесту.

Клас Question є підкласом класу Model з фреймворку Django, містить поля, що відображають питання, які включені до тесту: content (вміст питання) та quiz (зовнішній ключ, що посилається на тест, до якого відноситься це питання). Метод get_answers() повертає всі відповіді, які належать до даного питання.

Клас Answer є підкласом класу Model з фреймворку Django, містить поля, що відображають відповіді на питання у тесті: content (вміст відповіді), correct (чи є відповідь правильною) та question (зовнішній ключ, що посилається на питання, до якого відноситься ця відповідь).

Клас MarksOfUser є підкласом класу Model з фреймворку Django, містить поля, що відображають результати користувача у тесті: quiz (зовнішній ключ, що посилається на тест), user (зовнішній ключ, що посилається на користувача, що проходив тест) та score (оцінка, яку користувач отримав за тест).

Абстрактний клас AbstractAdvert є підкласом класу Model з фреймворку Django, містить спільні поля для всіх спадкоємців, такі як заголовок (title), унікальна URL-адреса (slug), автор (user), дата створення (created) та дата останнього оновлення (updated). Цей клас також визначає методи __str__() для рядкового представлення та save, що генерує унікальну URL-адресу (slug) при створенні нового об'єкта.

Класи Category та Technology успадковуються від AbstractAdvert та визначають категорії та технології.

Клас `BaseAdvert` успадковується `AbstractAdvert`, але визначає додаткові поля для конкретних типів оголошень. Ці поля включають зарплату (`salary`), валюту (`currency`), категорію (`category`), технології (`technologies`) та тип зайнятості (`type`). `BaseAdvert` є абстрактним класом, тому що він не представляє конкретне оголошення, а служить лише як базовий клас для інших класів, що успадковують його.

Клас `Advert` успадковується від `BaseAdvert` та визначає конкретне оголошення. Він додає поля для заголовка (`title`), опису (`description`) та типу оголошення (`advert_type`), який може бути або вакансією (`VACANCY`), або резюме (`RESUME`). Має також наступні методи: `get_absolute_url()` повертає URL для перегляду детальної інформації про оголошення, `is_resume()` повертає `True`, якщо оголошення є резюме, `is_vacancy()` повертає `True`, якщо оголошення є вакансією.

Клас `ResumeAdvert` є підкласом моделі `Advert` і є проксі-моделлю, що означає, що він не створює нову таблицю в БД, а замість цього просто дозволяє отримати доступ до даних з базової моделі `Advert`. Цей клас має метод `save()`, який перевіряє, чи не існує вже ідентифікатор для цієї моделі (тобто, чи не зберігається вона вже в БД), і якщо цього не відбувається, то задає значення поля `advert_type` на Резюме.

Клас `VacancyAdvert` є підкласом моделі `Advert` і є проксі-моделлю. Цей клас має метод `save()`, який перевіряє, чи не існує вже ідентифікатор для цієї моделі (тобто, чи не зберігається вона вже в БД), і якщо цього не відбувається, то задає значення поля `advert_type` на Вакансію.

Клас `Wishlist` є підкласом класу `Model` з фреймворку `Django`, який представляє список бажань користувачів. Він містить два поля: автор (`user`), яке зберігає посилання на користувача, який створив цей список, та оголошення (`adverts`), яке містить відношення багато-до-багатьох з моделлю `Advert`, що зберігає всі оголошення, які додані до цього списку бажань.

Клас `Room` є підкласом класу `Model` з фреймворку `Django`, містить наступні поля, як заголовок (`title`), автор (`user`), яке зберігає інформацію про

користувача, який створив кімнату, користувачі (users), яке зберігає інформацію про користувачів, які знаходяться в даній кімнаті. Цей клас має метод `__str__()` визначений для того, щоб повернути рядок, який буде використовуватися для представлення об'єкта.

Клас моделі `User` є підкласом вбудованого класу моделей Django `AbstractUser`, що відповідає за користувачів системи. Цей клас містить наступні поля: вік (age), логотип (photo), роль (type). Цей клас має методи: `is_recruiter()` повертає `True`, якщо користувач має роль Рекрутера, інакше – `False`, `is_applicant()` повертає `True`, якщо користувач має роль Претендента, інакше – `False`, `get_absolute_url()`, який повертає URL-адресу профілю користувача для використання у шаблонах Django.

Клас `Course` є підкласом абстрактного класу `AbstractAdvert`, що визначає загальні поля для всіх оголошень. Ця модель містить наступні поля: курс (course) назву курсу, дата закінчення (completed). Цей клас має метод `__str__()` визначений для того, щоб повернути рядок, який буде використовуватися для представлення об'єкта.

Клас `Education` є підкласом абстрактного класу `AbstractAdvert`, що визначає загальні поля для всіх оголошень. Ця модель містить наступні поля: навчальний заклад (institution), спеціальність (specialty), дата закінчення (completed). Цей клас має метод `__str__()` визначений для того, щоб повернути рядок, який буде використовуватися для представлення об'єкта.

Клас `Language` є підкласом абстрактного класу `AbstractAdvert`, що визначає загальні поля для всіх оголошень. Ця модель містить наступні поля: мова (locale), рівень володіння (level), сертифікат (certificate) та проходження співбесіди (interview). Цей клас має метод `__str__()` визначений для того, щоб повернути рядок, який буде використовуватися для представлення об'єкта.

Клас `WorkExperience` є підкласом абстрактного класу `AbstractAdvert`, що визначає загальні поля для всіх оголошень. Ця модель містить наступні поля: посада (position), компанія (company), початок діяльності (started), кінець

діяльності (`finished`). Цей клас має метод `__str__()` визначений для того, щоб повернути рядок, який буде використовуватися для представлення об'єкта.

Клас `Company` є підкласом абстрактного класу `AbstractAdvert`, що визначає загальні поля для всіх оголошень. Ця модель містить наступні поля: назва компанії (`company`), автор (`user`), логотип (`logo`), посилання на відеоконференцію (`meet_link`), відеопрезентація (`business_card`). Цей клас має метод `__str__()` визначений для того, щоб повернути рядок, який буде використовуватися для представлення об'єкта.

3.3 Опис реалізованих класів представлень

Клас представлення `About` є підкласом вбудованого у Django класу `TemplateView`. Цей клас містить властивість `template_name`, що визначає ім'я шаблону, який буде використовуватись для відображення сторінки.

Клас представлення `AdvertDetailView` є підкласом вбудованого у Django класу `DetailView`. Цей клас містить властивості: `model`, що вказує на модель, з якою пов'язане це представлення, `query_pk_and_slug` вказує, що запит до представлення буде містити як первинний ключ (`pk`), так і слаг (`slug`), `template_name` визначає ім'я шаблону, який буде використовуватись для відображення сторінки.

Клас представлення `ResumeCreateView` є підкласом вбудованого у Django класу `CreateView`. Цей клас містить властивості: `model`, що вказує на модель, з якою пов'язане це представлення, `template_name` визначає ім'я шаблону, який буде використовуватись для відображення сторінки, `form_class` вказує на клас форми, який буде використовуватись для створення резюме. Цей клас має перевизначений метод `form_valid()` для додаткової обробки форми перед її збереженням.

Клас представлення `ResumeUpdateView` є підкласом вбудованого у Django класу `UpdateView`. Цей клас містить властивості: `model`, що вказує на модель, з якою пов'язане це представлення, `template_name` визначає ім'я шаблону, який буде

використовуватись для відображення сторінки, `form_class` вказує на клас форми, який буде використовуватись для оновлення резюме, `query_pk_and_slug` вказує, що запит до представлення буде містити як первинний ключ (`pk`), так і слаг (`slug`). Цей клас має перевизначений метод `get_queryset()` для налаштування запиту до БД для отримання об'єктів моделі, які належать поточному користувачеві.

Клас представлення `ResumeDeleteView` є підкласом вбудованого у Django класу `DeleteView`, а також використовує `LoginRequiredMixin`, щоб переконатися, що користувач увійшов у систему перед доступом до цього представлення. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `template_name` визначає ім'я шаблону, який буде використовуватись для відображення сторінки, `form_class` вказує на клас форми, який буде використовуватись для видалення резюме, `query_pk_and_slug` вказує, що запит до представлення буде містити як первинний ключ (`pk`), так і слаг (`slug`), `success_url` визначає URL, на який буде перенаправлено після успішного видалення об'єкта, `permission_required` визначає необхідне дозволення. Цей клас має перевизначений метод `get_queryset()` для налаштування запиту до БД для отримання об'єктів моделі, які належать поточному користувачеві.

Клас представлення `VacancyCreateView` є підкласом вбудованого у Django класу `CreateView`, а також використовує `LoginRequiredMixin`, щоб переконатися, що користувач увійшов у систему перед доступом до цього представлення. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `template_name` визначає ім'я шаблону, який буде використовуватись для відображення сторінки, `form_class` вказує на клас форми, який буде використовуватись для створення вакансії, `permission_required` визначає необхідний дозвіл. Цей клас має перевизначений метод `form_valid()` для додаткової обробки форми перед її збереженням.

Клас представлення `HomeView` є підкласом вбудованого у Django класу `ListView`. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `paginate_by` визначає кількість об'єктів моделі, які будуть відображені на кожній сторінці списку, `paginate_orphans` визначає кількість

об'єктів моделі, які можуть залишитись на останній сторінці, `template_name` визначає ім'я шаблону, який буде використовуватись для відображення сторінки, `permission_required` визначає необхідний дозвіл. Цей клас має перевизначений метод `get_queryset()` для налаштування запиту до БД та отримання списку оголошень.

Клас представлення `QuizUpdateView` є підкласом вбудованого у Django класу `UpdateView`, а також використовує `LoginRequiredMixin`, щоб переконатися, що користувач увійшов у систему перед доступом до цього представлення. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `pk_url_kwarg` визначає ім'я параметра URL, що містить первинний ключ (pk) об'єкта, який потрібно оновити, `template_name` визначає ім'я шаблону, який буде використовуватись для відображення сторінки, `success_url` визначає URL-адресу, на яку буде перенаправлено користувача після успішного оновлення, `form_class` вказує на клас форми, який буде використовуватись для оновлення тесту. Цей клас має перевизначений метод `get_queryset()` для налаштування запиту до БД та отримання тесту, який потрібно оновити.

Клас представлення `QuizCreateView` є підкласом вбудованого у Django класу `CreateView`. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `template_name` визначає ім'я шаблону, який буде використовуватись для відображення сторінки, `success_url` визначає URL-адресу, на яку буде перенаправлено користувача після успішного створення, `form_class` вказує на клас форми, який буде використовуватись для створення тесту. Цей клас має перевизначений метод `form_valid()` для налаштування додаткових атрибутів об'єкта тесту перед збереженням у БД.

Клас представлення `QuizDeleteView` є підкласом вбудованого у Django класу `DeleteView`, а також використовує `LoginRequiredMixin`, щоб переконатися, що користувач увійшов у систему перед доступом до цього представлення. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `pk_url_kwarg` визначає ім'я параметра URL, що містить первинний ключ (pk) об'єкта, який потрібно видалити, `success_url` визначає URL-адресу, на

яку буде перенаправлено користувача після успішного видалення тесту, `permission_required` визначає необхідний дозвіл. Цей клас має перевизначений метод `get_queryset()` для налаштування запиту до БД для отримання тесту, який потрібно видалити.

Клас представлення `QuestionCreateView` є підкласом вбудованого у Django класу `CreateView`, а також використовує `LoginRequiredMixin`, щоб переконатися, що користувач увійшов у систему перед доступом до цього представлення. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `success_url` визначає URL-адресу, на яку буде перенаправлено користувача після успішного створення питання, `permission_required` визначає необхідний дозвіл, `form_class` вказує на клас форми, який буде використовуватись для створення питання. Цей клас має перевизначені методи: `get_context_data()` для отримання додаткових даних, `form_valid()` перевизначений для налаштування додаткових атрибутів об'єкта питання перед збереженням у БД.

Клас представлення `UserUpdateView` є підкласом вбудованого у Django класу `UpdateView`, а також використовує `LoginRequiredMixin`, щоб переконатися, що користувач увійшов у систему перед доступом до цього представлення. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `success_url` визначає URL-адресу, на яку буде перенаправлено користувача після успішного оновлення профілю, `form_class` вказує на клас форми, який буде використовуватись для оновлення профілю. Цей клас має перевизначений метод `get_queryset()` для отримання запиту до БД, який повертає об'єкт користувача, який потрібно оновити.

Клас представлення `UserDeleteView` є підкласом вбудованого у Django класу `DeleteView`, а також використовує `LoginRequiredMixin`, щоб переконатися, що користувач увійшов у систему перед доступом до цього представлення. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `form_class` вказує на клас форми, який буде використовуватись для видалення профілю. Цей клас має перевизначені методи: `get_queryset()` для отримання запиту до БД, який повертає об'єкт користувача, який потрібно

видалити, `get_success_url()` перевизначений для визначення URL-адреси, на яку буде перенаправлено користувача після успішного видалення профілю.

Клас представлення `CompanyCreateView` є підкласом вбудованого у Django класу `CreateView`, а також використовує `LoginRequiredMixin`, щоб переконатися, що користувач увійшов у систему перед доступом до цього представлення. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `template_name` визначає ім'я шаблону, яке буде використовуватись для відображення сторінки, `form_class` вказує на клас форми, який буде використовуватись для створення компанії, `permission_required` визначає необхідний дозвіл. Цей клас має перевизначений метод `form_valid()` для додаткової обробки форми перед її збереженням.

Клас представлення `CompanyUpdateView` є підкласом вбудованого у Django класу `UpdateView`, а також використовує `LoginRequiredMixin`, щоб переконатися, що користувач увійшов у систему перед доступом до цього представлення. Цей клас містить властивості: `model` вказує на модель, з якою пов'язане це представлення, `success_url` визначає URL-адресу, на яку буде перенаправлено користувача після успішного оновлення компанії, `form_class` вказує на клас форми, який буде використовуватись для оновлення компанії. Цей клас має перевизначений метод `get_queryset()` для отримання запиту до БД, який повертає об'єкт компанії, який потрібно оновити.

3.4 Висновки за розділом 3

У цьому проєкті використовується Django ORM – інструмент фреймворку Django, який дозволяє взаємодіяти з БД, використовуючи високорівневі методи Python. Це значно зменшило кількість коду, необхідного для створення таблиць та роботи з ними БД. Також було представлено схему алгоритму функціонування програми та докладно описано реалізовані класи.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма, яку розроблено в роботі, призначена для пошуку та публікації вакансій для працевлаштування. Дане ПЗ широкого спектра можливостей, а саме: пошуку вакансій за ключовими словами, авторизації через соціальні мережі, надання можливості тестування як попереднього етапу відбору кандидата на роботу, фільтрації та сортування вакансій і заявок на роботу за різними параметрами [21]-[22].

4.2 Умови виконання програми

Для нормального функціонування програми, повинні виконуватися наступні умови до апаратної частини сервера:

- процесор Intel® Xeon® X3440 і вище;
- оперативна пам'ять 16 Гб і вище;
- постійна пам'ять 1 Тб і вище.

4.3 Експлуатація програми

Для початку роботи з вебдодатком для пошуку та публікації вакансій для працевлаштування, потрібно звернутися за URL-адресою, після чого відкриється головна сторінка (рис. 4.1), яка визначає роботу користувача без авторизації. Вебсторінка дозволяє переглянути загальну інформацію про повний перелік вакансій, відсортувати їх за певної категорії, використати інструмент для пошуку вакансій за різними критеріями, ввівши в текстове поле потрібний запит, і натиснути кнопку «Пошук».

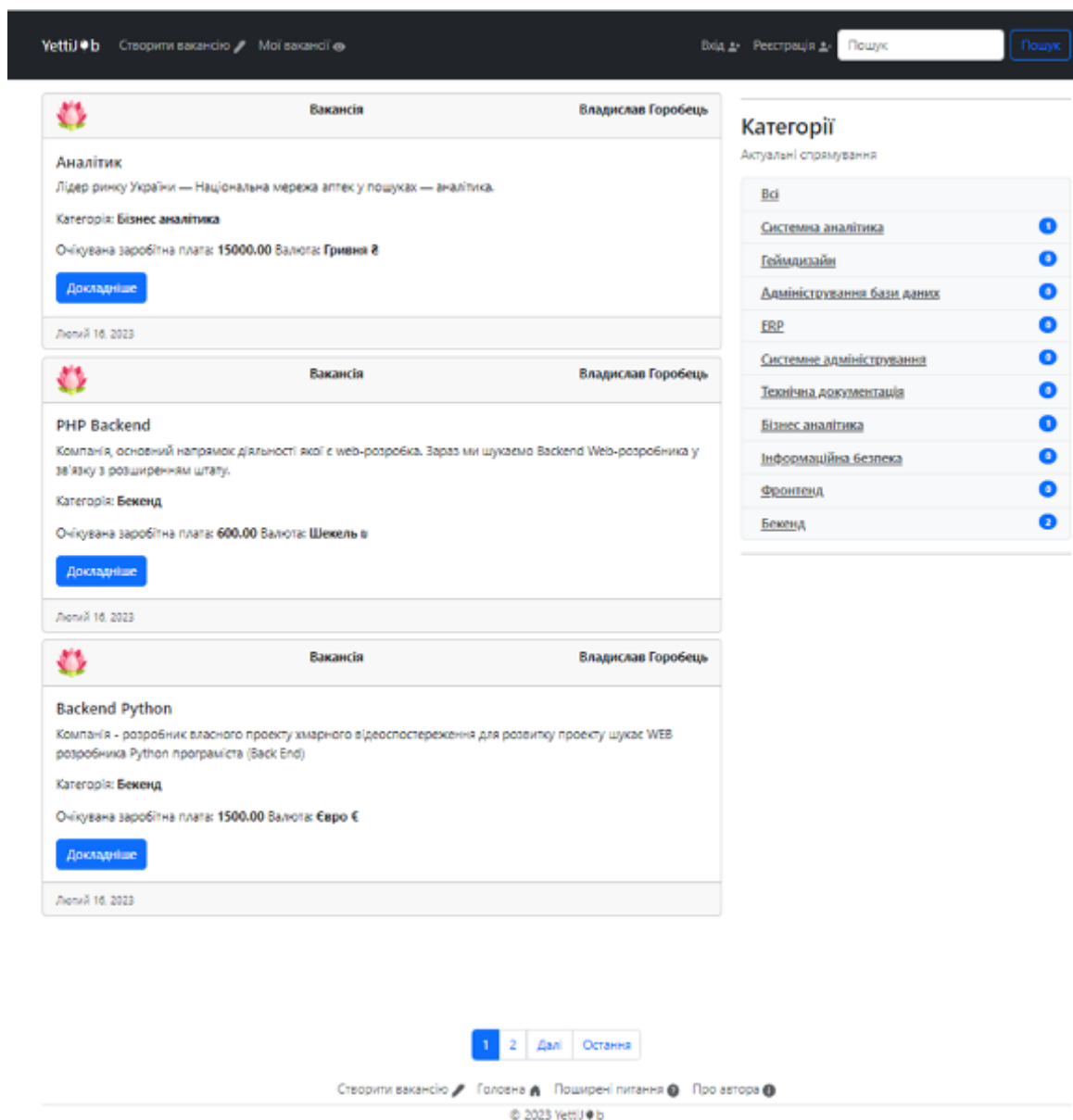


Рисунок 4.1 – Головна сторінка вебдодатку для неавторизованого користувача

Якщо потрібно перейти до обраної сторінки конкретної вакансії (рис. 4.2), треба натиснути на кнопку «Докладніше»: там можна знайти додаткову інформацію, таку як опис посади, вимоги до кандидатів та контактні дані роботодавця.

Yettū • b Створити резюме / Моє резюме

Дзень 16, 2021

Аналітик

Повне ім'я: **Владислав Горобець**

Візитівка компанії:



Категорія: **Бізнес аналітика**

Тип: **Повна зайнятість**

Email адреса: vladoshorobets@gmail.com

Очікувана заробітна плата: **15000.00** Валюта: **Гривня ₴**

Додаткова інформація:

Обов'язки 📌:

- розрахунок виконання умов по маркетинговим контрактам на щомісячній / щоквартальній основі;
- звірка розрахунків відділу з розрахунками маркетингових організацій, коригування розбіжностей у розрахунках;
- аналіз нових комерційних пропозицій;
- аналітична робота з великими обсягами інформації;
- аналіз, звітність з маркетингових проектів;
- прогноз виконання затверджених маркетингових пропозицій на щомісячній, щоквартальній основі;
- аналіз асортименту, складу маркетингових груп.

Тестування:

Назва Тесту: Інформаційні технології
Опис: Для перевірки знань на тему інформаційні технології
Кількість питань: 5
Тривалість: 30 секунд
Почати

Технології

Нам потрібно

- Java
- HTML
- Python

Компанія



**National University
"Zaporizhzhia
Polytechnic"**

Факультет комп'ютерних наук і технологій

Рисунок 4.2 – Перегляд вакансії

Щоб ознайомитись зі сторінками «Про автора» (рис. 4.3) та «Поширені питання» (рис. 4.4), треба натиснути на однойменні кнопки, що допоможе дізнатись більше інформації про автора даного вебзастосунку та зрозуміти основні функції та можливості вебдодатку.

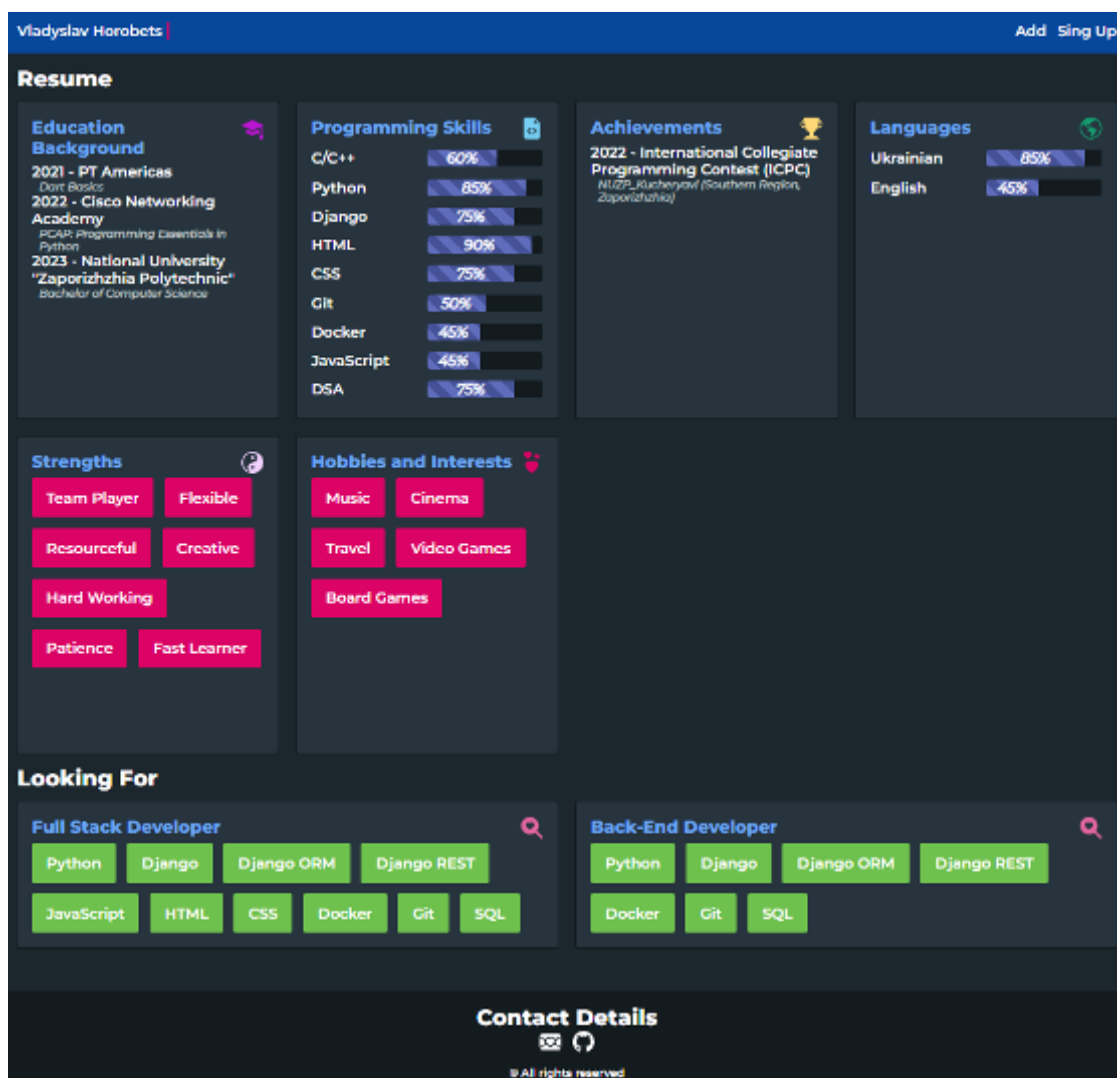


Рисунок 4.3 – Сторінка «Про автора»

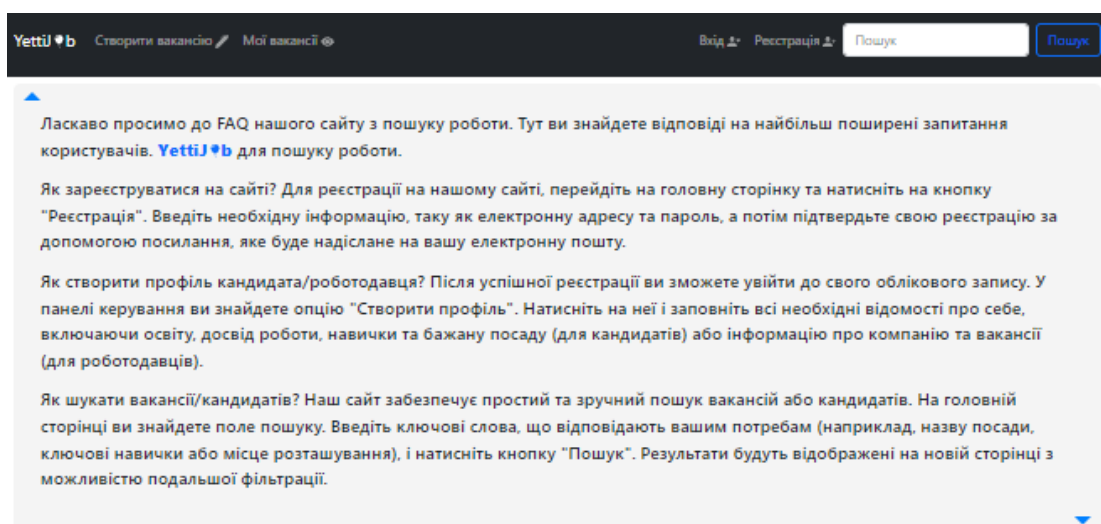


Рисунок 4.4 – Сторінка «Поширені питання»

Для отримання доступу до функцій, які вимагають авторизації, користувач може скористатися кнопками «Вхід» або «Реєстрація».

Як тільки користувач після реєстрації / авторизації опиняється в особистому кабінеті (рис.4.5), він може оновити дані свого профілю, створити / редагувати / видалити додаткову інформацію, видалити свій обліковий запис, натиснувши на кнопку «Видалити».

Yettij • b Створити резюме / Мої резюме

Обліковий запис Пошук

vladosik

Роль: Претендент

Обліковий запис

Ім'я користувача*

vladosik

Необхідно: 150 або менше символів: тільки букви, цифри та знаки @/./-/_

Ім'я

Прізвище

Вік

Email адреса

vladoshorobets3@gmail.com

Аватар*

Наразі users/default.png

Виберіть файл Файл не вибран

Оновити

Додаткова інформація

Освіта:

Дата закінчення: Червень 01, 2023

Навчальний заклад: НТУУ "КПІ", Україна
Факультет/Спеціальність: Радіотехнічний факультет
- Кафедра радіотехнічних систем

Редагувати Видалити

Додати освіту

Сертифікати:

Дата закінчення: Квітень 04, 2022

Назва курсу: Сертифікат основи керування ІТ продуктами

Редагувати Видалити

Додати сертифікат

Мови:

Інформація відсутня

Додати мову

Досвід роботи:

Інформація відсутня

Додати досвід роботи

Видалити

Рисунок 4.5 – Профіль користувача

Створити власне резюме користувач може за допомогою вбудованого до вебзастосунку текстового редактора, натиснувши на кнопку «Створити резюме» (рис. 4.6).

Yettü.b Створити резюме / Мій резюме Обліковий запис Пошук

Резюме

Посада*

Опис*

Зарплата*

Валута* Шекель ₪

Категорія*

Технології*

- Python
- Go
- Rust
- Dart

Тип зайнятості* Повна зайнятість

Додаткова інформація

Опубліковано

Створити

Рисунок 4.6 – Сторінка створення резюме

На сторінці певної вакансії користувач може додати оголошення до свого «Списку бажань» або пройти первинне тестування на обрану посаду (рис. 4.7) для подальшої співбесіди з роботодавцем.

Yetti! b
Створити резюме
Мої резюме
Обліковий запис
Пошук
Пошук

Тест "Інформаційні технології"
від National University "Zaporizhzhia Polytechnic"

Залишилося секунд: 28

Скільки революцій було у розвитку інформаційних технологій?

☐ 2
☐ 3
☐ 4
☐ 5

Що розуміється під даними про об'єкти, події та процеси?

☐ вміст баз знань
☐ необроблені повідомлення, що відображають окремі факти, процеси, події
☐ попередньо оброблена інформація
☐ повідомлення, що знаходяться у сховищах даних

Який процес схильний до зараження комп'ютерними вірусами?

☐ роботи з файлами
☐ форматування дискети
☐ вимкнення комп'ютера
☐ друку на принтері

Що потрібно для перевірки вірусу жорсткого диска?

☐ захищену програму
☐ завантажувальну програму
☐ файл з антивірусною програмою
☐ дискету з антивірусною програмою, захищену від запису

Як може з'явитися вірус на комп'ютері?

☐ переміщення з гнучкого диска
☐ при вирішенні математичного завдання
☐ при підключенні до комп'ютера модему
☐ мимоволі

Підтвердити
Головна

Рисунок 4.7 – Проходження первинного тестування

Як тільки роботодавець після реєстрації / авторизації опиняється в особистому кабінеті (рис. 4.8), він може оновити дані свого профілю, створити / редагувати / видалити інформацію про свою компанію, видалити свій обліковий запис, натиснувши на кнопку «Видалити».

The screenshot shows the user profile of Vladislav Horobets on the YettiJob platform. The header includes the YettiJob logo, navigation links for creating and viewing vacancies, and a search bar. The profile section on the left displays the user's role as a Recruiter, their account name, and various personal details in a form: username (Vlad), email (vladoshorobets@gmail.com), first name (Владислав), last name (Горобець), age (21), and avatar (users/2023/04/04/lotus.png). A right sidebar titled 'Додаткова інформація' (Additional Information) shows test results and a company profile for National University 'Zaporizhzhia Polytechnic'.

YettiJob Створити вакансію / Мої вакансії

Обліковий запис Пошук

Владислав Горобець

Роль: Рекрутер

Обліковий запис

Ім'я користувача*

Vlad

Необхідно: 150 або менше символів, тільки букви, цифри та знаки @/./+/-/_

Ім'я

Владислав

Прізвище

Горобець

Вік

21

Email адреса

vladoshorobets@gmail.com

Аватар*

Наразі: users/2023/04/04/lotus.png

Виберіть файл Файл не вибран

Оновити Видалити

Додаткова інформація

Тестування:

Мої тести

Додати тест

Додати питання

Результати тестування претендентів

Компанія:

National University "Zaporizhzhia Polytechnic"

Факультет комп'ютерних наук і технологій

Редагувати Видалити

Рисунок 4.8 – Профіль роботодавця

Роботодавець може створити вакансію, натиснувши на кнопку «Створити вакансію», також створити для вакансії первинне тестування (рис. 4.9).

The screenshot shows the 'Тест' (Test) creation page. It contains a form with the following fields: 'Назва*' (Name) with the value 'PHP для Junior', 'Опис*' (Description), 'Оголошення*' (Announcement) with a dropdown menu showing 'PHP Backend', 'Кількість питань*' (Number of questions) with the value '120', and 'Тривалість*' (Duration) with the value '4560'. Below the form, there is a note 'тривалість тесту в секундах' (test duration in seconds) and a green 'Додати' (Add) button.

YettiJob Створити вакансію / Мої вакансії

Обліковий запис Пошук

Тест

Назва*

PHP для Junior

Опис*

Оголошення*

PHP Backend

Кількість питань*

120

Тривалість*

4560

тривалість тесту в секундах

Додати

Рисунок 4.9 – Сторінка створення тесту

Роботодавець може додати до тесту запитання (рис. 4.10).

YettiU b Створити вакансію Мої вакансії Обліковий запис Пошук

Вміст*

1 + 2 = ?

Тест*

Інформаційні технології

Додати

Питання	Тест	Видалити
Скільки революцій було у розвитку інформаційних технологій?	Інформаційні технології	Видалити
Що розуміється під даними про об'єкти, події та процеси?	Інформаційні технології	Видалити
Який процес схильний до зараження комп'ютерними вірусами?	Інформаційні технології	Видалити
Що потрібно для перевірки вірусу жорсткого диска?	Інформаційні технології	Видалити
Як може з'явитися вірус на комп'ютері?	Інформаційні технології	Видалити

Рисунок 4.10 – Сторінка додавання питань до тесту

Роботодавець може переглянути результати тестування користувачів (рис. 4.11).

Користувач	Тест	Вакансія	Оцінка	Додати до чату	Видалити
vladyslavhorobets	Інформаційні технології	Аналітик	5.0	Додати	Видалити
admin	Інформаційні технології	Аналітик	0.0	Додати	Видалити
admin	Інформаційні технології	Аналітик	0.0	Додати	Видалити
admin	Інформаційні технології	Аналітик	0.0	Додати	Видалити
vladosik	Інформаційні технології	Аналітик	1.0	Додати	Видалити
vladosik	Інформаційні технології	Аналітик	0.0	Додати	Видалити
vladosik	Інформаційні технології	Аналітик	1.0	Додати	Видалити

Рисунок 4.11 – Сторінка перегляду результатів тестування

4.4 Тестування програми

Відповідно до технічного завдання було створено набір функцій, кожна з яких була ретельно протестована шляхом виклику та порівняння очікуваного результату з отриманим. Для кожної функції ці дії було виконано не менше одного разу. Всі результати тестування відповідали очікуваним, що свідчить про успішність реалізації програми. Тестування було завершено успішно, що в свою чергу підтверджує успішність реалізації програми.

Чеклист тестування програми представлений в табл. 4.1.

Таблиця 4.1 – Чеклист тестування програми

Перевірка	Результат
1	2
Реєстрація	+
Вибір ролі на сайті	+
Реєстрація за допомогою сторонніх вебдодатків	+
Авторизація	+
Редагування власного кабінету	+
Видалення акаунту	+
Додавання зображення аккаунта	+
Додавання освіти користувачем	+
Видалення освіти користувачем	+
Редагування освіти користувачем	+
Додавання мов користувачем	+
Редагування мов користувачем	+
Видалення мов користувачем	+
Додавання сертифікатів користувачем	+
Редагування сертифікатів користувачем	+

Продовження таблиці 4.1

1	2
Видалення сертифікатів користувачем	+
Додавання досвіду роботи користувачем	+
Редагування досвіду роботи користувачем	+
Видалення досвіду роботи користувачем	+
Додавання компанії роботодавцем	+
Редагування компанії роботодавцем	+
Видалення компанії роботодавцем	+
Додавання логотипу компанії роботодавцем	+
Додавання відеопрезентації компанії роботодавцем	+
Додавання вакансії роботодавцем	+
Редагування вакансії роботодавцем	+
Видалення вакансії роботодавцем	+
Перегляд вакансії користувачем	+
Додавання резюме користувачем	+
Редагування резюме користувачем	+
Видалення резюме користувачем	+
Перегляд резюме роботодавцем	+
Пошук вакансій користувачем	+
Пошук резюме роботодавцем	+
Перегляд відеопрезентації користувачем	+
Створення тесту для вакансії роботодавцем	+
Редагування тесту для вакансії роботодавцем	+
Видалення тесту для вакансії роботодавцем	+
Створення питань для тесту роботодавцем	+
Редагування питань для тесту роботодавцем	+
Видалення питань для тесту роботодавцем	+

Кінець таблиці 4.1

1	2
Перегляд результатів тестування користувачів роботодавцем	+
Пройдення користувачем тестування	+
Додавання резюме в «Список бажаного» роботодавцем	+
Перегляд резюме у «Списку бажаного» роботодавцем	+
Видалення резюме зі «Списку бажаного» роботодавцем	+
Додавання вакансії в «Список бажаного» користувачем	+
Перегляд вакансії у «Списку бажаного» користувачем	+
Видалення вакансії зі «Списку бажаного» користувачем	+
Створення кімнат роботодавцем	+
Видалення кімнат роботодавцем	+
Редагування кімнат роботодавцем	+
Занесення користувачів до кімнат роботодавцем	+
Перегляд кімнат користувачем	+

4.5 Висновки за розділом 4

Програма має розроблені окремі інтерфейси для користувачів та роботодавців. Кожен з них має свої функції та можливості, щоб забезпечити ефективну та зручну роботу з програмою [23].

Користувачі та роботодавці можуть використовувати програму для пошуку вакансій та кандидатів на роботу відповідно. Вони можуть переглядати та редагувати свої профілі, стежити за оновленнями, взаємодіяти з іншими користувачами через різні зв'язки, що забезпечують зручну навігацію та швидкий доступ до потрібної інформації.

Для забезпечення максимальної зручності та ефективності програма має велику кількість зв'язків між сторінками, які дозволяють швидко переміщатися

між різними секціями та функціями. Користувачі можуть легко знайти та переглянути потрібну інформацію, не витрачаючи забагато часу на пошук або навігацію по програмі.

Загалом програма надає зручні та ефективні інструменти для роботи з різними типами користувачів, що дозволяє забезпечити максимально комфортні умови для виконання їх завдань та досягнення поставлених цілей.

ВИСНОВКИ

В ході виконаної роботи було проведено аналіз предметної області, в межах якого було виділено основні поняття та методології щодо розробки вебзастосунку пошуку та публікації вакансій для працевлаштування, проведено огляд програмних аналогів, включаючи вебсайти Robota.ua, Jobs.ua, Work.ua, Trud.ua. Проаналізовано плюси та мінуси їх взаємодії з користувачами. Враховуючи в них відсутність первинного тестування кандидатів на вакансію, реєстрації / авторизації через сервіси GitHub, GitLab, багато платних функцій, саме це було визначено в якості особливості розробленого вебзастосунку.

Розробку вебзастосунку для пошуку та публікації вакансій для працевлаштування з можливістю первинного тестування користувачів було виконано за допомогою вебфреймворку Django з визначенням структури даних на основі використання БД, вимог до програми, розробленням структури програми і алгоритму роботи на основі цих вимог, реалізацією відповідних одиниць вебзастосунку. Для розробки використано мову програмування Python, мову розмітки HTML, мову стилю сторінок CSS, систему керування базами даних PostgreSQL. Для покращення структури проєкту використовувався шаблон проєктування MVT, що забезпечує чітке розділення функціональних компонентів системи.

Створена вебсистема дозволяє реалізувати можливості перегляду вакансій, резюме, перегляд профілів користувачів, створення оголошень з використанням текстового редактора, авторизація, реєстрація за допомогою сторонніх сервісів, перегляд відеопрезентацій, створення та перегляд кімнат з інформацією про компанію, а також проходження тестів, автоматичний підрахунок балів, перегляд оголошень за категоріями, додавання до профілю додаткової інформації [24].

Розроблене ПЗ сприяє тому, що роботодавці знайшли найкращих фахівців, відсіюючи необігрувані заявки, а кандидати на працевлаштування – найкращі робочі місця для їхнього професійного розвитку. Результати цієї роботи можуть бути взаємно корисним.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. 7 ключових етапів процесу підбору персоналу – Hurma [Електрон. ресурс]. – Режим доступу : <https://hurma.work/blog/7-klyuchovyh-etapiv-proczesu-pidboru-personalu/>.
2. Сайти пошуку роботи в Україні – Marketer [Електрон. ресурс]. – Режим доступу : <https://marketer.ua/ua/job-search-sites-in-ukraine/>.
3. Robota.ua [Електрон. ресурс]. – Режим доступу : <https://rabota.ua/ua>.
4. Work.ua [Електрон. ресурс]. – Режим доступу : <https://www.work.ua/>.
5. Jobs.ua [Електрон. ресурс]. – Режим доступу : <https://jobs.ua/>.
6. Trud.ua [Електрон. ресурс]. – Режим доступу : <https://trud.ua/>.
7. Guthals, S. GitHub For Dummies Example [Text] / Sarah Guthals. – 2nd Edition. – Hoboken : John Wiley & Sons, Inc., 2023. – 307 p.
8. Campbell, S. Python Vs PHP [Electronic resource] : What's the Difference Between Python and PHP? / S. Campbell. – Access mode : <https://www.guru99.com/python-vs-php.html>.
9. What is Python used for? 10 practical Python uses [Electronic resource]. – Access mode : <https://www.futurelearn.com/info/blog/what-is-pythonused-for>.
10. Django Tutorials – Real Python [Electronic resource]. – Access mode : <https://realpython.com/tutorials/django/>.
11. William, S. Vincent Django for Beginners [Text] : Build websites with Python and Django / William S. Vincent. – WelcomeToCode, 2021. – 296 p.
12. Що таке функціональні вимоги: приклади, визначення, повний посібник – visuresolutions [Електрон. ресурс]. – Режим доступу : <https://visuresolutions.com/uk/блог/функціональні-вимоги/>.
13. Matthew, N. Beginning Databases with PostgreSQL From Novice to Professional [Text] / Neil Matthew, Richard Stones. – Second Edition. – Apress, 2005. – 589 p.
14. Beginner's Guide to PostgreSQL – datacamp [Electronic resource]. – Access mode : <https://www.datacamp.com/tutorial/beginners-introduction-postgresql>.

15. The Model View Controller Pattern – MVC Architecture and Frameworks Explained. Freecodecamp [Electronic resource]. – Access mode : <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained>.
16. Django MVT [Electronic resource]. – Access mode : <https://www.javatpoint.com/django-mvt>.
17. Пол, Беррі. Вивчаємо програмування на Python [Текст] / Пол Беррі. – Харків: Фабула, 2021. – 624 с. : іл.
18. Марк, Лутц. Вивчаємо Python [Текст] / Марк Лутц. – 5-е видання. – Київ : Діалектика, 2020. – 832 с.
19. Matthes, E. Python Crash Course [Text] : A Hands-On, Project-Based Introduction / Eric Matthes. – 2nd Edition. – No Starch Press, 2019. – 544 p.
20. Django first steps for total beginners [Electronic resource] : a quick tutorial – Medium. – Access mode : <https://towardsdatascience.com/django-first-steps-for-the-total-beginners-a-quick-tutorial-5f1e5e7e9a8c>.
21. Sufyan bin Uzayr. Frontend Development The Ultimate Guide [Text] / Sufyan bin Uzayr. – First edition. – Boca Raton : CRC Press, 2023. – 178 p.
22. Matuszek, D. Quick JavaScript [Text] / David Matuszek. – First edition. – CRC Press, 2023. – 167 p.
23. Travis, D. Think Like a UX Researcher. How to Observe Users, Influence. Design, and Shape Business Strateg [Text] / David Travis, Philip Hodgso. – Boca Raton : CRC Press, 2019. – 289 p.
24. Горобець, В.І. Програмна система пошуку та публікації вакансій для працевлаштування [Текст] / В.І. Горобець, В. М. Льовкін // 27-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у ХХІ столітті». Зб. матеріалів форуму. Т. 6, Ч. І. – Харків : ХНУРЕ, 2023. – С. 407-408.

ДОДАТОК А
Технічне завдання

Вступ

Сучасні обставини примушують людей постійно шукати нові робочі місця, і Інтернет допомагає зробити цей процес швидшим і більш зручним. Завдяки Інтернету, люди можуть шукати вакансії в онлайн-середовищі, що зменшує втрату часу на огляд оголошень. Але серед великої кількості сайтів з працевлаштуванням, не так легко знайти, який забезпечує якісні та професійні послуги для пошуку роботи. З іншого боку, роботодавці повинні мати можливість швидко та ефективно знайти найкращих кандидатів, але процес відбору може бути складним. Щоб зменшити це ускладнення, можна використовувати первинне тестування, яке допоможе ефективно відсіяти кандидатів, які не відповідають вимогам вакансії, та зосередитися на кандидатах, які вже підійшли за всіма вимогами, що зменшить загальний час, потрібний на процес найму.

А.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Програмне забезпечення пошуку та публікації вакансій для працевлаштування» у відповідності з наказом № 87 від 04 квітня 2023 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма має використовуватися для отримання переліку вакансій чи резюме, пошуку потрібного оголошення, первинного тестування кандидатів на певну вакансію, створення тестів, додавання питань до тестів та перегляду результатів тестування роботодавцем.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Набір функціональних вимог, які має забезпечувати програма, включає:

- реєстрація;
- вибір ролі на сайті;
- реєстрація за допомогою сторонніх вебдодатків;
- авторизація;
- редагування власного кабінету;
- видалення аккаунту;
- додавання зображення аккаунта;
- додавання освіти користувачем;
- видалення освіти користувачем;
- редагування освіти користувачем;
- додавання мов користувачем;
- редагування мов користувачем;
- видалення мов користувачем;
- додавання сертифікатів користувачем;
- редагування сертифікатів користувачем;
- видалення сертифікатів користувачем;
- додавання досвіду роботи користувачем;
- редагування досвіду роботи користувачем;
- видалення досвіду роботи користувачем;
- додавання компанії роботодавцем;
- редагування компанії роботодавцем;
- видалення компанії роботодавцем;
- додавання логотипу компанії роботодавцем;
- додавання відеопрезентації компанії роботодавцем;
- додавання вакансії роботодавцем;
- редагування вакансії роботодавцем;

- видалення вакансії роботодавцем;
- перегляд вакансії користувачем;
- додавання резюме користувачем;
- редагування резюме користувачем;
- видалення резюме користувачем;
- перегляд резюме роботодавцем;
- пошук вакансій користувачем;
- пошук резюме роботодавцем;
- перегляд відеопрезентації користувачем;
- створення тесту для вакансії роботодавцем;
- редагування тесту для вакансії роботодавцем;
- видалення тесту для вакансії роботодавцем;
- створення питань для тесту роботодавцем;
- редагування питань для тесту роботодавцем;
- видалення питань для тесту роботодавцем;
- перегляд результатів тестування користувачів роботодавцем;
- проходження користувачем тестування;
- додавання резюме в «Список бажаного» роботодавцем;
- перегляд резюме у «Списку бажаного» роботодавцем;
- видалення резюме зі «Списку бажаного» роботодавцем;
- додавання вакансії в «Список бажаного» користувачем;
- перегляд вакансії у «Списку бажаного» користувачем;
- видалення вакансії зі «Списку бажаного» користувачем;
- створення кімнат роботодавцем;
- видалення кімнат роботодавцем;
- редагування кімнат роботодавцем;
- занесення користувачів до кімнат роботодавцем;
- перегляд кімнат користувачем.

А.3.2 Умови експлуатації

Головною умовою експлуатації проекту є розділення ролей між роботодавцем та користувачем. Роботодавець може переглядати резюме, створювати вакансії і тестування до них, додавати користувачів до них. Користувач може переглядати вакансії та кімнати з роботодавцями та проходити тестування.

А.3.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програми, повинні виконуватися наступні умови до апаратної частини сервера:

- процесор Intel® Xeon® X3440 і вище;
- оперативна пам'ять 16 Гб і вище;
- постійна пам'ять 1 Тб і вище.

А.4 Порядок контролю та приймання

Для виконання дипломної кваліфікаційної роботи критичним є виконання визначеного завдання, що включає безпосередньо завдання на роботу і технічне завдання, за темою, яка зафіксована наказом, у строки, які визначені завданням на роботу.

У підсумку виконується захист роботи за оприлюдненим наперед графіком перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла `urls.py` пакета `core`

```
from django.contrib import admin
from django.urls import path, include
from django.conf.urls.static import static
from django.conf import settings

urlpatterns = [path('admin/', admin.site.urls),
               path("", include('workspace.urls')), path("", include('webchat.urls')),
               path("", include('author.urls')), path('quiz/', include('quiz.urls')),]
if settings.DEBUG: urlpatterns += static(settings.MEDIA_URL,
                                         document_root=settings.MEDIA_ROOT)
```

Б.2 Текст файла `.env`

```
SECRET_KEY=django-insecure-+o5*i^j##@vf0$fag7vkg5#w#1d-
_$p0ivktdo#dgp#*)^3aky
DEBUG=True
```

Б.3 Текст файла `admin.py` пакета `workspace`

```
from django.contrib import admin
from .mixins import (AdvertMixin, ExtendedAdvertMixin, )
from .models import (Category, Advert, Technology,)
@admin.register(Category)
class CategoryAdmin(ExtendedAdvertMixin, admin.ModelAdmin):pass
@admin.register(Technology)
class TechnologyAdmin(ExtendedAdvertMixin, admin.ModelAdmin):pass
@admin.register(Advert)
class AdvertAdmin(ExtendedAdvertMixin, admin.ModelAdmin):
    list_display = ('pk', 'title', 'category', 'published', 'created')
```

```

readonly_fields = ('user', 'advert_type', 'slug', 'created', 'updated')
filter_horizontal = ('technologies',)
list_filter = ('category', 'type', 'published', 'currency',)
save_on_top = True
fieldsets = ((None, {'classes': ('wide',),
                    'fields': ('title', 'description', 'type', 'category',
                               'technologies', ('salary', 'currency'), 'about', 'published',),}),
              ('Додаткова інформація', {'classes': ('collapse',),
                    'fields': ('user', 'advert_type', 'slug', 'created', 'updated',),})),)
def get_fieldsets(self, request, obj):
    if not obj: fieldsets = ((None, {'classes': ('wide',),
                    'fields': ('title', 'description', 'type', 'category', 'technologies',
                               ('salary', 'currency'), 'about', 'published',),})),)
    return fieldsets
    return super().get_fieldsets(request, obj)

```

Б.4 Текст файлу apps.py пакета workspace

```

from django.apps import AppConfig,
class WorkspaceConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'workspace'

```

Б.5 Текст файлу forms.py пакета workspace

```

from django.forms import ModelForm
from .models import (ResumeAdvert, VacancyAdvert,)
class ResumeForm(ModelForm):
    class Meta:
        model = ResumeAdvert

```

```

fields = ['title','description',
          'salary', 'currency', 'category','technologies','type', 'about', 'published',]
class VacancyForm(ModelForm):
    class Meta:
        model = VacancyAdvert
        fields = ['title','description','salary',
                  'currency', 'category','technologies','type', 'about', 'published',]

```

Б.6 Текст файлу mixins.py пакета workspace

```

class AdvertMixin:
    list_editable = ('published',)
    readonly_fields = ('user', 'created', 'updated')
    save_as = True
    list_per_page = 10
    def save_model(self, request, obj, form, change):
        if not hasattr(obj, 'user'): obj.user = request.user
        return super().save_model(request, obj, form, change)
class ExtendedAdvertMixin(AdvertMixin):
    list_display = ('pk', 'title', 'published', 'created')
    list_display_links = ('pk', 'title')
    search_fields = ('title', 'about')
    fieldsets = ((None, {'classes': ('wide'),'fields': ('title','about','published',),}),
                 ('Додаткова інформація', {'classes': ('collapse',),
                  'fields': ('user','created','updated',),}),)
    def get_fieldsets(self, request, obj):
        if not obj: fieldsets = ((None, {'classes': ('wide',),'fields':
('title','about','published',),}),)
        return fieldsets
    return super().get_fieldsets(request, obj)

```

Б.7 Текст файлу models.py пакета workspace

```

from django.db import models
from django.urls import reverse
from django.core.validators import MinValueValidator
from django.conf import settings
from ckeditor.fields import RichTextField
from .utils import unicode_unique_slug
class AbstractAdvert(models.Model):
    title = models.CharField(verbose_name='назва', max_length=128)
    slug = models.SlugField(verbose_name='url', unique=True, editable=False)
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE, verbose_name='користувач',)
    published = models.BooleanField(verbose_name='опубліковано',
        default=True,)
    created = models.DateTimeField(verbose_name='дата створення',
        auto_now_add=True,)
    updated = models.DateTimeField(verbose_name='дата оновлення',
        auto_now=True,)
    about = RichTextField(verbose_name='додаткова інформація', blank=True)
    class Meta: abstract = True
    def __str__(self): return self.title
    def save(self, *args, **kwargs):
        if not self.pk: self.slug = unicode_unique_slug(self.title)
        super().save(*args, **kwargs)
class Category(AbstractAdvert):
    class Meta:
        verbose_name = 'категорія'
        verbose_name_plural = 'категорії'
class Technology(AbstractAdvert):

```

```

class Meta:
    verbose_name = 'технологія'
    verbose_name_plural = 'технології'
class BaseAdvert(AbstractAdvert):
    class Type(models.TextChoices):
        PARTTIME = 'PT', 'Неповна зайнятість'
        FULLTIME = 'FT', 'Повна зайнятість'
    class Currency(models.TextChoices):
        DOLLAR = 'USD', 'Долар $'
        EURO = 'EUR', 'Євро €'
        HRYVNIA = 'UAH', 'Гривня ₴'
        SHEKEL = 'ILS', 'Шекель ₪'
    salary = models.DecimalField(
        verbose_name='зарплата',max_digits=10,decimal_places=2,
        validators=[MinValueValidator(0.00),])
    currency = models.CharField(verbose_name='валюта',
        max_length=3,choices=Currency.choices,default=Currency.SHEKEL,)
    category = models.ForeignKey(Category,
        on_delete=models.SET_NULL,null=True,
        limit_choices_to={'published': True},verbose_name='категорія',)
    technologies = models.ManyToManyField(Technology,
        verbose_name='технології',)
    type = models.CharField(verbose_name='тип зайнятості',
        max_length=2,choices=Type.choices,default=Type.FULLTIME,)
    class Meta:
        abstract = True
class Advert(BaseAdvert):
    class AdvertTypeChoices(models.TextChoices):
        RESUME = 'RM', 'Резюме'
        VACANCY = 'VY', 'Вакансія'

```

```

title = models.CharField(verbose_name='посада', max_length=128)
description = models.CharField(verbose_name='опис', max_length=256)
advert_type = models.CharField(verbose_name='тип оголошення',
    max_length=2, choices=AdvertTypeChoices.choices,)
class Meta:
    verbose_name = 'оголошення'
    verbose_name_plural = 'оголошення'
    def get_absolute_url(self): return reverse("workspace:advert_detail",
kwargs={'pk': self.pk, 'slug': self.slug})
    def is_resume(self): return True if self.advert_type==
self.AdvertTypeChoices.RESUME else False
    def is_vacancy(self): return True if
self.advert_type==self.AdvertTypeChoices.VACANCY else False
class ResumeAdvert(Advert):
    class Meta:proxy = True
    def save(self, *args, **kwargs):
        if not self.pk: self.advert_type = self.AdvertTypeChoices.RESUME
        super().save(*args, **kwargs)
class VacancyAdvert(Advert):
    class Meta:proxy = True
    def save(self, *args, **kwargs):
        if not self.pk: self.advert_type = self.AdvertTypeChoices.VACANCY
        super().save(*args, **kwargs)
class Wishlist(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,verbose_name='користувач',)
    adverts = models.ManyToManyField(Advert,verbose_name='оголошення',)
    class Meta:
        verbose_name = 'список бажань'
        verbose_name_plural = 'список бажань'

```

Б.8 Текст файла `urls.py` пакета `workspace`

```

from django.urls import path
from .views import (About, HomeView, AdvertDetailView, MyAdvertView,
    ResumeCreateView, ResumeUpdateView, ResumeDeleteView,
    VacancyCreateView, VacancyUpdateView, VacancyDeleteView)
app_name = 'workspace'
urlpatterns = [path("", HomeView.as_view(), name='home'),
    path('about/', About.as_view(), name='about'),
    path('advert/', MyAdvertView.as_view(), name='my_advert'),
    path('<int:category>/', HomeView.as_view(), name='home'),
    path('advert/<int:pk>/<slug:slug>/', AdvertDetailView.as_view(),
        name='advert_detail'),
    path('resume/create', ResumeCreateView.as_view(), name='resume_create'),
    path('resume/update/<int:pk>/<slug:slug>',
        ResumeUpdateView.as_view(), name='resume_update'),
    path('resume/delete/<int:pk>/<slug:slug>',
        ResumeDeleteView.as_view(), name='resume_delete'),
    path('vacancy/create', VacancyCreateView.as_view(), name='vacancy_create'),
    path('vacancy/update/<int:pk>/<slug:slug>',
        VacancyUpdateView.as_view(), name='vacancy_update'),
    path('vacancy/delete/<int:pk>/<slug:slug>',
        VacancyDeleteView.as_view(), name='vacancy_delete'),]

```

Б.9 Текст файла `utils.py` пакета `workspace`

```

import datetime
from django.utils.text import slugify
from unicode import unicode
def unicode_unique_slug(text):

```

```

now = datetime.datetime.today()
unicode = unicode(f'{text}{now.timestamp()}')
return slugify(unicode)

```

Б.10 Текст файла views.py пакета workspace

```

from django.db import models
from django.views.generic import (TemplateView, ListView, DetailView,
    CreateView, UpdateView, DeleteView)
from     django.contrib.auth.mixins     import     PermissionRequiredMixin,
LoginRequiredMixin
from django.urls import reverse_lazy
from .models import (
    Advert, Category, ResumeAdvert, VacancyAdvert, Technology,)
from .forms import (ResumeForm, VacancyForm,)
class About(TemplateView):
    template_name = 'workspace/about.html'
class HomeView(ListView):
    model = Advert
    paginate_by = 3
    template_name = 'workspace/advert_list.html'
    paginate_orphans = 2
    def get_queryset(self):
        user = self.request.user
        queryset = super().get_queryset()
        if     user.is_anonymous     or     user.is_applicant:     queryset     =
queryset.filter(advert_type=self.model.AdvertTypeChoices.VACANCY)
        else:
            queryset
            =
queryset.filter(advert_type=self.model.AdvertTypeChoices.RESUME)

```

```

        queryset = (queryset.select_related('category', 'user')
            .prefetch_related(models.Prefetch('technologies',
                queryset=Technology.objects.filter(published=True).only('title'))))
        category = self.kwargs.get('category', None)
        if category:
            return queryset.filter(models.Q(published=True) &
models.Q(category=category))
        return queryset.filter(published=True)
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        if self.request.user.is_anonymous or self.request.user.is_applicant:
            context['categories'] = (Category.objects.filter(published=True)
                .annotate(advert_count=models.Count('advert',
filter=models.Q(advert__advert_type=Advert.AdvertTypeChoices.VACANCY))))
        else:
            context['categories'] = (Category.objects.filter(published=True)
                .annotate(advert_count=models.Count('advert',
filter=models.Q(advert__advert_type=Advert.AdvertTypeChoices.RESUME))))
        return context
    class MyAdvertView(LoginRequiredMixin, ListView):
        model = Advert
        paginate_by = 4
        template_name = 'workspace/advert_list.html'
        paginate_orphans = 2
        def get_queryset(self):
            user = self.request.user
            queryset = super().get_queryset()
            if user.is_recruiter:
                queryset = queryset.filter(
                    models.Q(advert_type=self.model.AdvertTypeChoices.VACANCY) &
models.Q(user=user.pk),)

```

```

else: queryset = queryset.filter(
    models.Q(advert_type=self.model.AdvertTypeChoices.RESUME) &
models.Q(user=user.pk))
    queryset = (queryset.select_related('category',
'user').prefetch_related(models.Prefetch('technologies',queryset=Technology.objects.filt
er(published=True).only('title'))))
    category = self.kwargs.get('category', None)
    if category: return queryset.filter(models.Q(published=True) &
models.Q(category=category))
    return queryset.filter(published=True)
def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    if self.request.user.is_applicant:
        context['categories'] = (Category.objects.filter(published=True)
        .annotate(advert_count=models.Count('advert',
filter=models.Q(advert__advert_type=Advert.AdvertTypeChoices.VACANCY))))
    else:
        context['categories'] = (Category.objects.filter(published=True)
        .annotate(advert_count=models.Count('advert',
filter=models.Q(advert__advert_type=Advert.AdvertTypeChoices.RESUME))))
    return context
class AdvertDetailView(DetailView):
    model = ResumeAdvert
    query_pk_and_slug = True
    template_name = 'workspace/advert_detail.html'
    def get_queryset(self):
        queryset = super().get_queryset()
        return (queryset.select_related('category', 'user')
        .prefetch_related('technologies','user__quiz_set',
        'user__course_set', 'user__education_set', 'user__language_set',))

```

```

class ResumeCreateView(PermissionRequiredMixin, CreateView):
    model = ResumeAdvert
    template_name = 'workspace/resume_create_form.html'
    form_class = ResumeForm
    permission_required = 'workspace.add_resumeadvert'
    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)

class ResumeUpdateView(PermissionRequiredMixin, UpdateView):
    model = ResumeAdvert
    form_class = ResumeForm
    template_name = 'workspace/resume_update_form.html'
    query_pk_and_slug = True
    permission_required = 'workspace.change_resumeadvert'
    def get_queryset(self):
        user=self.request.user
        return super().get_queryset().filter(user=user)

class ResumeDeleteView(PermissionRequiredMixin, DeleteView):
    model = ResumeAdvert
    query_pk_and_slug = True
    success_url = reverse_lazy('workspace:home')
    template_name = 'workspace/resume_confirm_delete.html'
    permission_required = 'workspace.delete_resumeadvert'
    def get_queryset(self):
        user = self.request.user
        return super().get_queryset().filter(user=user)

class VacancyCreateView(PermissionRequiredMixin, CreateView):
    model = VacancyAdvert
    template_name = 'workspace/vacancy_create_form.html'
    form_class = VacancyForm

```

```

permission_required = 'workspace.add_vacancyadvert'
def form_valid(self, form): form.instance.user = self.request.user
    return super().form_valid(form)
class VacancyUpdateView(PermissionRequiredMixin, UpdateView):
    model = VacancyAdvert
    form_class = VacancyForm
    template_name = 'workspace/vacancy_update_form.html'
    query_pk_and_slug = True
    permission_required = 'workspace.change_vacancyadvert'
    def get_queryset(self):returnsuper().get_queryset().filter(self.request.user =user)
class VacancyDeleteView(PermissionRequiredMixin, DeleteView):
    model = VacancyAdvert
    query_pk_and_slug = True
    success_url = reverse_lazy('workspace:home')
    template_name = 'workspace/vacancy_confirm_delete.html'
    permission_required = 'workspace.delete_vacancyadvert'
    def get_queryset(self):returnsuper().get_queryset().filter(user=self.request.user)
class About(TemplateView):
    template_name = 'workspace/about.html'

```

Б.11 Текст файлу views.py пакета quiz

```

from django.shortcuts import render, redirect
from django.urls import reverse, reverse_lazy
from django.views.generic import (
ListView, DetailView, CreateView, UpdateView, DeleteView)
    from     django.contrib.auth.mixins     import     PermissionRequiredMixin,
LoginRequiredMixin
    from .models import (Quiz,Question, Answer, MarksOfUser,)
    from django.http import JsonResponse

```

```

from django.contrib.auth.decorators import (login_required,
permission_required)

from .forms import QuizForm, QuestionForm
from django.forms import inlineformset_factory

class QuizListView(LoginRequiredMixin, ListView):
    model = Quiz
    template_name = 'quiz/quiz_list.html'
    def get_queryset(self):
        user = self.request.user
        queryset = super().get_queryset()
        return queryset.filter(user=user)

class QuizCreateView(PermissionRequiredMixin, CreateView):
    model = Quiz
    template_name = 'quiz/add_quiz.html'
    success_url = reverse_lazy('quiz:add_question')
    form_class = QuizForm
    permission_required = 'quiz.add_quiz'
    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)

class QuizUpdateView(PermissionRequiredMixin, UpdateView):
    model = Quiz
    pk_url_kwarg = 'myid'
    template_name = 'quiz/edit_quiz.html'
    success_url = reverse_lazy('quiz:quiz_list')
    form_class = QuizForm
    permission_required = 'quiz.change_quiz'
    def get_queryset(self):
        user = self.request.user
        queryset = super().get_queryset()
        return queryset.filter(user=user)

```

```

class QuizDeleteView(PermissionRequiredMixin, DeleteView):
    model = Quiz
    pk_url_kwarg = 'myid'
    success_url = reverse_lazy('quiz:quiz_list')
    permission_required = 'quiz.delete_quiz'
    def get_queryset(self):
        user = self.request.user
        queryset = super().get_queryset()
        return queryset.filter(user=user)

class QuestionCreateView(PermissionRequiredMixin, CreateView):
    model = Question
    template_name = 'quiz/add_question.html'
    success_url = reverse_lazy('quiz:add_question')
    form_class = QuestionForm
    permission_required = 'quiz.add_question'
    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        user = self.request.user
        context.update({'questions': Question.objects.filter(quiz__user=user.pk)})
        return context

```

Б.12 Текст файла `urls.py` пакета `quiz`

```

from django.urls import path
from . import views
app_name = 'quiz'
urlpatterns = [path('list/', views.QuizListView.as_view(), name='quiz_list'),

```

```

        path('delete_quiz/<int:myid>', views.QuizDeleteView.as_view(),
name='delete_quiz'),
        path('<int:myid>/', views.quiz, name="quiz"),
        path('add_quiz/', views.QuizCreateView.as_view(), name='add_quiz'),
        path('add_question/', views.QuestionCreateView.as_view(),
name='add_question'), ]

```

Б.13 Текст файлу models.py пакета quiz

```

from django.db import models
from django.conf import settings
from django.core.validators import MinValueValidator
class Quiz(models.Model):
    title = models.CharField(verbose_name='назва', max_length=50)
    description = models.CharField(verbose_name='опис', max_length=500)
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE, verbose_name = 'користувач',)
    advert = models.ForeignKey('workspace.VacancyAdvert',
        on_delete=models.CASCADE,
        verbose_name = 'оголошення', null=True,)
    number_of_questions = models.IntegerField(verbose_name='кількість
питань',
        validators=[MinValueValidator(1)], default=1,)
    time = models.IntegerField(verbose_name='тривалість',
        validators=[MinValueValidator(1)],
        help_text='тривалість тесту в секундах', default=1,)
    class Meta:
        verbose_name = 'тест'
        verbose_name_plural = 'тести'
    def __str__(self): return self.title

```

```

def get_questions(self): return self.question_set.all()
class Question(models.Model):
    content = models.CharField(verbose_name='вміст', max_length=200)
    quiz = models.ForeignKey(Quiz, on_delete=models.CASCADE,
        verbose_name='тест',)
    class Meta:
        verbose_name = 'питання'
        verbose_name_plural = 'питання'
    def __str__(self): return self.content
    def get_answers(self): return self.answer_set.all()
class Answer(models.Model):
    content = models.CharField(verbose_name='вміст', max_length=200)
    correct = models.BooleanField(verbose_name='правильна відповідь?',
        default=False,)
    question = models.ForeignKey(Question, on_delete=models.CASCADE,
        verbose_name='питання',)
    class Meta:
        verbose_name = 'відповідь'
        verbose_name_plural = 'відповіді'
    def __str__(self): return f'питання: {self.question.content}, відповідь:
{self.content}, правильно: {self.correct}'
class MarksOfUser(models.Model):
    quiz = models.ForeignKey(Quiz, on_delete=models.CASCADE,
        verbose_name = 'тест')
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE, verbose_name = 'користувач',)
    score = models.FloatField(verbose_name='оцінка')
    class Meta:
        verbose_name = 'оцінка'
        verbose_name_plural = 'оцінки'

```

```
def __str__(self):return str(self.quiz)
```

Б.14 Текст файла forms.py пакета quiz

```
from django import forms
from .models import Quiz, Question
class QuizForm(forms.ModelForm):
    class Meta:
        model = Quiz
        fields = ('title', 'description','advert','number_of_questions', 'time')
class QuestionForm(forms.ModelForm):
    class Meta:
        model = Question
        fields = ('content', 'quiz',)
```

Б.15 Текст файла apps.py пакета quiz

```
from django.apps import AppConfig
class QuizConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "quiz"
```

Б.16 Текст файла admin.py пакета author

```
from django.contrib import admin
from .mixins import (AdvertMixin,)
from .models import (
    User,Course,Education,Language,WorkExperience, Company,)
@admin.register(User)
class UserAdmin(admin.ModelAdmin):pass
```

```

@admin.register(Course)
class CourseAdmin(AdvertMixin, admin.ModelAdmin):
    list_display = ('pk', 'course', 'completed', 'published', 'created')
    list_display_links = ('pk', 'course')
    fieldsets = ((None, {
        'classes': ('wide',), 'fields': ('course', 'completed', 'about', 'published',),}),
        ('Додаткова інформація', {'classes': ('collapse',), 'fields':
('user', 'created', 'updated',),}),))

    def get_fieldsets(self, request, obj):
        if not obj: fieldsets = ((None, {
            'classes': ('wide',), 'fields': ('course', 'completed', 'about', 'published',),}),)
        return fieldsets
    return super().get_fieldsets(request, obj)

@admin.register(Education)
class EducationAdmin(AdvertMixin, admin.ModelAdmin):
    list_display = ('pk', 'institution', 'specialty', 'completed', 'published', 'created')
    list_display_links = ('pk', 'institution', 'specialty')
    search_fields = ('institution', 'specialty', 'about')
    fieldsets = ((None, {'classes': ('wide',),
        'fields': ('institution', 'specialty', 'completed', 'about', 'published',),}),
        ('Додаткова інформація', {'classes': ('collapse',), 'fields':
('user', 'created', 'updated',),}),))

    def get_fieldsets(self, request, obj):
        if not obj: fieldsets = ((None, {'classes': ('wide',),
            'fields': ('institution', 'specialty', 'completed', 'about', 'published',),}),)
        return fieldsets
    return super().get_fieldsets(request, obj)

@admin.register(Language)
class LanguageAdmin(AdvertMixin, admin.ModelAdmin):
    list_display = ('pk', 'locale', 'interview', 'published', 'created')

```

```

list_display_links = ('pk', 'locale')
list_filter = ('locale', 'level', 'interview', 'published')
fieldsets = ((None, {'classes': ('wide',),
    'fields': ('locale', 'level', 'certificate', 'about', 'interview', 'published',),}),
    ('Додаткова інформація', {
        'classes': ('collapse',), 'fields': ('user', 'created', 'updated',),})),)

def get_fieldsets(self, request, obj):
    if not obj: fieldsets = ((None, {'classes': ('wide',),
        'fields': ('locale', 'level', 'certificate', 'about', 'interview', 'published',),}),)
    return fieldsets
    return super().get_fieldsets(request, obj)

@admin.register(WorkExperience)
class WorkExperienceAdmin(AdvertMixin, admin.ModelAdmin):
    list_display = ('pk', 'position', 'company', 'published', 'created')
    list_display_links = ('pk', 'position', 'company')
    search_fields = ('position', 'company', 'about')
    fieldsets = ((None, {'classes': ('wide', ), 'fields': ('position', 'company',
        ('started', 'finished'), 'about', 'published',),}),
        ('Додаткова інформація', {'classes': ('collapse',), 'fields':
('user', 'created', 'updated',),})))

def get_fieldsets(self, request, obj):
    if not obj: fieldsets = ((None, {'classes': ('wide',),
        'fields': ('position', 'company', ('started', 'finished'), 'about', 'published',),}),)
    return fieldsets
    return super().get_fieldsets(request, obj)

@admin.register(Company)
class CompanyAdmin(AdvertMixin, admin.ModelAdmin):
    list_display = ('pk', 'company', 'user', 'published', 'created')
    list_display_links = ('pk', 'company')
    fieldsets = ((None, {'classes': ('wide', ),

```

```

        'fields': ('company','about','published', 'meet_link', 'logo', 'business_card',),}),
        ('Додаткова інформація', {
            'classes': ('collapse',),'fields': ('user', 'created', 'updated',))}))
def get_fieldsets(self, request, obj):
    if not obj: fieldsets = ((None, {'classes': ('wide',),
        'fields': ('position', 'company','started', 'finished'), 'about', 'published',),}),)
    return fieldsets
    return super().get_fieldsets(request, obj)

```

Б.17 Текст файлу apps.py пакета author

```

from django.apps import AppConfig
class AuthorConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'author'

```

Б.18 Текст файлу forms.py пакета author

```

from django import forms
from allauth.account.forms import SignupForm
from allauth.account.adapter import get_adapter
from allauth.account.utils import (setup_user_email,)
from .models import (
    User, Course, Education, Language, WorkExperience,Company,)
class MyCustomSignupForm(SignupForm):
    type = forms.CharField(label='Роль',max_length=2,
        widget=forms.Select(choices=User.Type.choices),required=True,)
    def save(self, request):
        if self.account_already_exists:
            self._send_account_already_exists_mail(request)

```

```

        return
    adapter = get_adapter(request)
    user = adapter.new_user(request)
    user.type = self.cleaned_data.get('type')
    adapter.save_user(request, user, self)
    self.custom_signup(request, user)
    setup_user_email(request, user, [])
    return user

class CourseModelForm(forms.ModelForm):
    class Meta:
        model = Course
        fields = ('course', 'completed', 'about', 'published',)

class CompanyModelForm(forms.ModelForm):
    class Meta:
        model = Company
        fields = ('company', 'logo', 'about', 'meet_link', 'business_card',)

class EducationModelForm(forms.ModelForm):
    class Meta:
        model = Education
        fields = ('institution', 'specialty', 'completed', 'about', 'published',)

class LanguageModelForm(forms.ModelForm):
    class Meta:
        model = Language
        fields = ('locale', 'level', 'certificate', 'interview', 'about', 'published',)

class WorkExperienceModelForm(forms.ModelForm):
    class Meta:
        model = WorkExperience
        fields = ('position', 'company', 'started', 'finished', 'about', 'published',)

class UserModelForm(forms.ModelForm):
    class Meta:

```

```
model = User
fields = ('username', 'first_name', 'last_name', 'age', 'email', 'photo',)
```

Б.19 Текст файлу mixins.py пакета author

```
from django.urls import reverse
class AdvertMixin:
    list_editable = ('published',)
    readonly_fields = ('user', 'created', 'updated')
    save_as = True
    list_per_page = 10
    def save_model(self, request, obj, form, change):
        if not hasattr(obj, 'user'): obj.user = request.user
        return super().save_model(request, obj, form, change)
class ImmutableViewMixin:
    def get_success_url(self):
        user = self.request.user
        return reverse('author:profile', kwargs={'pk': user.pk})
class MutableViewMixin(ImmutableViewMixin):
    def get_queryset(self):
        user = self.request.user
        return super().get_queryset().filter(user=user)
```

Б.20 Текст файлу models.py пакета author

```
from ckeditor.fields import RichTextField
from django.conf import settings
from django.contrib.auth.models import AbstractUser
from django.core.validators import (MaxValueValidator, MinLengthValidator,
    MinValueValidator, FileExtensionValidator)
```

```

from django.db import models
from django.urls import reverse
from django.contrib.auth.models import Group
from django.db.models.signals import post_save
from django.dispatch import receiver
class User(AbstractUser):
    class Type(models.TextChoices):
        RECRUITER = 'RR', 'Рекрутер'
        APPLICANT = 'AA', 'Претендент'
    age = models.SmallIntegerField(verbose_name='вiк',
        validators=[MinValueValidator(18),MaxValueValidator(100)],
        blank=True,null=True,)
    photo = models.ImageField(verbose_name='аватар',
        upload_to='users/%Y/%m/%d',default='users/default.png',)
    type = models.CharField(verbose_name='роль',max_length=2,
        choices=Type.choices,default=Type.APPLICANT,)
    @property
    def is_recruiter(self):
        return True if self.type == self.Type.RECRUITER else False
    @property
    def is_applicant(self):
        return True if self.type == self.Type.APPLICANT else False
    def get_absolute_url(self):
        return reverse('author:profile', kwargs={"pk": self.pk})
    @receiver(post_save, sender=User)
    def add_group_user(sender, instance, created, **kwargs):
        group = None
        if created:
            if instance.type == User.Type.RECRUITER:
                try:group = Group.objects.get(name='Рекрутер')

```

```

        except Group.DoesNotExist:pass
    elif instance.type == User.Type.APPLICANT:
        try:group = Group.objects.get(name='Претендент')
        except Group.DoesNotExist:pass
    if group:group.user_set.add(instance)
class AbstractAdvert(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        verbose_name='користувач',)
    published = models.BooleanField(verbose_name='опубліковано',
        default=True,)
    created = models.DateTimeField(
        verbose_name='дата створення',auto_now_add=True,)
    updated = models.DateTimeField(
        verbose_name='дата оновлення',auto_now=True,)
    about = RichTextField(verbose_name='додаткова інформація',blank=True,)
    class Meta:
        abstract = True
class Course(AbstractAdvert):
    course = models.CharField(verbose_name='назва курсу',
        max_length=128,validators=[MinLengthValidator(10)],)
    completed = models.DateField(verbose_name='дата закінчення')
    class Meta:
        verbose_name = 'курс'
        verbose_name_plural = 'курси'
    def __str__(self):
        return self.course
class Education(AbstractAdvert):
    institution = models.CharField(verbose_name='навчальний заклад',
        max_length=128,validators=[MinLengthValidator(10)],)

```

```

specialty = models.CharField(verbose_name='факультет/спеціальність',
    max_length=128,validators=[MinLengthValidator(10)],)
completed = models.DateField(verbose_name='дата закінчення')
class Meta:
    verbose_name = 'освіта'
    verbose_name_plural = 'освіта'
class Language(AbstractAdvert):
    class Locale(models.TextChoices):
        ENGLISH = 'EN', 'Англійська'
        FRENCH = 'FR', 'Французька'
        GERMAN = 'DE', 'Німецька'
        POLISH = 'PL', 'Польська'
        UKRAINIAN = 'UK','Українська'
        HINDI = 'HI', 'Хінді'
        __empty__ = 'Мова'
    class Level(models.TextChoices):
        BEGINNER = 'A1', 'Початковий'
        INTERMEDIATE = 'B1', 'Середній'
        ADVANCED = 'C1', 'Просунутий'
        PROFICIENT = 'C2', 'Досконалий'
        __empty__ = 'Рівень володіння'
    locale = models.CharField(
        verbose_name='мова',max_length=2,choices=Locale.choices,)
    level = models.CharField(verbose_name='рівень володіння',
        max_length=2,choices=Level.choices,)
    certificate = models.CharField(verbose_name='назва сертифікату',
        max_length=128,blank=True,)
    interview = models.BooleanField(
        verbose_name='можу проходити співбесіду цією мовою',default=False,)
    class Meta:

```

```

verbose_name = 'мова'
verbose_name_plural = 'мови'
unique_together = ('locale', 'user')
def __str__(self): return self.locale
class WorkExperience(AbstractAdvert):
    position = models.CharField(verbose_name='посада',
                                max_length=64,validators=[MinLengthValidator(10)],)
    company = models.CharField(verbose_name='назва
компанії',max_length=64,
                                validators=[MinLengthValidator(10)],)
    started = models.DateField(verbose_name='початок роботи')
    finished = models.DateField(verbose_name='закінчення')
    class Meta:
        verbose_name = 'досвід'
        verbose_name_plural = 'досвід'
class Company(AbstractAdvert):
    company = models.CharField(verbose_name='назва компанії',
                                max_length=64,validators=[MinLengthValidator(10)],)
    user = models.OneToOneField(settings.AUTH_USER_MODEL,
                                on_delete=models.CASCADE,
                                verbose_name='користувач',primary_key=True,)
    logo = models.ImageField(verbose_name='логотип',
                                upload_to='users/companies/%Y/%m/%d',
                                default='users/companies/default.png',)
    meet_link = models.URLField(
        verbose_name='Посилання на відеоконференцію',blank=True,)
    business_card = models.FileField(verbose_name='візитка',blank=True,
                                      validators=[FileExtensionValidator(allowed_extensions=['mp4',])])
    class Meta:
        verbose_name = 'компанія'

```

```

verbose_name_plural = 'компанії'
def __str__(self): return self.company

```

Б.21 Текст файлу urls.py пакета author

```

from django.urls import path
from .views import (UserUpdateView, CompanyUpdateView,
    UserDeleteView, CourseCreateView,
    CourseUpdateView, CourseDeleteView,
    WorkExperienceCreateView,
    WorkExperienceUpdateView,
    WorkExperienceDeleteView,
    EducationCreateView, EducationUpdateView,
    EducationDeleteView, LanguageCreateView,
    LanguageUpdateView, LanguageDeleteView,
    CompanyCreateView, CompanyDeleteView,)
app_name = 'author'
urlpatterns = [
    path('account/profile/<int:pk>', UserUpdateView.as_view(), name='profile'),
    path('account/profile/delete/<int:pk>',
        UserDeleteView.as_view(), name='profile_delete'),
    path('account/profile/course/',
        CourseCreateView.as_view(), name='course_create'),
    path('account/profile/course/update/<int:pk>',
        CourseUpdateView.as_view(), name='course_update'),
    path('account/profile/course/<int:pk>',
        CourseDeleteView.as_view(), name='course_delete'),
    path('account/profile/workexperience/',
        WorkExperienceCreateView.as_view(), name='work_experience_create'),
    path('account/profile/workexperience/update/<int:pk>',

```

```

    WorkExperienceUpdateView.as_view(), name='work_experience_update'),
    path('account/profile/workexperience/<int:pk>',
        WorkExperienceDeleteView.as_view(),
        name='work_experience_delete'),
    path('account/profile/education/',
        EducationCreateView.as_view(), name='education_create'),
    path('account/profile/education/update/<int:pk>',
        EducationUpdateView.as_view(), name='education_update'),
    path('account/profile/education/<int:pk>',
        EducationDeleteView.as_view(), name='education_delete'),
    path('account/profile/language/',
        LanguageCreateView.as_view(), name='language_create'),
    path('account/profile/language/update/<int:pk>',
        LanguageUpdateView.as_view(), name='language_update'),
    path('account/profile/language/<int:pk>',
        LanguageDeleteView.as_view(), name='language_delete'),
    path('account/profile/company/',
        CompanyCreateView.as_view(), name='company_create'),
    path('account/profile/company/<int:pk>',
        CompanyDeleteView.as_view(), name='company_delete'),
    path('account/profile/company/update/<int:pk>',
        CompanyUpdateView.as_view(), name='company_update'),]

```

Б.22 Текст файла views.py пакета author

```

from django.views.generic import (CreateView, UpdateView, DeleteView,)
from django.contrib.auth.mixins import (
    LoginRequiredMixin, PermissionRequiredMixin,)
from django.urls import reverse
from .models import (User, Education, Language, WorkExperience,

```

```

    Course, Company,)
from .forms import (UserModelForm, CourseModelForm,
    EducationModelForm, LanguageModelForm,
    WorkExperienceModelForm, CompanyModelForm,)
from .mixins import (ImmutableViewMixin, MutableViewMixin,)
class UserUpdateView(LoginRequiredMixin, UpdateView):
    model = User
    form_class = UserModelForm
    def get_queryset(self):
        user = self.request.user
        return super().get_queryset().filter(pk=user.pk)
class UserDeleteView(LoginRequiredMixin, DeleteView):
    model = User
    def get_queryset(self):
        user = self.request.user
        return super().get_queryset().filter(pk=user.pk)
    def get_success_url(self): return reverse('workspace:home')
class CourseCreateView(PermissionRequiredMixin, ImmutableViewMixin,
CreateView):
    model = Course
    form_class = CourseModelForm
    template_name = 'author/model_form.html'
    permission_required = 'author.add_course'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context.update({'title': 'Додати Курс', 'legend': 'Курс', 'btn': 'Додати',})
        return context
    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)

```

```

class CourseUpdateView(PermissionRequiredMixin, ImmutableViewMixin,
UpdateView):
    model = Course
    form_class = CourseModelForm
    template_name = 'author/model_form.html'
    permission_required = 'author.change_course'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context.update({'title': 'Оновити Курс', 'legend': 'Курс', 'btn': 'Оновити',})
        return context

class CourseDeleteView(PermissionRequiredMixin, MutableViewMixin,
DeleteView):
    model = Course
    permission_required = 'author.delete_course'

class WorkExperienceCreateView(PermissionRequiredMixin,
ImmutableViewMixin, CreateView):
    model = WorkExperience
    form_class = WorkExperienceModelForm
    template_name = 'author/model_form.html'
    permission_required = 'author.add_workexperience'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context.update({
            'title': 'Додати Досвід Роботи', 'legend': 'Досвід Роботи', 'btn': 'Додати',})
        return context
    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)

class WorkExperienceUpdateView(PermissionRequiredMixin,
ImmutableViewMixin, UpdateView):

```

```

model = WorkExperience
form_class = WorkExperienceModelForm
template_name = 'author/model_form.html'
permission_required = 'author.change_workexperience'
def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context.update({
        'title': 'Оновити Досвід Роботи', 'legend': 'Досвід Роботи', 'btn':
'Оновити',})
    return context
class WorkExperienceDeleteView(PermissionRequiredMixin,
MutableViewMixin, DeleteView):
    model = WorkExperience
    permission_required = 'author.delete_workexperience'
class EducationCreateView(PermissionRequiredMixin, ImmutableViewMixin,
CreateView):
    model = Education
    form_class = EducationModelForm
    template_name = 'author/model_form.html'
    permission_required = 'author.add_education'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context.update({'title': 'Додати Освіту', 'legend': 'Освіта', 'btn': 'Додати',})
        return context
    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)
class EducationUpdateView(PermissionRequiredMixin, ImmutableViewMixin,
UpdateView):
    model = Education

```

```

form_class = EducationModelForm
template_name = 'author/model_form.html'
permission_required = 'author.change_education'
def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context.update({'title': 'Додати Освіту', 'legend': 'Освіта', 'btn': 'Додати', })
    return context
class EducationDeleteView(PermissionRequiredMixin, MutableViewMixin,
DeleteView):
    model = Education
    permission_required = 'author.delete_education'
class LanguageCreateView(PermissionRequiredMixin, ImmutableViewMixin,
CreateView):
    model = Language
    form_class = LanguageModelForm
    template_name = 'author/model_form.html'
    permission_required = 'author.add_language'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context.update({'title': 'Додати Мову', 'legend': 'Мова', 'btn': 'Додати', })
        return context
    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)
class LanguageUpdateView(PermissionRequiredMixin, ImmutableViewMixin,
UpdateView):
    model = Language
    form_class = LanguageModelForm
    template_name = 'author/model_form.html'
    permission_required = 'author.change_language'

```

```

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context.update({'title': 'Оновити Мову', 'legend': 'Мова', 'btn': 'Оновити', })
    return context

class LanguageDeleteView(PermissionRequiredMixin, MutableViewMixin,
DeleteView):
    model = Language

class CompanyCreateView(PermissionRequiredMixin, ImmutableViewMixin,
CreateView):
    model = Company
    form_class = CompanyModelForm
    template_name = 'author/model_form.html'
    permission_required = 'author.add_company'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context.update({'title': 'Додати Компанію', 'legend': 'Компанія', 'btn':
'Додати', })
        return context
    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)

class CompanyUpdateView(PermissionRequiredMixin, ImmutableViewMixin,
UpdateView):
    model = Company
    form_class = CompanyModelForm
    template_name = 'author/model_form.html'
    permission_required = 'author.change_company'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

```

```

        context.update({'title': 'Оновити Компанію', 'legend': 'Компанія', 'btn':
'Оновити',})
    return context

class CompanyDeleteView(PermissionRequiredMixin, MutableViewMixin,
DeleteView):
    model = Company

```

Б.23 Текст файлу apps.py пакета webchat

```

from django.apps import AppConfig
class WebchatConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'webchat'

```

Б.24 Текст файлу models.py пакета webchat

```

from django.db import models
from django.conf import settings
class Room(models.Model):
    title = models.CharField(verbose_name='назва', max_length=128)
    user = models.OneToOneField(settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
verbose_name='користувач',primary_key=True,)
    users = models.ManyToManyField(settings.AUTH_USER_MODEL,
        related_name='+',verbose_name='користувачі',)
    class Meta:
        verbose_name = 'кімната'
        verbose_name_plural = 'кімнати'
    def __str__(self):
        return self.title

```

ДОДАТОК В
Слайди презентації

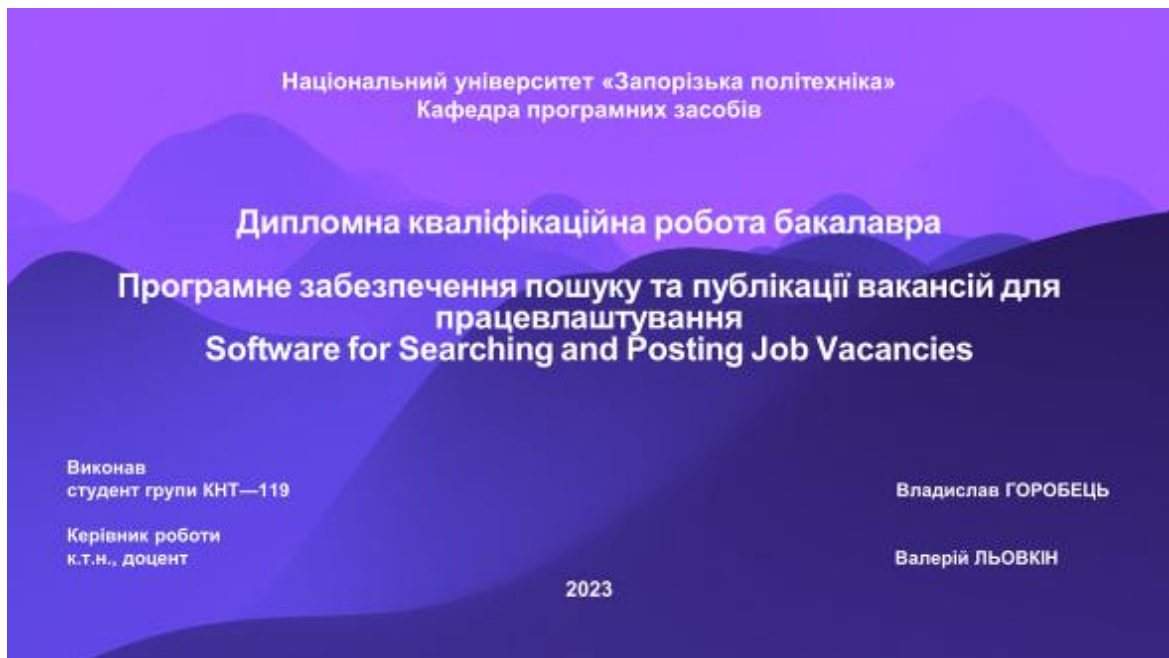


Рисунок В.1 – Слайд № 1

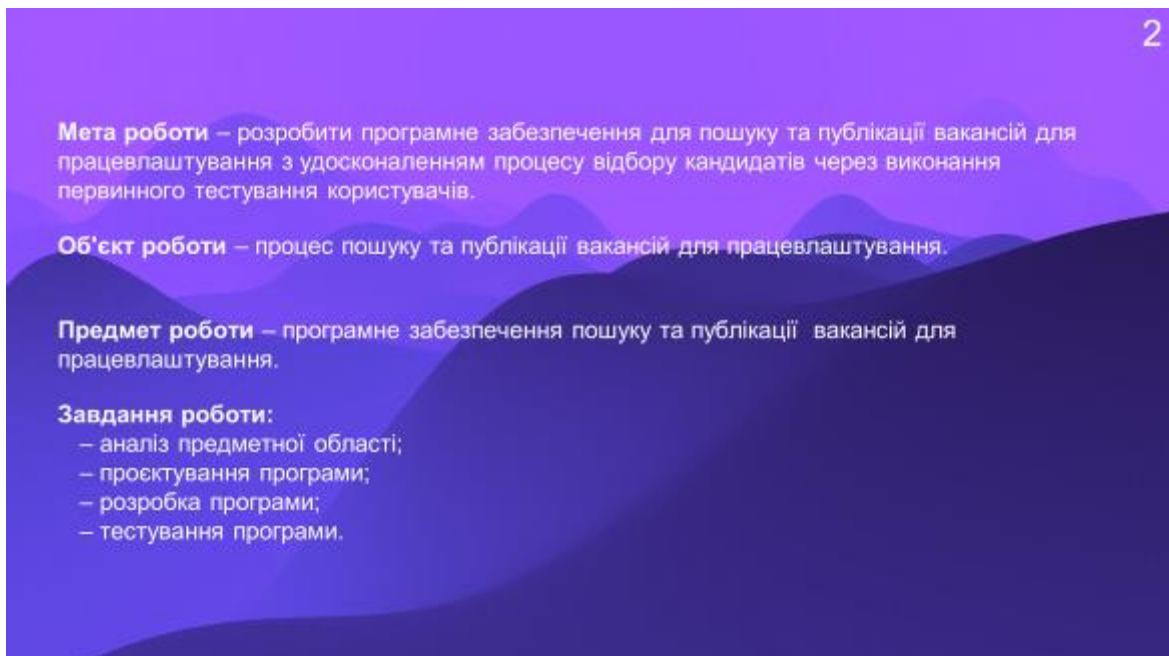


Рисунок В.2 – Слайд № 2

3

Порівняння програмних аналогів

Платформа	Robota.ua	Work.ua	Jobs.ua	Trud.ua	YettiJob
Категорія сайту	Пошук роботи	Пошук роботи	Пошук роботи	Пошук роботи	Пошук роботи
Реєстрація користувачів	+	+	-	-	+
Первинне тестування кандидатів	-	-	-	-	+
Наявність рекламних оголошень	+	+	+	+	-
Реєстрація / авторизація через GitHub, GitLab	-	-	-	-	+
Платні функції	+	+	+	+	-

Рисунок В.3 – Слайд № 3

4

Вибір мови програмування для розробки

Критерій	Python	PHP	Go
Тип мови	Мова загального призначення	Мова для веброзробки	Мова загального призначення
Підтримка Unicode	+	+/-	+
Підтримка об'єктно-орієнтованого програмування	+	+/-	+/-
Підтримка функціонального програмування	+	+/-	+/-
Підтримка асинхронного програмування	+	+/-	+
Наявність фреймворків для створення вебзастосунків	+	+	+
Наявність бібліотек для роботи з зображеннями	+	+/-	-
First Class Function	+	+	-
Анонімні функції	+	+	-
Замикання	+	+	-
Garbage Collector	+	+/-	+

Рисунок В.4 – Слайд № 4

5

Вибір фреймворку для розробки

Критерій	Django	Flask
Мова програмування	Python	Python
Безпека	+	+/-
Адміністративний інтерфейс	+	-
Роутінг	+	+/-
ORM	+	-
Керування	+	-

Рисунок В.5 – Слайд № 5

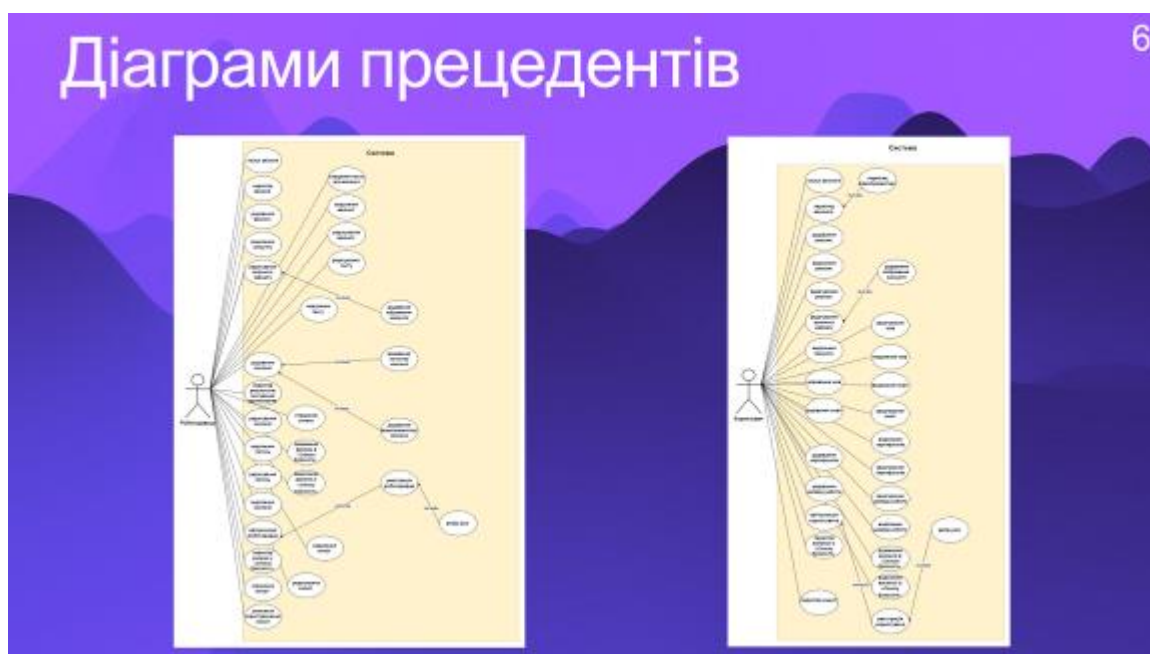


Рисунок В.6 – Слайд № 6

Схема бази даних

7

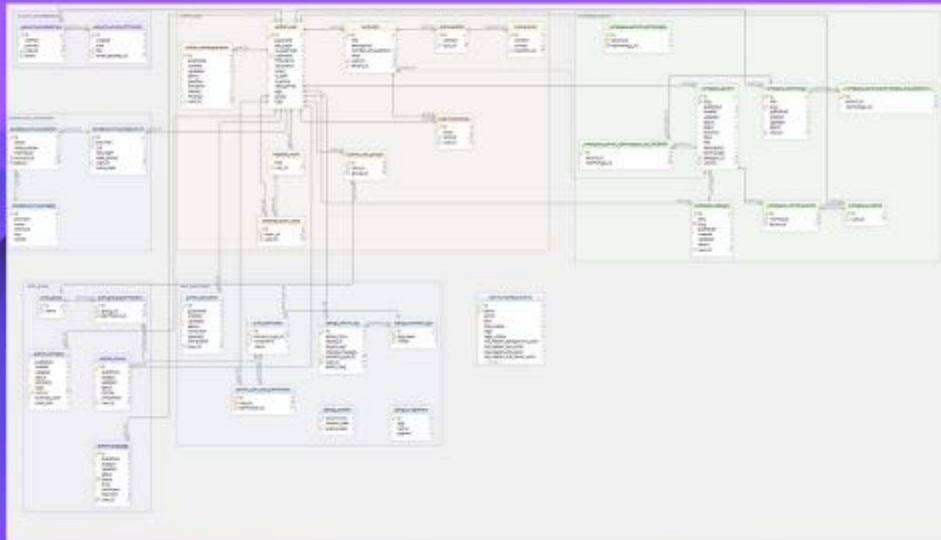


Рисунок В.7 – Слайд № 7

Шаблон проектування Model-View-Template

8



Рисунок В.8 – Слайд № 8

Схема функціонування програми ⁹

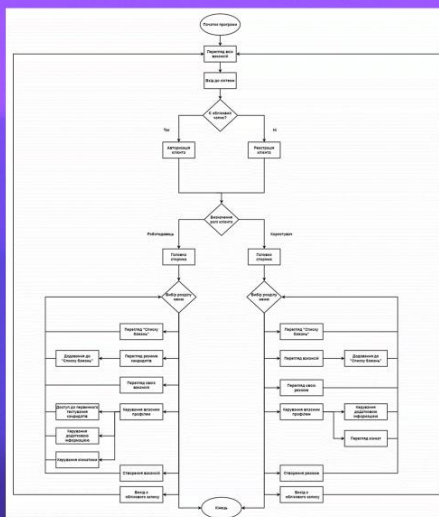


Рисунок В.9 – Слайд № 9

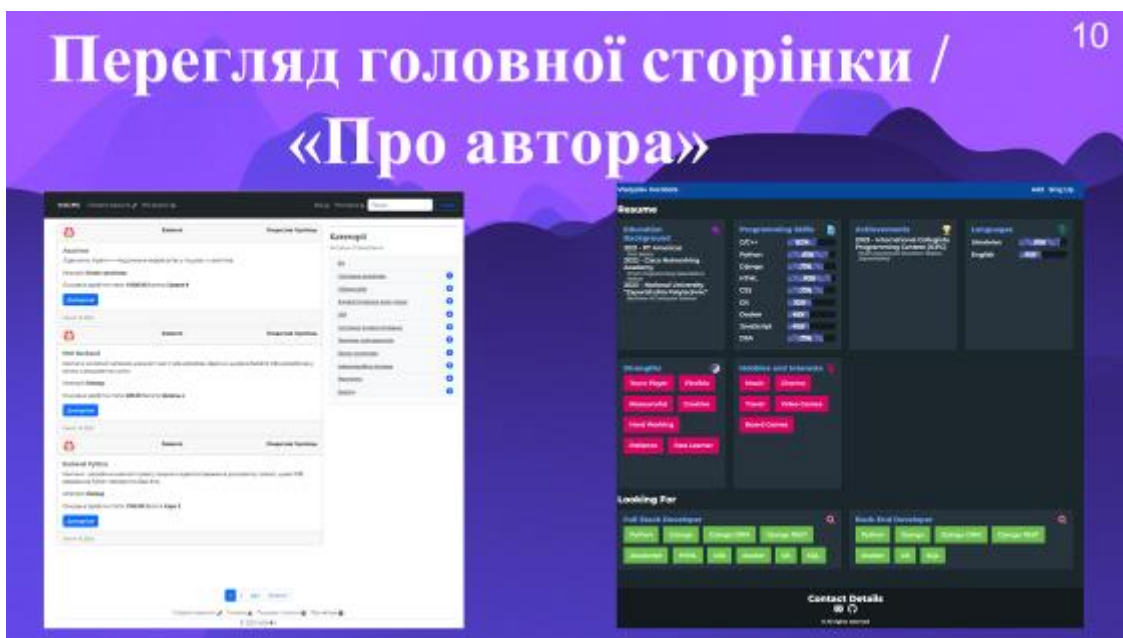


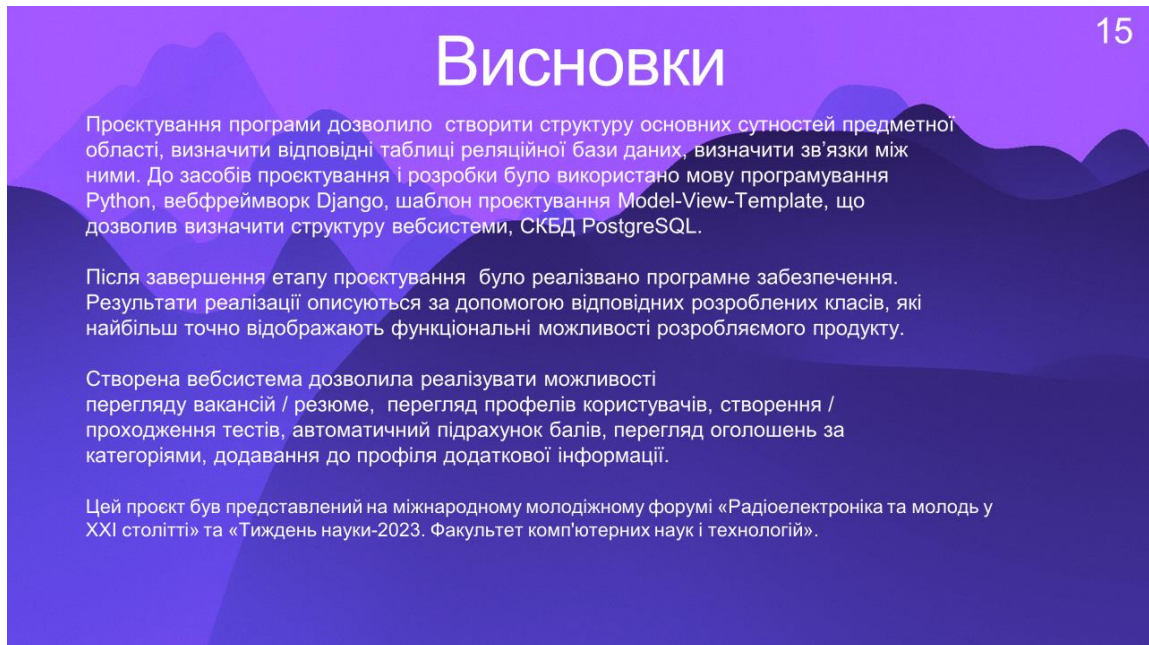
Рисунок В.10 – Слайд № 10

Пройдення тестування / перегляд профілю

Рисунок В.12 – Слайд № 12

[illegible]

Рисунок В.14 – Слайд № 14



Висновки

15

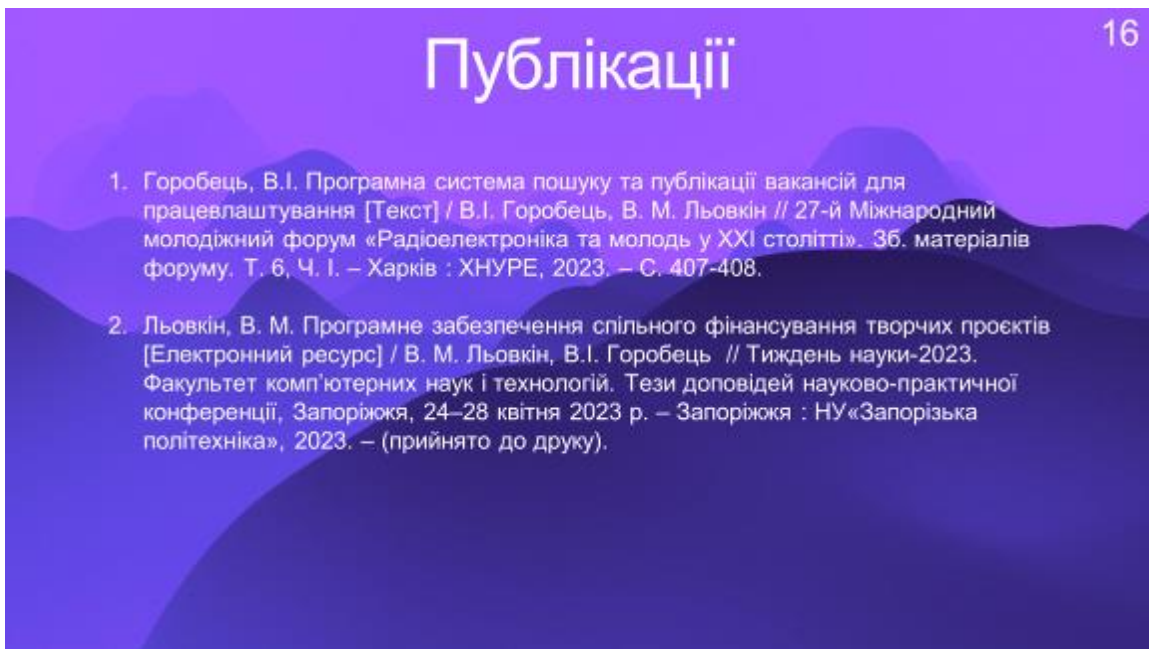
Проектування програми дозволило створити структуру основних сутностей предметної області, визначити відповідні таблиці реляційної бази даних, визначити зв'язки між ними. До засобів проектування і розробки було використано мову програмування Python, вебфреймворк Django, шаблон проектування Model-View-Template, що дозволив визначити структуру вебсистеми, СКБД PostgreSQL.

Після завершення етапу проектування було реалізовано програмне забезпечення. Результати реалізації описуються за допомогою відповідних розроблених класів, які найбільш точно відображають функціональні можливості розробляемого продукту.

Створена вебсистема дозволила реалізувати можливості перегляду вакансій / резюме, перегляд профелів користувачів, створення / проходження тестів, автоматичний підрахунок балів, перегляд оголошень за категоріями, додавання до профіля додаткової інформації.

Цей проект був представлений на міжнародному молодіжному форумі «Радіoeлектроніка та молодь у XXI столітті» та «Тиждень науки-2023. Факультет комп'ютерних наук і технологій».

Рисунок В.15 – Слайд № 15



Публікації

16

1. Горобець, В.І. Програмна система пошуку та публікації вакансій для працевлаштування [Текст] / В.І. Горобець, В. М. Льовкін // 27-й Міжнародний молодіжний форум «Радіoeлектроніка та молодь у XXI столітті». 36. матеріалів форуму. Т. 6, Ч. І. – Харків : ХНУРЕ, 2023. – С. 407-408.
2. Льовкін, В. М. Програмне забезпечення спільного фінансування творчих проєктів [Електронний ресурс] / В. М. Льовкін, В.І. Горобець // Тиждень науки-2023. Факультет комп'ютерних наук і технологій. Тези доповідей науково-практичної конференції, Запоріжжя, 24–28 квітня 2023 р. – Запоріжжя : НУ«Запорізька політехніка», 2023. – (прийнято до друку).

Рисунок В.16 – Слайд № 16