

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Веб-сайт пошуку роботи»

Виконав студент групи ІПЗк-20-1
МИКОЛЮК Владислав Олександрович

Керівник роботи:
МІНТІЙ Ірина Сергіївна

Консультант:
ЧИЖМОТРИЯ Олексій Володимирович

Рецензент:
КРАВЧЕНКО Світлана Миколаївна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **МИКОЛЮК Владислав Олександрович**

Керівник роботи: **МІНТІЙ Ірина Сергіївна**

Тема роботи: **«Веб-сайт пошуку роботи»,**

затверджена Наказом закладу вищої освіти від «14» лютого 2023 р. №70с

Вихідні дані для роботи: **використання веб-орієнтованих технологій для створення сайту пошуку роботи**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О. В.	16.02.2023р.	16.02.2023р.
2	Чижмотря О. В.	16.02.2023р.	16.02.2023р.
3	Чижмотря О. В.	16.02.2023р.	16.02.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	16 лютого 2023 – 1 березня 2023	Виконано
2	Проектування структури системи	2 березня 2023 - 9 березня 2023	Виконано
3	Написання програмного коду	10 березня - 1 квітня 2023	Виконано
4	Тестування	2 квітня 2023 – 7 квітня 2023	Виконано
5	Аналіз одержаних результатів	8 квітня 2023 – 10 квітня 2023	Виконано
6	Оформлення пояснювальної записки	11 квітня 2023 – 4 травня 2023	Виконано
7	Попередній захист	5 травня 2023	Виконано

Реферат

Випускна кваліфікаційна робота складається з веб-сайту пошуку роботи та пояснювальної записки. Пояснювальна записка до випускної роботи містить 93 сторінки, 20 ілюстрацій та 1 таблицю, список літературних джерел містить 17 найменувань.

Метою кваліфікаційної роботи є розробка веб-сайту пошуку роботи, за допомогою якого користувачі зможуть ефективно знаходити вакансії та розміщувати свої резюме. Сайт надає можливість шукачам роботи створювати облікові записи, заповнювати детальні профілі, переглядати та відгукатися на вакансії, а також отримувати рекомендації на основі їхніх вподобань та навичок. Роботодавці можуть реєструватися на сайті, створювати вакансії, переглядати резюме кандидатів та здійснювати відповідні наймальні дії. Основною метою є забезпечення зручного та ефективного процесу пошуку та працевлаштування для обох сторін - шукачів роботи та роботодавців.

У роботі визначено основні завдання на розробку веб-сайту, проаналізовано аналоги. Наведено сценарії роботи веб-сайту, загальну структуру, опис алгоритмів.

КЛЮЧОВІ СЛОВА: ВЕБ-САЙТ, BOOTSTRAP, LARAVEL, PHP.

					ІПЗ.КР.Б – 121 – 23 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		В.О. Миколюк			Пояснювальна записка «Веб-сайт пошуку роботи»			Літ.	Арк.	Аркуші	
Керівник		І.С.Мінтій								4	93
Консульт.		О.В. Чижморя						Житомирська політехніка група ІПЗк-20-1			
Рецензент.		С.М. Кравченко									
Зав. каф.		А.В. Морозов									

Abstract

The graduation thesis consists of a job search website and an explanatory memorandum. The explanatory memorandum for the graduation thesis includes 93 pages, 20 illustrations, and 1 table. The list of literary sources contains 17 references.

The aim of this diploma project is to develop a job search website that allows users to efficiently find job vacancies and post their resumes. The website provides job seekers with the ability to create accounts, fill out detailed profiles, browse and apply for job vacancies, and receive recommendations based on their preferences and skills. Employers can register on the website, create job postings, view candidate resumes, and take appropriate hiring actions. The main goal is to ensure a convenient and effective job search and employment process for both job seekers and employers.

The project defines the main tasks for the development of the website, analyzes existing analogues, and presents the website's workflows, overall structure, and algorithmic descriptions.

Keywords: Website, Bootstrap, Laravel, PHP.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ САЙТУ ПОШУКУ РОБОТИ	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту.....	11
1.3. Вибір архітектури веб-сайту пошуку роботи.....	14
1.4. Інструментальні засоби реалізації програмного продукту.....	17
Висновки до розділу 1	21
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-САЙТУ ПОШУКУ РОБОТИ	24
2.1. Визначення варіантів використання системи.....	24
2.2. Розробка бази даних системи.....	28
2.3. Проектування та реалізація бази даних роботи системи	35
2.4. Реалізація веб-додатку	38
Висновки до розділу 2	50
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-САЙТОМ ПОШУКУ РОБОТИ	51
3.1. Порядок встановлення та налаштування системи	51
3.2. Структура інтерфейсу веб-сайту пошуку роботи	53
3.3. Тестування роботи веб-сайту	64
Висновки до розділу 3	66
ВИСНОВКИ.....	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	69
ДОДАТКИ.....	71

Вступ

Актуальність теми: У сучасному світі зміни технологій і зростання конкуренції на ринку праці ставлять перед працівниками і роботодавцями нові завдання та вимоги. Для багатьох людей пошук роботи є дуже складним завданням, яке може зайняти багато часу і зусиль. Відповідно до цього, інтернет-ресурси, спрямовані на допомогу в пошуку роботи, зарекомендували себе як зручний та ефективний інструмент у розв'язанні даної проблеми.

Метою даної кваліфікаційної роботи є: розробка веб-сайту пошуку роботи, який надасть можливість користувачам знаходити відповідні пропозиції про роботу, а роботодавцям знайти відповідних працівників. Завдання полягає в створенні зручного інтерфейсу, що дасть змогу швидко та легко здійснювати пошук вакансій та резюме, а також внесенні інноваційних рішень для покращення функціональності сайту.

Об'єктом дослідження є: розробка, проектування та реалізація веб-сайту пошуку роботи.

Предметом дослідження є: використання веб-орієнтованих технологій для створення сайту пошуку роботи.

Методи дослідження: в роботі було використано методи об'єктно орієнтованого проектування, веб-програмування та дизайну

Перший розділ кваліфікаційної роботи присвячений теоретичним основам розробки веб-сайту пошуку роботи. В ньому проводиться аналіз інформаційної бази та літературних джерел, визначаються основні завдання впровадження інформаційних технологій за напрямком дослідження, проводиться аналіз існуючих аналогів та визначається архітектура розроблюваного додатку.

Другий розділ розкриває питання проектування та розробки веб-сайту пошуку роботи. В ньому визначаються функціональні вимоги до сайту,

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

описується процес проектування та розробки додатку, надає визначення функціональних вимог до сайту, а також описується етап проектування та розробки додатку. Окрім того, будуть розглянуті технології, які використовуються для розробки веб-додатків, та обґрунтування вибору конкретного інструментарію для створення сайту пошуку роботи.

Другий розділ також містить опис архітектури веб-додатку, його функціональних можливостей та особливостей реалізації. Тут буде розглянуто питання, пов'язані з використанням баз даних для зберігання та обробки інформації, а також інших технологій, необхідних для правильної роботи сайту пошуку роботи.

В третьому розділі наведено діаграми розгортання та компонентів системи, які спрощують розуміння архітектури та функціональності веб-сайту пошуку роботи. Ці діаграми дають змогу легше сприймати, як система функціонує та які компоненти відповідають за які аспекти роботи веб-сайту пошуку роботи.

Окрім цього, в третьому розділі було наведено структуру інтерфейсу користувача та адміністратора. Проведено тестування та виправлено помилки.

Пояснювальна записка до випускної роботи містить 93 сторінки, 20 ілюстрацій та 1 таблицю, список літературних джерел містить 17 найменувань.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ САЙТУ ПОШУКУ РОБОТИ

1.1. Постановка задачі

Постановка задачі полягає у конкретизації цілей і завдань проекту, які необхідно вирішити для створення сайту пошуку роботи на Laravel. Основна мета полягає у створенні ефективного і зручного інструменту для роботодавців та шукачів, який дозволить їм знайти взаємовигідні робочі пропозиції або кандидатів для вакансій.

В рамках дослідження та розробки цього сайту, поставлено наступні конкретні задачі:

1. Розробка функціональності пошуку роботи: Сайт повинен мати зручний і ефективний механізм пошуку, що дозволить роботягам знайти релевантні вакансії залежно від їхніх навичок, досвіду роботи та інших параметрів. Також необхідно розробити інструмент для роботодавців, що дозволить їм знаходити потенційних кандидатів, враховуючи вимоги до вакансій.
2. Оптимізація та розширення функціоналу: Важливим аспектом є оптимізація сайту, забезпечення його швидкодії та ефективності. Також враховуючи розвиток сфери працевлаштування, сайт повинен мати можливість розширення функціоналу та адаптації до нових вимог та технологій.
3. Реалізація системи авторизації та профілів користувачів: Для забезпечення безпеки та ідентифікації користувачів, необхідно розробити систему авторизації. Крім того, кожен користувач повинен мати свій особистий профіль, де він зможе вказати свої дані, навички, досвід та іншу інформацію, яка буде використовуватись при пошуку вакансій або кандидатів.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

4. Розробка адміністративної панелі: Сайт повинен мати адміністративну панель, яка дозволить модераторам керувати вакансіями, користувачами, здійснювати модерацію та надавати підтримку користувачам.
5. Реалізація системи фільтрації результатів пошуку: Додати можливість фільтрування результатів пошуку за різними критеріями, такими як розташування, категорія, ключові слова тощо. Це дозволить користувачам отримувати більш точні та релевантні результати.
6. Розробка системи сповіщень та підписок: Додати можливість користувачам підписуватись на отримання сповіщень про нові вакансії, що відповідають їхнім критеріям пошуку. Розробити механізм відправки сповіщень через електронну пошту.

Не функціональні вимоги до сайту - це вимоги, що не стосуються безпосередньо функціональності системи, але визначають його якості, характеристики та параметри використання. Основна відмінність між функціональними та не функціональними вимогами полягає в тому, що функціональні вимоги описують, що система повинна робити, а не функціональні - як вона повинна це робити [\[2\]](#).

Не функціональні вимоги включають:

1. Надійність: Вимоги до стабільності, надійності та доступності системи. Наприклад, максимальний час відновлення після збою, частота виникнення помилок, можливість резервного копіювання даних та відновлення роботи системи.
2. Безпека: Вимоги до захисту конфіденційності, цілісності та доступності даних. Наприклад, необхідність шифрування даних, обмеження доступу до конфіденційної інформації, захист від несанкціонованого доступу та зламу системи.
3. Сприйняття: Час відповіді системи для запитів не повинен перевищувати 10 секунд

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

Результатом виконання поставленого завдання буде розроблений функціональний сайт, який забезпечує можливість користувачам шукати роботу, переглядати вакансії, створювати профілі та надсилати резюме. Сайт повинен мати зручний та інтуїтивно зрозумілий інтерфейс, добре організовану навігацію та ефективний пошуковий механізм.

1.2. Аналіз аналогів програмного продукту

Для порівняння веб-сайтів, які займаються пошуком роботи в Україні, буде корисним провести аналіз наступних сайтів: rabota.ua, work.ua.

Rabota.ua - це один з провідних українських сайтів пошуку роботи, який відрізняється від інших подібних сайтів своїм дизайном та функціоналом. Сайт використовує великий список фільтрів для пошуку вакансій, що дозволяє користувачам швидко та ефективно знайти відповідну роботу. Робота.ua надає різноманітні можливості пошуку роботи, моніторинг змін у вакансіях, повідомлення про нові вакансії тощо [8].

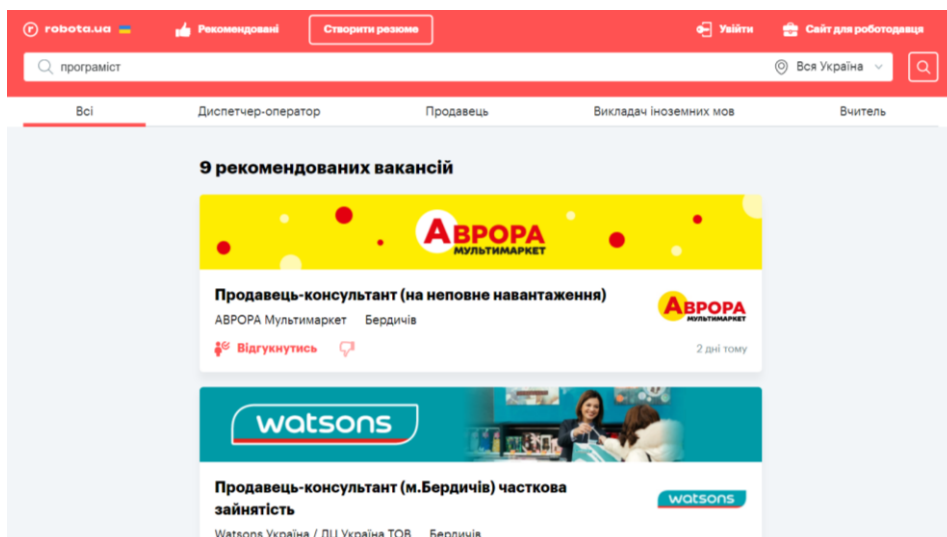


Рисунок 1.1 - Інтерфейс rabota.ua

Основні переваги сайту rabota.ua з точки зору програміста:

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

- Технічна підтримка та оновлення: Сайт регулярно оновлюється, додавання нових функцій та покращення вже існуючих. Також, на сайті працює ефективна технічна підтримка, що дозволяє вчасно вирішувати будь-які технічні проблеми, які користувачі можуть зустріти.
- Простота та зручність користування: Сайт має інтуїтивно зрозумілий та зручний інтерфейс, що робить його простим та зручним у використанні для користувачів. Також, на сайті працює швидкий та ефективний пошук вакансій.

Недоліки сайту robota.ua з точки зору програміста:

- Швидкість роботи сайту. На деяких пристроях веб-сайт завантажується дуже повільно. Це може бути зв'язано з недостатнім оптимізуванням коду та зображень.
- Робота з мобільними пристроями. Не всі сторінки сайту адаптовані для роботи з мобільними пристроями. Наприклад, деякі форми для заповнення заявок можуть бути складні для використання на маленьких екранах.
- Навігація. Деякі користувачі можуть знайти навігацію сайту не зовсім зручною та не інтуїтивно зрозумілою. Це може призвести до того, що користувачі не зможуть знайти потрібну їм інформацію.
- Не ефективна фільтрація вакансій. Система фільтрації вакансій може бути не дуже ефективною. Наприклад, відсутність фільтрування за заробітною платою може зробити пошук вакансій для деяких користувачів складнішим.
- Відсутність можливості збереження пошукових запитів. На сайті відсутня можливість збереження пошукових запитів, що може бути корисним для користувачів, які регулярно шукають вакансії в певних категоріях.
- Неоптимальна робота пошукової системи. Пошукова система може бути не дуже точною та неоптимальною. Наприклад, можуть з'являтися зайві

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

вакансії в результаті пошуку, або ж потрібна вакансія може не відображатись в результаті пошуку.

Work.ua - це один з найбільших сайтів пошуку роботи в Україні. Сайт пропонує користувачам широкий вибір вакансій, можливість створення резюме та розміщення оголошень про пошук роботи.

Однією з переваг work.ua є його зручний інтерфейс, що дозволяє швидко знайти необхідну інформацію та взаємодіяти з сайтом. Дизайн сайту мінімалістичний і чистий, що робить його зрозумілим і легким у використанні.

Work.ua також має добре організовану базу даних вакансій та резюме, що дозволяє зручно знаходити необхідну інформацію. Сайт пропонує широкий вибір фільтрів для пошуку вакансій, таких як регіон, галузь, рівень зарплати та інші [9].

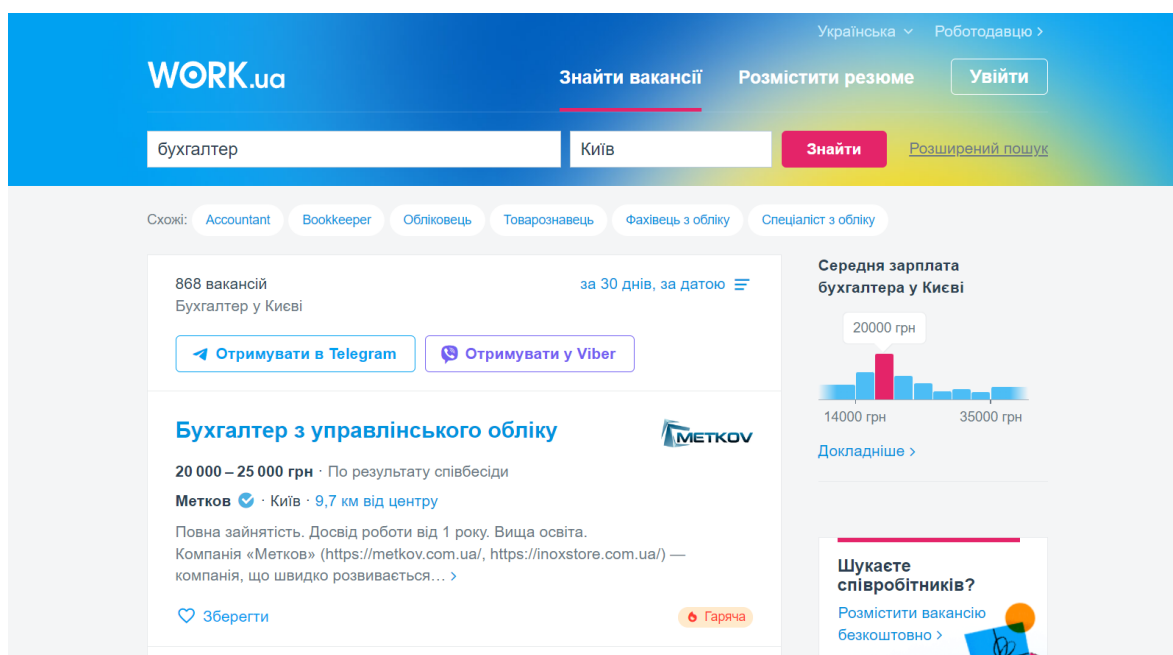


Рисунок 1.2 - Інтерфейс work.ua

Оцінка work.ua з точки зору програміста може бути наступною:

Плюси:

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

- Модерні технології: сайт work.ua використовує останні версії фреймворків і бібліотек JavaScript, що дозволяє розробникам створювати якісний інтерфейс і забезпечувати високу продуктивність сайту.
- Можливість додавання вакансій і резюме: сайт надає можливість додавати вакансії та резюме, що дозволяє користувачам бути в постійному пошуку роботи і перевіряти наявні вакансії.

Мінуси:

- Спам: в деяких випадках на сайті може бути багато небажаних повідомлень та спаму, що може стати дратівливим для користувачів.
- Відсутність регулярних оновлень: сайт може не оновлюватись регулярно, що може призвести до того, що він стане менш привабливим для користувачів.

Зважаючи на проведений аналіз сайтів work.ua та rabota.ua, було зроблено висновок, що розроблений веб-сайт пошуку роботи має ряд переваг. По-перше, сайт має інтуїтивно зрозумілий та легкий у використанні інтерфейс, що дозволить користувачам швидко знаходити необхідну інформацію про вакансії та роботодавців. По-друге, сайт надає можливість користувачам здійснювати пошук вакансії не лише серед тих що є на сайті, а й на інших сайтах України. По-третє, розроблений сайт має можливість відстежувати статус заявок на роботу, що дозволить користувачам контролювати процес пошуку роботи та вести переговори з роботодавцями у зручний для них час. Отже, розроблюваний веб-сайт пошуку роботи має конкурентні переваги порівняно зі своїми аналогами, що зробить його бажаним інструментом для пошуку роботи для багатьох користувачів.

1.3. Вибір архітектури веб-сайту пошуку роботи

Для успішного розроблення сайту пошуку роботи використання відповідної архітектури є важливим кроком. Під час вибору архітектурного

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

рішення було розглянуто кілька варіантів, включаючи Model-View-Controller (MVC), RESTful API та інші шаблони проектування.

MVC є одним з популярних шаблонів проектування, який забезпечує чітке розділення логіки бізнес-логіки, представлення даних та обробку запитів. Його переваги полягають у легкій модульності і розширюваності коду, покращеній повторній використовуваності та можливості паралельної роботи декількох розробників. Однак, також існує потенційний недолік у вигляді можливості створення занадто великих контролерів та складності управління великими проектами [1].

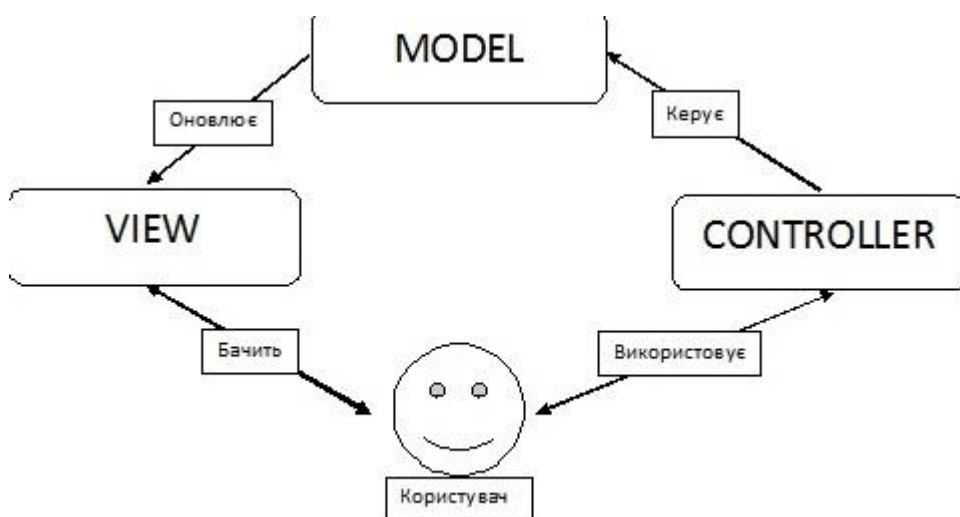


Рисунок 1.3 - Схема архітектури MVC

RESTful API є стандартним підходом для розробки веб-служб, який спрощує взаємодію з зовнішніми додатками або сервісами. Ця архітектура забезпечує незалежність між клієнтом та сервером, дозволяючи їм розвиватися незалежно. Вона також підтримує різні формати даних, такі як JSON або XML, і має вбудовану підтримку кешування для зниження навантаження на сервер. Однак, робота з RESTful API вимагає додаткових зусиль для визначення і налаштування маршрутів та управління HTTP-запитами [7].

Крім того, також було розглянуто інші архітектурні підходи, такі як розширена шарова архітектура за допомогою сервісів, що дозволяють створювати окремі класи для виконання конкретних завдань. Цей підхід полегшує керування кодом та забезпечує його модульність і перевикористовуваність.

Урахування особливостей кожної архітектури та їх плюсів та мінусів дозволить ефективно побудувати сайт пошуку роботи, забезпечити його масштабованість, модульність та підтримку у майбутньому.

MVC має багато переваг, які роблять його відмінним вибором для реалізації веб-сайту пошуку роботи в рамках кваліфікаційної роботи:

1. Підтримка повторного використання коду: Оскільки MVC розділяє логіку додатку на компоненти, це дозволяє легко перевикористовувати код. Модель може бути використана безпосередньо в інших проектах, представлення може бути повторно використане для відображення даних в різних контекстах, а контролер може мати загальні функціональність, яка може бути використана в інших частинах проекту.
2. Покращена тестованість: Оскільки логіка додатку розділена на компоненти, кожен з них може бути протестований окремо. Це дозволяє проводити модульне тестування моделі, представлення та контролера без необхідності тестувати всю систему в цілому. Це полегшує виявлення помилок та забезпечує вищу якість програмного забезпечення.
3. Підтримка паралельної розробки: Завдяки розділенню відповідальностей, розробка може відбуватись паралельно. Команда може працювати над моделлю, представленням та контролером незалежно одна від одної, що полегшує процес розробки та зменшує ймовірність конфліктів у коді.
4. Розширюваність: MVC надає гнучкість та легкість розширення функціональності. Оскільки кожен компонент має чітко визначені

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

обов'язки, додавання нових функцій або модифікація існуючих стає простішим і не впливає на решту системи.

Таким чином, використання патерна проектування MVC у кваліфікаційній роботі обґрунтовано його перевагами, такими як розділення відповідальностей, підтримка повторного використання коду, полегшення тестування, підтримка паралельної розробки та розширюваність. Це допоможе створити структуровану, гнучку та легко розширювану систему пошуку роботи.

1.4. Інструментальні засоби реалізації програмного продукту

Існують різні мови програмування для веб-розробки, які використовуються для роботи з інтернет-технологіями. Деякі з них спеціально призначені для створення веб-додатків, тоді як інші можуть використовуватися для роботи з будь-якими типами ресурсів. Деякі мови, такі як PHP, набули популярності та широкого визнання протягом тривалого часу.

Мови програмування для веб-розробки можна поділити на дві основні групи: клієнтські та серверні. Клієнтські мови, такі як HTML, CSS та JavaScript, використовуються для розробки веб-інтерфейсу, який відображається у веб-браузері користувача. HTML використовується для створення структури та розмітки веб-сторінок, CSS - для оформлення та стилізації цих сторінок, а JavaScript - для динамічного та інтерактивного взаємодії з користувачем [\[10\]](#).

Серверні мови програмування, які включають PHP, Python, Ruby та інші, використовуються для обробки запитів від клієнта, взаємодії з базами даних та формування відповідей, які надсилаються на клієнтський браузер. Ці мови дозволяють розробникам створювати потужні веб-додатки зі складною логікою та взаємодією з даними.

Кожна з цих мов має свої особливості, переваги та недоліки. Вибір мови програмування залежить від конкретних вимог проекту, рівня досвіду

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

розробника та специфіки задачі. Правильний вибір мови програмування допомагає забезпечити ефективну та масштабовану розробку веб-додатків.

1.4.1. PHP

PHP (Hypertext Preprocessor) є однією з найпопулярніших мов програмування для веб-розробки. Вона була спеціально створена для розробки динамічних веб-сторінок і взаємодії з базами даних [\[14\]](#). Ось деякі ключові особливості мови PHP для веб-розробки:

1. Синтаксис: Синтаксис PHP схожий на синтаксис C, що робить його зрозумілим для багатьох програмістів. PHP-код вбудовується безпосередньо в HTML-сторінки, що дозволяє комбінувати статичний контент з динамічним.
2. Розширюваність: PHP має велику кількість розширень і бібліотек, які дозволяють розробникам розширити функціональність своїх додатків. Це означає, що можна використовувати готові рішення для роботи з електронною поштою, обробки зображень, генерації PDF-файлів та багатьох інших завдань.
3. Підтримка фреймворків: PHP має багато популярних фреймворків для веб-розробки, таких як Laravel, Symfony, CodeIgniter, Yii та інші. Ці фреймворки надають структуру, готові компоненти та набір інструментів для швидкої та ефективної розробки веб-додатків.
4. Платформна незалежність: PHP є платформно-незалежною мовою, що означає, що вона може працювати на різних веб-серверах, таких як Apache, Nginx, IIS та інші. Ви також можете розробляти на PHP в будь-якій операційній системі, такій як Windows, Linux або macOS.

Окрім плюсів, слід також перерахувати і мінуси:

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

1. Безпека: РНР в минулому мала проблеми з безпекою, існували вразливості та можливості для атак. Хоча з часом були внесені поліпшення і виправлення, все ж розробники повинні бути обережними та дотримуватися найкращих практик забезпечення безпеки.
2. Мовні особливості: РНР має деякі мовні особливості та конструкції, які можуть бути складними для розуміння та використання. Наприклад, деякі непослідовності у синтаксисі або варіативність назв функцій та параметрів можуть вимагати більшої уваги до деталей під час програмування.

1.4.2. Ruby

Ruby є мовою програмування, яка здобула популярність серед веб-розробників завдяки своїй простоті, елегантності та зосередженості на продуктивності розробки. Особливо популярним фреймворком для веб-розробки на Ruby є Ruby on Rails (часто називають просто Rails) [\[17\]](#).

Ось деякі ключові особливості мови програмування Ruby для веб-розробки:

1. Синтаксис: Ruby має приємний та зрозумілий синтаксис, що нагадує природну мову. Він підтримує використання динамічних методів та операторів, що спрощує розробку та робить код більш зрозумілим.
2. Об'єктно-орієнтовані можливості: Ruby є повністю об'єктно-орієнтованою мовою, де все є об'єктом. Вона підтримує класи, спадкування, поліморфізм та багато інших ООП-концепцій, що сприяють розробці простих і логічно структурованих програм.
3. Ruby on Rails: Ruby on Rails є потужним фреймворком для швидкої розробки веб-додатків. Він надає готові рішення та конвенції, що допомагають розробникам ефективно працювати та зосереджуватися на бізнес-логіці, а не на низькорівневих деталях. Rails має вбудовану

підтримку для маршрутизації, взаємодії з базами даних, кешування, автентифікації та багатьох інших функцій.

4. Багата екосистема: Ruby має широку спільноту розробників, що веде до наявності великої кількості готових бібліотек та пакетів, які значно спрощують розробку веб-додатків. Це дозволяє розробникам швидко та ефективно будувати функціональність, використовуючи вже існуючі рішення.
5. Масштабованість: Ruby та Rails підтримують горизонтальну масштабованість, що означає, що вони можуть легко обробляти зростаючі навантаження шляхом розподілу завдань на багато серверів.

Крім перерахованого вище переліку плюсів, варто також враховувати і вірогідні мінуси, деякі з яких значно впливають на загальну оцінку:

1. Швидкодія: у порівнянні з деякими іншими мовами програмування Ruby може бути менш ефективною з точки зору швидкодії та продуктивності. Це особливо помітно при обробці великого обсягу даних або вимогах до високої швидкості виконання.
2. Менша популярність та ресурси: у порівнянні з PHP або Python, Ruby має меншу спільноту розробників та менше кількості готових ресурсів, таких як фреймворки, бібліотеки або документація. Це може ускладнити пошук підтримки або вирішення проблем під час розробки.
3. Ресурсоємність: Ruby відома своєю високою вживаністю ресурсів, таких як пам'ять та процесорний час. Це може призводити до збільшеного споживання пам'яті та більшої потреби у потужних серверах для підтримки веб-додатків, особливо якщо вони мають велику кількість одночасних користувачів.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

1.4.3. Python

Python - це високорівнева, універсальна мова програмування, яка здобула значну популярність у сфері веб-розробки. Хоча Python і використовується в різних галузях, включаючи наукові дослідження та штучний інтелект, він також має багато фреймворків та інструментів, спеціально призначених для веб-розробки [16]. Ось деякі ключові особливості мови Python для веб-розробки:

1. Зручний синтаксис: синтаксис Python дуже зрозумілий та читабельний. Він нагадує англійську мову та має мінімальну кількість символів. Це робить код більш зрозумілим та підтримує швидшу розробку.
2. Швидкість розробки: завдяки зручному синтаксису, багатій екосистемі та фреймворкам, розробка веб-додатків на Python може бути швидкою та ефективною. Фреймворки, такі як Django, пропонують готові рішення для автентифікації, маршрутизації, роботи з базами даних та багато іншого.
3. Розширюваність та інтеграція: Python може легко інтегруватися з іншими мовами програмування, такими як C, C++, або Java. Це дозволяє використовувати швидкі низькорівневі бібліотеки для оптимізації певних частин веб-додатків.

Нарівні з плюсами, у мови програмування Python також є і мінуси, зокрема:

1. Доступ до бази даних: програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.
2. Помилки виконання: це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле

число, у майбутньому може містити рядок, що може призвести до помилок виконання.

3. Неєфективна для пам'яті: ця мова програмування використовує великий обсяг пам'яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам'яті.

Було обрано мову програмування PHP та фреймворк Laravel з кількох причин.

По-перше, PHP є однією з найпопулярніших мов програмування для веб-розробки. Вона має велику спільноту розробників, широкий вибір ресурсів та документацію, що робить її легкою для вивчення та використання.

По-друге, PHP дозволяє швидко розробляти веб-додатки завдяки своїй простоті та гнучкості. Вона має вбудовану підтримку взаємодії з базами даних, роботи з формами та обробки HTTP-запитів, що спрощує процес розробки.

По-третє, фреймворк Laravel є потужним інструментом для розробки веб-додатків на основі PHP. Він надає готові компоненти та структуру для швидкої розробки високоякісних додатків. Laravel пропонує зручний механізм маршрутизації, шаблонів, ORM для роботи з базами даних, аутентифікацію та авторизацію, що спрощує розробку складних функціональностей [\[15\]](#).

Крім того, Laravel має активну спільноту розробників, що забезпечує підтримку, оновлення та наявність розширень та пакетів для розширення можливостей фреймворку.

Вибір мови програмування PHP та фреймворку Laravel обґрунтовується їхньою популярністю, простотою використання, широкими можливостями та наявністю активної спільноти розробників. Ці фактори сприяють швидкому розробленню веб-додатків високої якості та забезпечують підтримку та розширення в майбутньому.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		22

Висновки до розділу 1

В першому розділі було поставлено технічне завдання, проаналізовані існуючі аналоги, такі як: robota.ua, work.ua. На основі аналізу було зроблено висновки, щодо функціоналу веб-сайту.

Для реалізації було вирішено використати такі інструментальні засоби: Laravel, базу даних MySQL, Bootstrap. Наведено їх короткий опис.

Було розглянуто існуючі архітектури та обрано патерн проектування MVC.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-САЙТУ ПОШУКУ РОБОТИ

2.1. Визначення варіантів використання системи

Діаграма використання є одним з ключових елементів моделювання вимог у розробці програмного забезпечення. Вона використовується для візуалізації та аналізу взаємодії між акторами (користувачами, системами або зовнішніми компонентами) та системою, яка розробляється.

Основна мета діаграми використання полягає в тому, щоб зрозуміти, як користувачі та інші актори взаємодіють з системою, які дії вони виконують та які результати отримують. Вона дозволяє розробникам програмного забезпечення та стейкхолдерам отримати уявлення про функціональність системи, її можливості та вимоги [\[12\]](#).

Діаграма використання допомагає виявити основні функціональність системи, ідентифікувати акторів та їх ролі, визначити вимоги до системи та зрозуміти потреби користувачів. Вона також може використовуватися для виявлення помилок або недоліків у проектуванні перед початком розробки.

Діаграма використання наведена на рисунку 2.1.

Вимоги користувачів

Усі користувачі мають доступ до наступних функцій:

1. Перегляд вакансій;
2. Фільтрація вакансій;
3. Пошук вакансій на сайті;
4. Пошук вакансій на інших сайтах України;
5. Перегляд детальної інформації про вакансії;
6. Можливість зареєструватись на сайті;

Авторизовані користувачі з правами «Пошукач» мають додатковий доступ до наступних функцій:

1. Вхід у вебзастосунок;
2. Створення резюме;
3. Редагування резюме;
4. Видалення резюме;
5. Додавати вакансії в обране;
6. Видаляти вакансії з обраного;
7. Подавати заявку на вакансію;
8. Відслідковувати статус заявки;

Авторизовані користувачі з правами «Роботодавець» мають додатковий доступ до наступних функцій:

1. Вхід у вебзастосунок;
2. Перегляд резюме;
3. Пошук резюме;
4. Фільтрація резюме;
5. Перегляд детальної інформації про резюме;
6. Створення вакансії;
7. Редагування вакансії;
8. Видалення вакансії;
9. Відслідковувати резюме, що були подані;

Внутрішні користувачі (адміністратори) мають доступ до наступних функцій:

1. Управління вмістом БД
2. Керування користувачами
3. Модерація вакансій та резюме
4. Налаштовування розсилки електронних листів

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		25

Не функціональні вимоги

1. Сприйняття

- Час, потрібний для навчання інструментами роботи з інформаційною системою для звичайних користувачів – 2-3 години, а для досвідчених – 1 година;
- Час відповіді системи для звичайних запитів не повинен перевищувати 5 секунд, а для більш складних запитів – 10 сек;
- Інтерфейс представлення ІС повинен бути інтуїтивно зручним для користувача та не вимагати від нього додаткової підготовки.

2. Надійність

- Доступність – час, потрібний для обслуговування системи не потрібен перевищувати 10% від загального часу роботи.
- Середній час безперервної роботи – 20 робочих днів.
- Максимальна норма помилок в роботі системи – 1 помилка на 1000 запитів користувачів.

3. Продуктивність

Система повинна підтримувати мінімум 100 одночасно працюючих користувачів, пов'язаних з спільною базою даних.

Аналіз користувацьких та не функціональних вимог дозволив виділити сутності, що забезпечать реалізацію програмного комплексу системи.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

має доступ до усіх функцій, якими може користуватись не зареєстрований користувач а також має можливість створювати, редагувати та видаляти вакансії. Роботодавець має доступ до бази резюме(тільки перегляд та фільтрація). Також роботодавець може відслідковувати подані резюме на його вакансію.

Користувач з правами «Модератор» має безпосередній доступ до бази даних. Модератор переглядає новостворені резюме та вакансії і перевіряє чи були дотримані правила заповнення. Якщо ці правила були порушено, то адміністратор видаляє резюме чи вакансію а користувач отримує сповіщення з причиною видалення.

2.2. Розробка бази даних системи

У Laravel бази даних створюються за допомогою міграцій. Міграції є потужним інструментом для керування структурою бази даних. Вони дозволяють розробникам описувати та зберігати структуру бази даних у вигляді коду PHP, що дозволяє легко створювати, змінювати та синхронізувати бази даних між різними оточеннями.

У міграції Laravel можна описувати структуру бази даних, включаючи таблиці, поля, індекси, зовнішні ключі та інші складові бази даних. Міграція описує потрібні зміни структури бази даних і дозволяє створювати, змінювати або видаляти таблиці та їхні складові за допомогою коду.

Спроектowana та розроблена база даних містить 10 таблиць. Враховуючи всі умови та вимоги до розроблюваного сервісу складено наступну базу даних (рис 2.2).

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

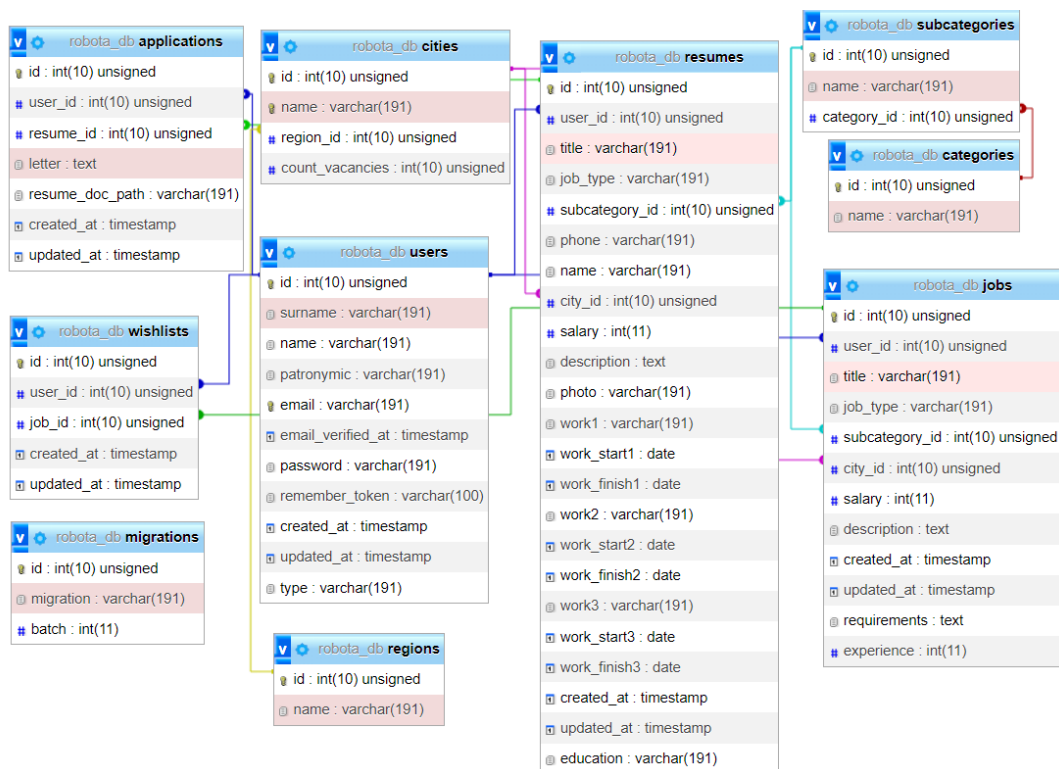


Рисунок 2.2 - Діаграма бази даних

Давайте розглянемо приклад міграції, яка створює таблицю 'jobs'. У цій таблиці ми зберігатимемо інформацію про вакансії. Таблиця міститиме стовбці, які посилаються на інші таблиці, щоб встановлювати зв'язок між різними елементами нашої бази даних. Давайте подивимося на код цієї міграції та його пояснення.

```

<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTableJobs extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('jobs', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('user_id')->unsigned();
            $table->foreign('user_id')->references('id')->on('users');
            $table->string('title');
        });
    }
}
  
```

```

        $table->string('job_type');
        $table->integer('subcategory_id')->unsigned();
        $table->foreign('subcategory_id')->references('id')-
>on('subcategories');
        $table->integer('city_id')->unsigned();
        $table->foreign('city_id')->references('id')->on('cities');
        $table->integer('salary');
        $table->text('requirements');
        $table->integer('experience');
        $table->text('description');
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('jobs');
}
}

```

Розглянемо пояснення кожного стовбця таблиці:

1. ***\$table->increments('id');*** - Цей стовбець називається 'id' і є первинним ключем таблиці. Він автоматично збільшується при додаванні нових записів.
2. ***\$table->integer('user_id')->unsigned();*** - Стовбець 'user_id' є зовнішнім ключем, який посилається на стовбець 'id' в таблиці 'users'. Він вказує на користувача, який створив цю вакансію.
3. ***\$table->string('title');*** Цей стовбець 'title' зберігає назву вакансії.
4. ***\$table->string('job_type');*** Стовбець 'job_type' визначає тип вакансії, наприклад, повна зайнятість або неповна зайнятість.
5. ***\$table->integer('subcategory_id')->unsigned();*** Стовбець 'subcategory_id' є зовнішнім ключем, який посилається на стовбець 'id' в таблиці 'subcategories'. Він вказує на підкатегорію, до якої належить ця вакансія.
6. ***\$table->integer('city_id')->unsigned();*** Стовбець 'city_id' є зовнішнім ключем, який посилається на стовбець 'id' в таблиці 'cities'. Він вказує на місто, де розташована ця вакансія.

7. ***\$table->integer('salary');*** Цей стовбець 'salary' зберігає заробітну плату, пов'язану з вакансією.
8. ***\$table->text('requirements');*** Стовбець 'requirements' зберігає вимоги до кандидатів, необхідні для даної вакансії.
9. ***\$table->integer('experience');*** Цей стовбець 'experience' зберігає необхідний досвід роботи для цієї вакансії.
10. ***\$table->text('description');*** Стовбець 'description' містить опис вакансії.
11. ***\$table->timestamps();*** Цей метод додає стовбці 'created_at' та 'updated_at', які автоматично зберігають дату та час створення та оновлення запису.

Ця міграція дозволяє нам створювати таблицю 'jobs' у нашій базі даних, з відповідними стовбцями для зберігання інформації про вакансії. У разі необхідності ми також можемо легко відкочувати міграцію, використовуючи метод 'down()', що дозволяє нам видалити таблицю 'jobs'.

Інші таблиці бази даних також будуть створені за допомогою міграцій.

Таблиця «users» - таблиця з інформацією про зареєстрованих користувачів сайту пошуку роботи. Містить наступні поля:

1. 'name' - рядкове поле, яке містить ім'я користувача.
2. 'surname' - рядкове поле, яке містить прізвище користувача.
3. 'patronymic' - рядкове поле, яке містить по батькові користувача.
4. 'email' - рядкове поле, яке містить електронну пошту користувача. Це унікальне поле, тобто кожна адреса повинна бути унікальною в межах таблиці.
5. 'email_verified_at' - поле типу timestamp, яке вказує на дату та час підтвердження електронної пошти користувачем. Це поле може мати значення null, якщо електронна пошта ще не була підтверджена.

6. 'password' - рядкове поле, яке містить захешований пароль користувача.
7. 'type' - рядкове поле, яке вказує тип користувача. Зазвичай це може бути значення "Роботодавець" або "Пошукач".
8. 'remember_token' - рядкове поле, яке використовується для "запам'ятовування" користувача під час аутентифікації.
9. 'created_at' і 'updated_at' - Поля типу timestamp, які автоматично зберігають дату та час створення та оновлення запису.

Таблиця «regions» - таблиця з назвами регіонів України. Містить наступні поля:

1. 'id': це стовбець називається 'id' і є первинним ключем таблиці. Він автоматично збільшується при додаванні нових записів.
2. 'name': це рядкове поле, яке містить назву регіону. Воно використовується для збереження назви конкретного регіону.

Таблиця «cities» - таблиця з назвами міст України. Містить наступні поля:

1. 'id': цей стовбець є первинним ключем таблиці. Він автоматично збільшується при додаванні нових записів.
2. 'name': це рядкове поле, яке містить назву міста. Кожне місто повинно мати унікальну назву, тому це поле встановлене як унікальне ('unique').
3. 'region_id': це цілочисельне поле, яке вказує на ідентифікатор регіону, до якого належить це місто. Це зовнішній ключ, який посиляється на стовбець 'id' в таблиці 'regions'.
4. 'count_vacancies': це цілочисельне поле, яке зберігає кількість вакансій, що доступні в цьому місті. За замовчуванням, воно встановлене на 0.

Таблиця «categories» - таблиця з назвами загальних категорій для резюме та вакансій.

1. 'id': цей стовбець є первинним ключем таблиці. Він автоматично збільшується при додаванні нових записів.
2. 'name': це рядкове поле, яке містить назву категорії.

Таблиця «subcategories» - таблиця з назвами підкатегорій для резюме та вакансій.

1. 'id': цей стовбець є первинним ключем таблиці. Він автоматично збільшується при додаванні нових записів.
2. 'name': це рядкове поле, яке містить назву підкатегорії.
3. 'category_id': це цілочисельне поле, яке вказує на ідентифікатор категорії, до якої належить ця підкатегорія. Це зовнішній ключ, який посилається на стовбець 'id' в таблиці 'categories'.

Таблиця «resumes» - це таблиця, яка містить записи про резюме створені користувачами-пошукачами.

1. 'id': це унікальний ідентифікатор резюме. Використовується як первинний ключ таблиці.
2. user_id: це цілочисельне поле, яке посилається на ідентифікатор користувача у таблиці "users". Воно використовує зовнішній ключ для зв'язку з таблицею "users".
3. title: це рядкове поле, яке містить заголовок резюме.
4. subcategory_id: це цілочисельне поле, яке посилається на ідентифікатор підкатегорії у таблиці "subcategories". Воно використовує зовнішній ключ для зв'язку з таблицею "subcategories".
5. name: це рядкове поле, яке містить ім'я.
6. city_id: це цілочисельне поле, яке посилається на ідентифікатор міста у таблиці "cities". Воно використовує зовнішній ключ для зв'язку з таблицею "cities".

7. salary: це цілочисельне поле, яке містить інформацію про заробітну плату. це поле може мати значення NULL, оскільки воно позначене як nullable.
8. description: це текстове поле, яке містить опис резюме.
9. photo: це рядкове поле, яке містить шлях до фотографії.
10. work1, work2, work3: ці рядкові поля містять інформацію про місце роботи. У таблиці є три таких поля.
11. work_start1, work_start2, work_start3: ці поля - дати, які вказують на початок роботи. В таблиці є три таких поля.
12. work_finish1, work_finish2, work_finish3: ці поля - дати, які вказують на закінчення роботи. В таблиці є три таких поля.
13. timestamps: це поле автоматично керує часом створення та оновлення запису. Включає два поля: created_at (дата та час створення запису) та updated_at (дата та час останнього оновлення запису).

Таблиця «applications» - це таблиця, яка містить записи про заявки пошукачів на отримання роботи. Містить наступні поля:

1. id: унікальний ідентифікатор заявки. Використовується як первинний ключ таблиці.
2. user_id: цілочисельне поле, яке посилається на ідентифікатор користувача у таблиці "users". Використовується як зовнішній ключ для зв'язку з таблицею "users".
3. resume_id: цілочисельне поле, яке посилається на ідентифікатор резюме у таблиці "resumes". Використовується як зовнішній ключ для зв'язку з таблицею "resumes".
4. letter: текстове поле, яке містить лист супроводу або додаткові коментарі до заявки. Це поле може мати значення NULL, оскільки воно позначене як nullable.

5. `resume_doc_path`: рядкове поле, яке містить шлях до документу резюме, який може бути прикріплений до заявки. Це поле також може мати значення NULL, оскільки воно позначене як nullable.
6. `timestamps`: поля, які автоматично керують датою та часом створення та оновлення запису. Включає два поля: `created_at` (дата та час створення запису) та `updated_at` (дата та час останнього оновлення запису).

Таблиця `wishlists` містить обрані вакансії для пошукача. Містить наступні поля:

1. `id`: унікальний ідентифікатор обраного. Використовується як первинний ключ таблиці.
2. `user_id`: цілочисельне поле, яке посилається на ідентифікатор користувача у таблиці `"users"`. Використовується як зовнішній ключ для зв'язку з таблицею `"users"`.
3. `job_id`: цілочисельне поле, яке посилається на ідентифікатор вакансії у таблиці `"jobs"`. Використовується як зовнішній ключ для зв'язку з таблицею `"jobs"`.

Проектування архітектури бази даних грає важливу роль у розробці проекту. Від цього етапу залежить зручність доступу до даних, швидкість виконання запитів і оптимальне використання ресурсів обраної бази даних.

2.3. Проектування та реалізація алгоритмів роботи системи

Діаграма послідовності є типом діаграми UML (Unified Modeling Language), яка використовується для візуалізації послідовності взаємодії між об'єктами або компонентами системи в часовому порядку. Вона детально описує, як об'єкти взаємодіють між собою, передаючи повідомлення та виконуючи певні дії [\[13\]](#).

Діаграми послідовності використовуються для опису алгоритмів роботи системи з точки зору взаємодії між її компонентами. Вони зосереджені на послідовності повідомлень та дій, які відбуваються між об'єктами в системі протягом певного часу. Це дозволяє розуміти порядок виконання операцій, обмін даними та залежності між компонентами системи.

Діаграми послідовності допомагають не тільки описати алгоритми роботи системи, але й виявити можливі проблеми або неузгодженості між компонентами. Вони також можуть бути використані для виявлення оптимальних шляхів оптимізації та покращення взаємодії між об'єктами.

Алгоритм рекомендацій є одним з найважливіших компонентів сайту пошуку роботи, оскільки він забезпечує користувачам персоналізовані та релевантні пропозиції вакансій. Цей алгоритм базується на аналізі даних з резюме шукачів роботи та вакансій з бази даних.

Основна мета алгоритму рекомендацій полягає в тому, щоб знайти найкращі вакансії для конкретного шукача роботи, враховуючи його вимоги, навички, досвід роботи та інші параметри. Цей алгоритм аналізує резюме шукача та враховує фактори, такі як підкатегорія роботи, місто, зарплата та досвід роботи, для знаходження найбільш підходящих вакансій.

На рисунку 2.3 зображено діаграму послідовності рекомендацій.



Рисунок 2.3 - Діаграма послідовності рекомендацій

Діаграма послідовності системи пошуку роботи ілюструє взаємодію різних компонентів системи під час процесу пошуку вакансій та розміщення резюме.

Перш за все, користувачі, які включають шукачів роботи та роботодавців реєструються в системі та взаємодіють з інтерфейсом користувача, де вони можуть здійснювати різні дії, такі як введення параметрів пошуку або розміщення вакансій.

Користувач модератор модерує вакансії та резюме. Якщо вони задовольняють правила, то модератор дозволяє їх публікацію. В іншому випадку модератор забороняє публікацію і відправляє користувачу повідомлення з причиною відмови. Якщо резюме шукача проходить перевірку та публікується, користувач може подати заявку на вакансію. Роботодавець може переглядати заявки та резюме шукачів. Діаграма послідовності системи пошуку роботи зображена на рисунку 2.4.

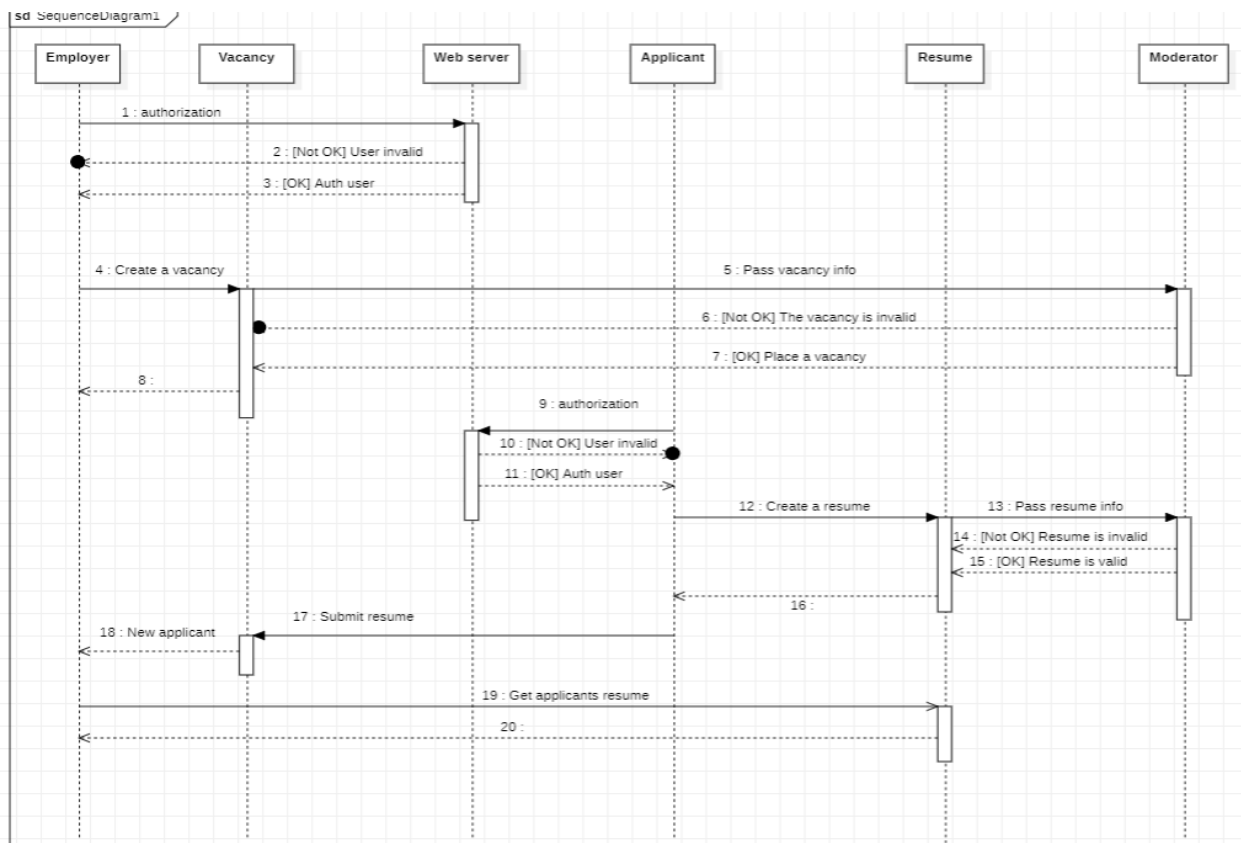


Рисунок 2.4 - Діаграма послідовності системи пошуку роботи

2.4. Реалізація веб-додатку

Реалізація веб-додатку на Laravel починається з налаштування з'єднання з базою даних в конфігураційному файлі `.env`. Встановлюємо настройки для з'єднання з базою даних: назва БД, користувач, пароль. Окрім цього в конфігураційному файлі налаштуємо наш сайт для роботи з сервісом «Sendgrid», який використовується для відправлення повідомлень на електронну пошту. Налаштування з'єднання з БД та з сервісом Sendgrid в конфігураційному файлі зображено на рисунку 2.5.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=robota_db
DB_USERNAME=root
DB_PASSWORD=

MAIL_DRIVER=smtp
MAIL_HOST=smtp.sendgrid.net
MAIL_PORT=587
MAIL_USERNAME=apikey
MAIL_PASSWORD=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
MAIL_ENCRYPTION=tls
```

Рисунок 2.5 - Конфігураційний файл .env

Laravel використовує патерн проектування MVC (Model-View-Controller), це означає, що логіка додатку буде розподілена між моделями, контролерами та шаблонами.

Модель в Laravel представляє бізнес-логіку та дані додатку. Вона взаємодіє з базою даних або будь-якими іншими джерелами даних, забезпечуючи методи для отримання, збереження, оновлення та видалення даних. Моделі зазвичай розташовуються у директорії app/Models. Laravel використовує ORM (Object-Relational Mapping) Eloquent, що дозволяє вам легко взаємодіяти з базою даних за допомогою PHP-класів [6]. Моделі також можуть включати валідацію даних та інші дії, пов'язані з бізнес-логікою.

Для кожної таблиці бази даних були створені моделі для взаємодії з ними. Створені моделі зображені на рисунку 2.6

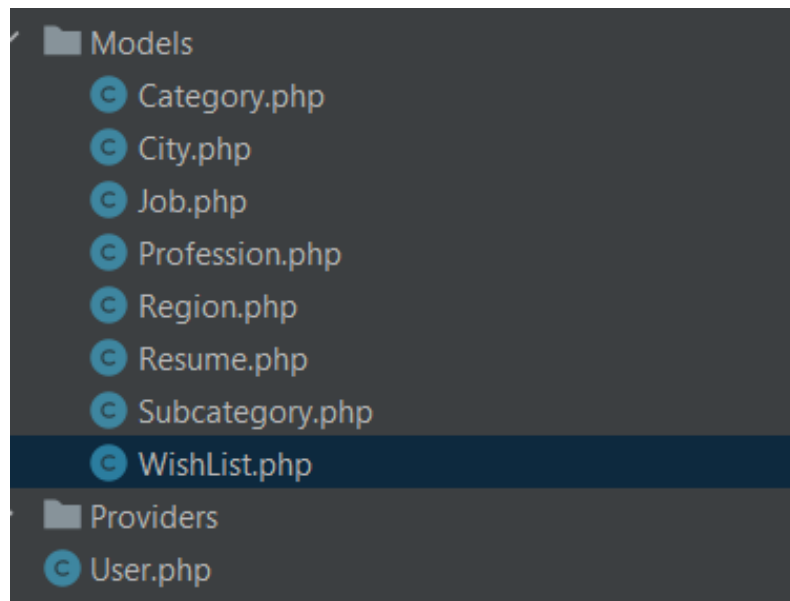


Рисунок 2.6 - Створені моделі

Модель Resume – це модель, що відповідає за дані, що знаходяться в таблиці resumes.

```
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Resume extends Model
{
    protected $table = 'resumes';

    protected $fillable =
    [
        'user_id',
        'title',
        'subcategory_id',
        'phone',
        'name',
        'city_id',
        'job_type',
        'education',
        'salary',
        'description',
        'photo',
        'work1',
        'work_start1',
        'work_finish1',
        'work2',
        'work_start2',
        'work_finish2',
        'work3',
        'work_start3',
        'work_finish3',
    ]
}
```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'created_at',
        'updated_at'
    ];

    public function subcategory()
    {
        return $this->hasOne('App\Models\Subcategory', 'id',
'subcategory_id');
    }

    public function city()
    {
        return $this->hasOne('App\Models\City', 'id', 'city_id');
    }

    public function user()
    {
        return $this->hasOne('App\User', 'id', 'user_id');
    }
}

```

Розглянемо цю модель:

- Модель Resume наслідує (розширює) клас Model фреймворку Laravel;
- Змінна \$table вказує, що ця модель взаємодіє з таблицею бази даних з назвою resumes;
- Масив \$fillable вказує які поля можуть бути заповнені масово. Це означає, що вказані поля можуть бути одночасно встановлені значеннями з масиву даних, переданих в метод create() – для створення нового запису в базі даних, а також update() – для зміни запису.
- Метод subcategory() визначає зв'язок один до одного (hasOne) з моделлю Subcategory. Зв'язок встановлюється через поле 'subcategory_id' моделі Resume та поле 'id' моделі Subcategory.
- Метод city() визначає зв'язок один до одного (hasOne) з моделлю City. Зв'язок встановлюється через поле 'city_id' моделі Resume та поле 'id' моделі City.
- Метод user() визначає зв'язок один до одного (hasOne) з моделлю User. Зв'язок встановлюється через поле user_id моделі Resume та поле id моделі User.

Контролер в Laravel відповідає за обробку запитів користувача та взаємодію з моделями та шаблонами. Він приймає запити, викликає необхідні методи моделей для отримання даних та передає ці дані до відповідних шаблонів для відображення [5]. Контролери зазвичай розташовуються у директорії “app/Http/Controllers”. Вони можуть мати різні методи для різних дій, таких як показ списку даних, створення нового запису, відображення конкретного запису тощо.

Деякі контролери може генерувати безпосередньо сам Laravel. До таких контролерів відносять контролери реєстрації та авторизації користувачів. Для цього в командному рядку в директорії проекту потрібно ввести команду `php artisan make:auth`. Після успішного виконання команди будуть автоматично створені наступні компоненти:

1. Маршрути для аутентифікації, включаючи реєстрацію, вхід, вихід та інші дії, які пов'язані з авторизацією користувачів;
2. Контролер `Auth\RegisterController`, який містить логіку реєстрації нових користувачів;
3. Контролер `Auth\LoginController`, який містить логіку входу користувачів у систему;
4. Шаблони у директорії `resources/views/auth`, які відображають форми реєстрації, входу, скидання та зміни пароля;

Розглянемо код контролера `RegisterController`

```
<?php
namespace App\Http\Controllers\Auth;

use App\User;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Illuminate\Foundation\Auth\RegistersUsers;

class RegisterController extends Controller
{
```

```

use RegistersUsers;

/**
 * Where to redirect users after registration.
 *
 * @var string
 */
protected $redirectTo = '/home';

/**
 * Create a new controller instance.
 *
 * @return void
 */
public function __construct()
{
    $this->middleware('guest');
}

/**
 * Get a validator for an incoming registration request.
 *
 * @param array $data
 * @return \Illuminate\Contracts\Validation\Validator
 */
protected function validator(array $data)
{
    return Validator::make($data, [
        'surname' => ['required', 'string', 'max:255'],
        'name' => ['required', 'string', 'max:255'],
        'patronymic' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255'],
        'unique:users',
        'password' => ['required', 'string', 'min:6', 'confirmed'],
    ]);
}

protected function create(array $data)
{
    return User::create([
        'surname' => $data['surname'],
        'name' => $data['name'],
        'patronymic' => $data['patronymic'],
        'email' => $data['email'],
        'type' => $data['type'],
        'password' => Hash::make($data['password']),
    ]);
}

public function showRegistrationForm()
{
    return view('auth.register');
}
}

```

Розглянемо що робить цей контролер:

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

1. Властивість `$redirectTo` - вказує, куди перенаправити користувача після успішної реєстрації;
2. `__construct()` - конструктор контролера, де встановлюється middleware 'guest', що означає, що лише незалоговані користувачі можуть мати доступ до методів цього контролера;
3. `validator(array $data)` - цей метод визначає правила валідації для даних реєстрації. Використовуючи фасад Validator, він перевіряє, чи введені дані відповідають заданим правилам;
4. `create(array $data)` - цей метод створює нового користувача після успішної валідації даних. Використовуючи метод `User::create()`, він створює новий запис у таблиці користувачів з введеними;
5. `showRegistrationForm()` - цей метод повертає вид `auth.register`, який відображає форму реєстрації для користувачів.

Структура папки `App\Http\Controllers` зображена на рисунку 2.7.

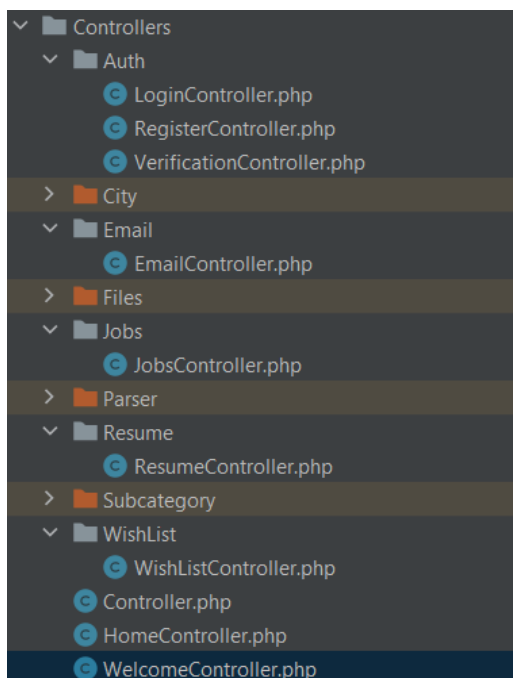


Рисунок 2.7 - Структура папки Controllers

В таблиці 2.1 наведено пояснення про контролери та їх основні задачі.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

Контролери та їх основні задачі

№	Назва контролера	Основні задачі
1	LoginController	Відповідає за авторизацію існуючих користувачів
2	RegisterController	Відповідає за реєстрацію нових користувачів та створення запису в таблиці users
3	JobsController	Відповідає за створення, редагування, видалення та відображення вакансій створених роботодавцем
4	ResumeController	Відповідає за створення, редагування, видалення та відображення резюме створених пошукачем
5	WishListController	Відповідає за створення, видалення та відображення обраних вакансій
6	WelcomeController	Відповідає за відображення контенту головної сторінки сайту
7	EmailController	Відповідає за надсилання повідомлень на електронну пошту користувачів
8	Controller	Базовий клас, від якого наслідуються всі інші контролери

Після створення моделей та контролерів необхідно створити маршрути.

Файл routes/web.php в Laravel використовується для визначення маршрутів (routes) для веб-додатку. У цьому файлі визначається, які дії (actions) повинні бути виконані при обробці запитів до різних URL-адрес.

```
<?php
use Illuminate\Support\Facades\Mail;
/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
```

```

| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', 'WelcomeController@index')->name('welcome');

#Registration and Auth
Route::group([], function()
{
    Route::get('register', 'Auth\RegisterController@showRegistrationForm')->name('showRegistrationForm');

    Route::post('register', 'Auth\RegisterController@register')->name('register');

    Route::get('login', 'Auth>LoginController@showLoginForm')->name('showLoginForm');

    Route::post('login', 'Auth>LoginController@login')->name('login');

    Route::get('logout', 'Auth>LoginController@logout')->name('logout');
});

Route::group(['prefix' => 'resume'], function()
{
    Route::get('/', 'Resume\ResumeController@index')->name('resume.index');

    Route::get('/create', 'Resume\ResumeController@create')->name('resume.create');

    Route::post('store', 'Resume\ResumeController@store')->name('resume.store');

    Route::get('show/{id}', 'Resume\ResumeController@show')->name('resume.show');

    Route::get('/my_resumes', 'Resume\ResumeController@my_resumes')->name('resume.my_resumes');

    Route::get('delete/{id}', 'Resume\ResumeController@delete')->name('resume.delete');

    Route::get('edit/{id}', 'Resume\ResumeController@edit')->name('resume.edit');
});

Route::group(['prefix' => "jobs"], function()
{
    Route::get('/', 'Jobs\JobsController@index')->name('jobs.index');

    Route::get('by-region', 'Jobs\JobsController@byRegion')->name('jobs.by-region');
}

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Route::get('create', 'Jobs\JobsController@create')->name('jobs.create');

Route::post('store', 'Jobs\JobsController@store')->name('jobs.store');

Route::get('show/{id}', 'Jobs\JobsController@show')->name('jobs.show');

Route::get('apply/{id}', 'Jobs\JobsController@showApplyForm')->name('jobs.showApplyForm');
});

Route::get('ukrainian-resources', function()
{
    return view('faq');
})->name('ukrainian-resources');

Route::group(['prefix' => 'wishlist'], function()
{
    Route::get('/', 'WishList\WishListController@index')->name('wishlist.index');

    Route::post('store', 'WishList\WishListController@store')->name('wishlist.store');

    Route::post('delete', 'WishList\WishListController@delete')->name('wishlist.delete');
});

```

Файл `routes/web.php` є важливим компонентом структури додатку Laravel, оскільки він визначає, як HTTP-запити будуть маршрутизовані до відповідних контролерів та дій. Коли користувач відкриває певну URL-адресу, Laravel перевіряє цей файл, щоб знайти відповідний маршрут і виконати відповідну дію (наприклад, виклик контролера та його методу).

Використання `routes/web.php` дозволяє логічно організувати маршрути додатку та забезпечити правильну обробку запитів.

Для надсилання електронних повідомлень про вакансії пошукачу та про заявки для роботодавця був використаний сервіс Sendgrid. Це хмарний сервіс електронної пошти, який надає інфраструктуру для надсилання і приймання електронних листів. Він дозволяє веб-додаткам надсилати електронну пошту з

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

високою доставкою та забезпечує потужні інструменти для керування електронними повідомленнями.

На рисунку 2.8 Зображено отриманий API-ключ Sendgrid

API Keys			Create API Key
NAME	API KEY	ACTION	
myapikey API Key ID: ykCVqE9zQliGv-EEWqoKg	*****	⚙️ ▼	
rabota_ztu_api_key API Key ID: ct04vlyTS70cnPAYGTtU9g	*****	⚙️ ▼	
myapikey API Key ID: LOfyGB1ZTTu6OtfYgkXXQ	*****	⚙️ ▼	
rabota_ztu_api_key API Key ID: f7zErZRBsuZol_2L9WGfxA	*****	⚙️ ▼	
rabota_ztu_api_key API Key ID: DYghobfSLQeaJHp4C-7BQ	*****	⚙️ ▼	
rabota_ztu_api_key API Key ID: BdNCJ14RQKWpCwCU4qs9oQ	*****	⚙️ ▼	

Рисунок 2.8 - Отриманий API-ключ Sendgrid

Після того, як отримали API-ключ, потрібно зробити налаштування в конфігураційному файлі .env. На рисунку 2.9 Зображено настройки конфігураційного файла для роботи з сервісом sendgrid.

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.sendgrid.net
MAIL_PORT=587
MAIL_USERNAME=apikey
MAIL_PASSWORD=S6.ykCVqE9zQliGv-EEWqoKg.iLQdMptvL2HoRn9JYEo-f0LnFlegQhAiEGxQ0c8RtS0
MAIL_ENCRYPTION=tls
```

Рисунок 2.9 - Настройки конфігураційного файла для роботи з сервісом sendgrid

Після настройки конфігураційного файлу необхідно створити функцію для надсилання електронного повідомлення.

```
$array =
[
    'name' => $data['name'],
    'job_title' => $data['title'],
    'job_description' => $data['description'],
```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		48

```

        'job_requirements' => $data['requirements'],
        'job_conditions' => $data['conditions'],
        'company_name' => $data['company_name'],
    ];

Mail::send('emails.test', $array, function($message)
{
    $message->to($data['email'])->from("goodprogrammer@ukr.")-
>subject("Welcome");
});

```

Використовується шаблон повідомлення. Код шаблону:

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Нова вакансія</title>
</head>
<body>
<p>Шановний(a) {{ $name }},</p>
<p>Ми раді повідомити вам про нашу нову вакансію:</p>
<ul>
    <li>Назва вакансії: {{ $job_title }}</li>
    <li>Опис: {{ $job_description }}</li>
    <li>Вимоги: {{ $job_requirements }}</li>
    <li>Умови: {{ $job_conditions }}</li>
</ul>
<p>Якщо ви зацікавлені у цій вакансії, будь ласка, зв'яжіться з нами.</p>
<p>З повагою,<br>
    {{ $company_name }}</p></body>
</html>

```

При успішному виконанні функції на електронну пошту користувача приходить повідомлення, що зображено на рисунку 2.10

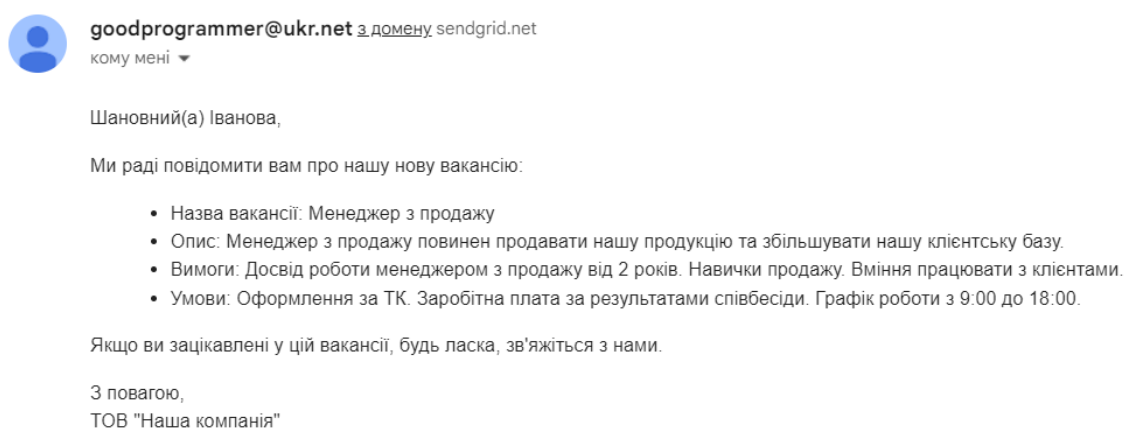


Рисунок 2.10 - Повідомлення про нову вакансію

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

Висновки до розділу 2

В даному розділі було спроектовано діаграми використання та діаграми послідовності. Проведено розробку загальної схеми роботи додатку. Розглянуто та спроектовано алгоритми роботи додатку.

Наведені деякі частини коду. Були створені такі можливості: створення та редагування резюме/вакансій, перегляд даних, перехід по сторінкам, вхід та реєстрація для користувача.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-САЙТОМ ПОШУКУ РОБОТИ

3.1. Порядок встановлення та налаштування системи

Діаграма розгортання в UML використовується для моделювання фізичного розгортання артефактів на вузлах системи. Вузли зображуються у вигляді прямокутних паралелепіпедів, в яких розташовані артефакти, що зображуються у вигляді прямокутників. Вузли можуть мати підвузли, які представляються як вкладені прямокутні паралелепіпеди.

Діаграма розгортання надає можливість візуалізувати фізичне розташування компонентів системи на фізичних вузлах, таких як сервери, комп'ютери або пристрої IoT. Кожен вузол може мати свої власні артефакти, такі як програми, файли, конфігураційні файли тощо, які виконуються або розгортаються на цьому вузлі.

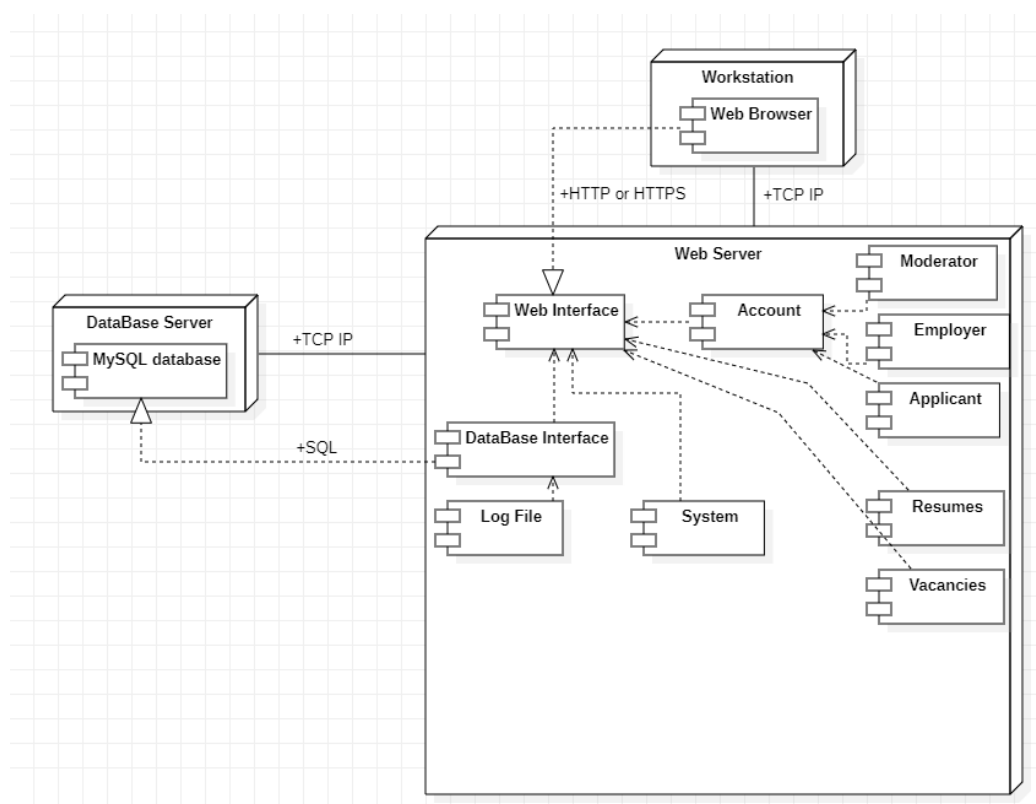


Рисунок 3.1 - Діаграма розгортання системи

Вузли:

- Database Server – сервер бази даних
- Workstation – персональна ЕОМ користувача
- Web Server – веб сервер

Діаграма компонентів дозволяє описати фізичне уявлення системи і визначити її архітектуру, встановлюючи залежності між компонентами програмного забезпечення. Цей графічний інструмент використовує компоненти, інтерфейси та залежності як основні елементи для відображення структури та взаємозв'язків системи [11].

Важливо відзначити, що діаграма компонентів є одним із ключових інструментів при розробці програмного забезпечення, оскільки вона дозволяє зрозуміти структуру системи та забезпечити її правильну організацію та функціонування.

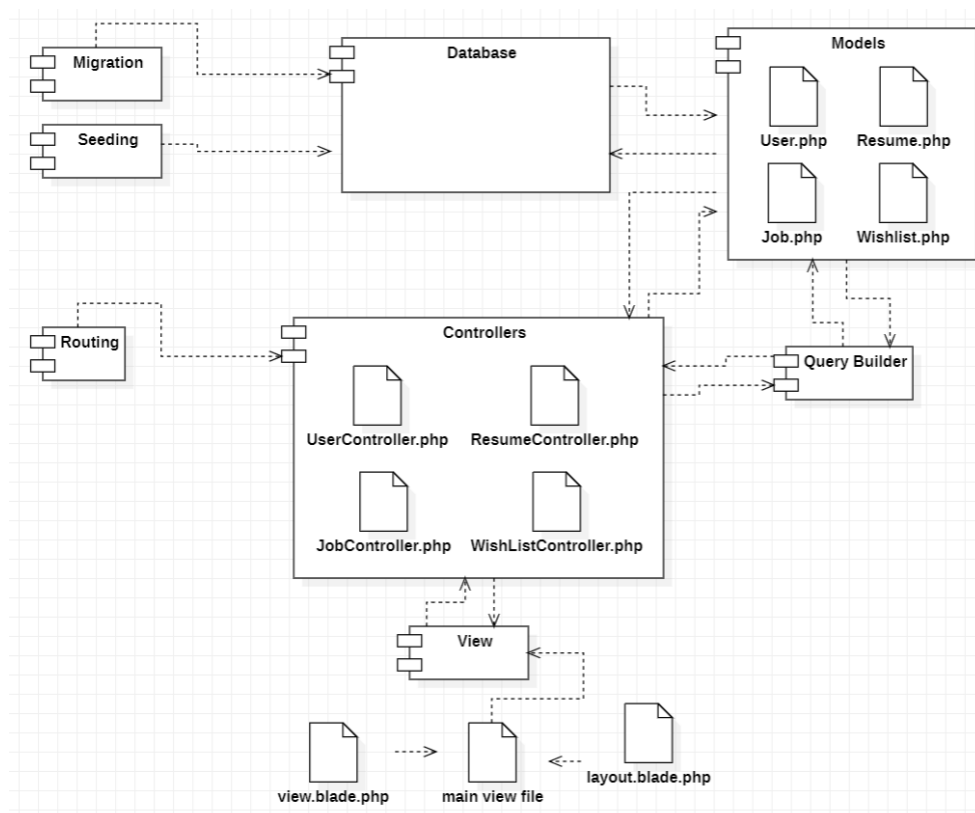


Рисунок 3.2 - Діаграма компонентів

На діаграмі компонентів зображено такі елементи:

1. Routing – маршрутизатор Laravel;
2. Database – база даних;
3. Migrations – спеціальні класи, що створюють структуру таблиць бази даних;
4. Seeding – спеціальний механізм заповнення бази даних початковими значеннями;
5. Models – моделі для доступу до таблиць бази даних.
6. Query Builder – спеціальний клас, який дозволяє будувати запити до бази даних;
7. Controllers - це класи, які відповідають за обробку вхідних запитів, координування взаємодії з моделями та відображенням результатів у відповідні вигляди (views);
8. View - це компонент, який відповідає за візуальне представлення даних інтерфейсу користувача, де відображаються дані з моделей та інші елементи інтерфейсу.

Використання діаграми компонентів дозволяє детально розглянути фізичну організацію системи та встановити взаємозв'язки між різними компонентами програмного забезпечення. Вона надає зрозуміле та графічне представлення архітектури системи, що допомагає розробникам та інженерам зрозуміти взаємодію між компонентами та забезпечити їх ефективну інтеграцію.

3.2. Структура інтерфейсу веб-сайту пошуку роботи

Спершу, що потрібно зробити – це перейти на сам сайт. Користувач має можливість переглядати контент сайту без реєстрації, але його можливості будуть обмежені. Користувач має можливість зареєструватись як роботодавець,

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

або як пошукач. Форма реєстрації має поля для вводу прізвища, ім'я, по-батькові, електронна пошта а також пароль з підтвердженням.

Створіть акаунт

Персональна інформація
[Вже маєте акаунт?](#)

Я роботодавець ▾

Прізвище
Прізвище

Ім'я
Ім'я

По-батькові
По-батькові

E-mail
E-mail

Пароль
Пароль

Повторіть пароль
Повторіть пароль

ЗАРЕЄСТРУВАТИСЬ

Рисунок 3.3 – Форма реєстрації

Після успішного входу здійснюється перенаправлення на головну сторінку сайту. На цій сторінці знаходиться форма пошуку вакансії (рисунок 3.4), а також список останніх доданих вакансій (рисунок 3.5).

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		54

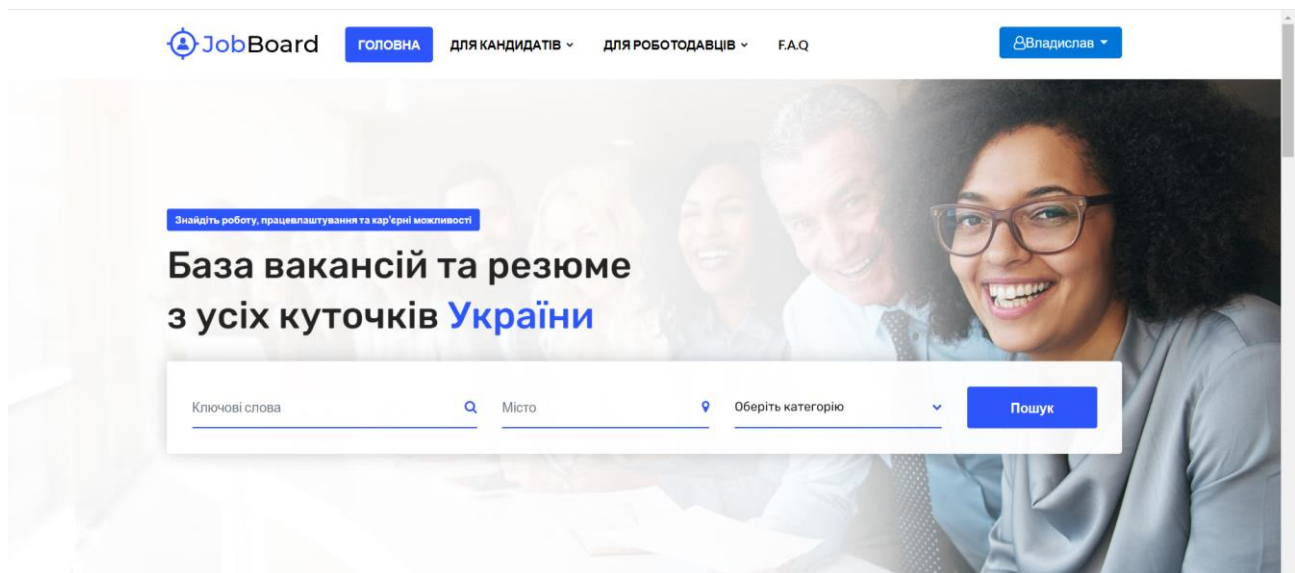


Рисунок 3.4 - Форма пошуку вакансії на головній сторінці

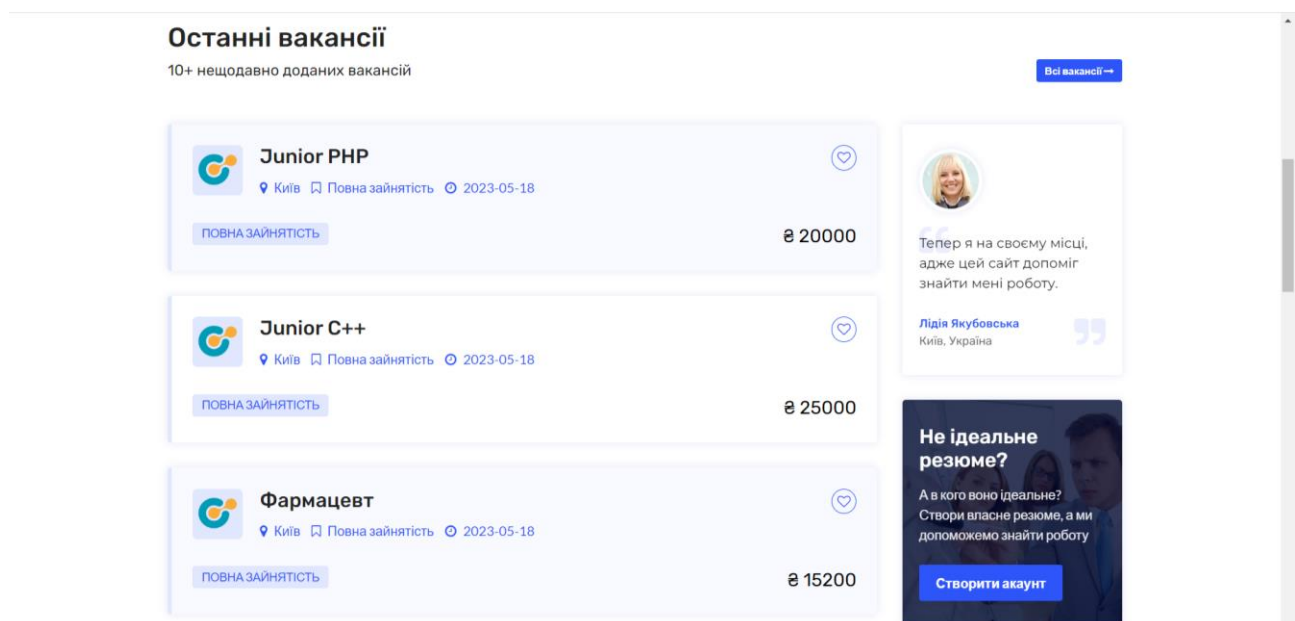


Рисунок 3.5 - Список останніх доданих вакансій на головній сторінці

Окрім вакансій, що знаходяться на головній сторінці, користувач може переглянути більше вакансій за допомогою пошуку з фільтрами. Сторінка пошуку вакансій з фільтрами зображена на рисунку 3.6.

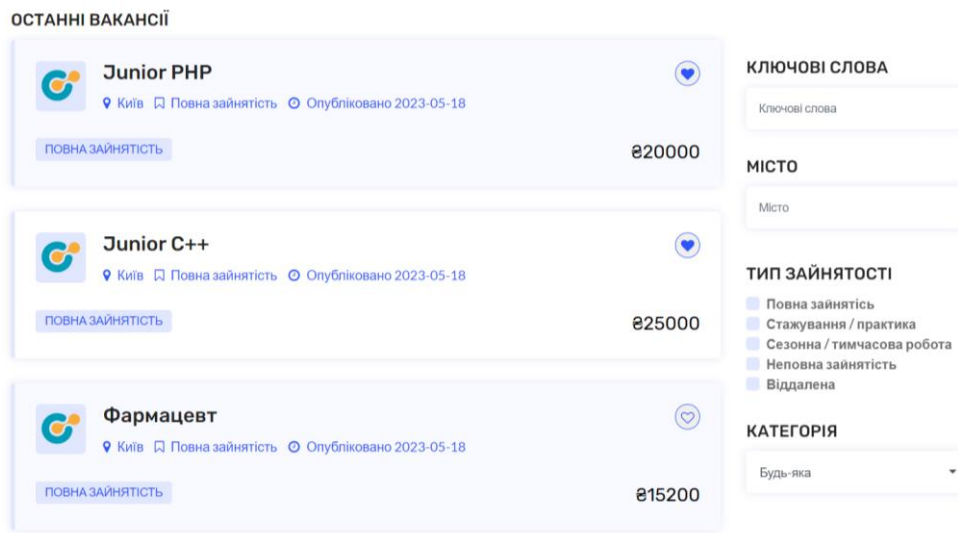


Рисунок 3.6 – Пошук вакансій з фільтрами

При натисканні лівою клавішою миші на вакансію буде здійснено перехід на сторінку з її детальним описом. Сторінка вакансії зображена на рисунку 3.7

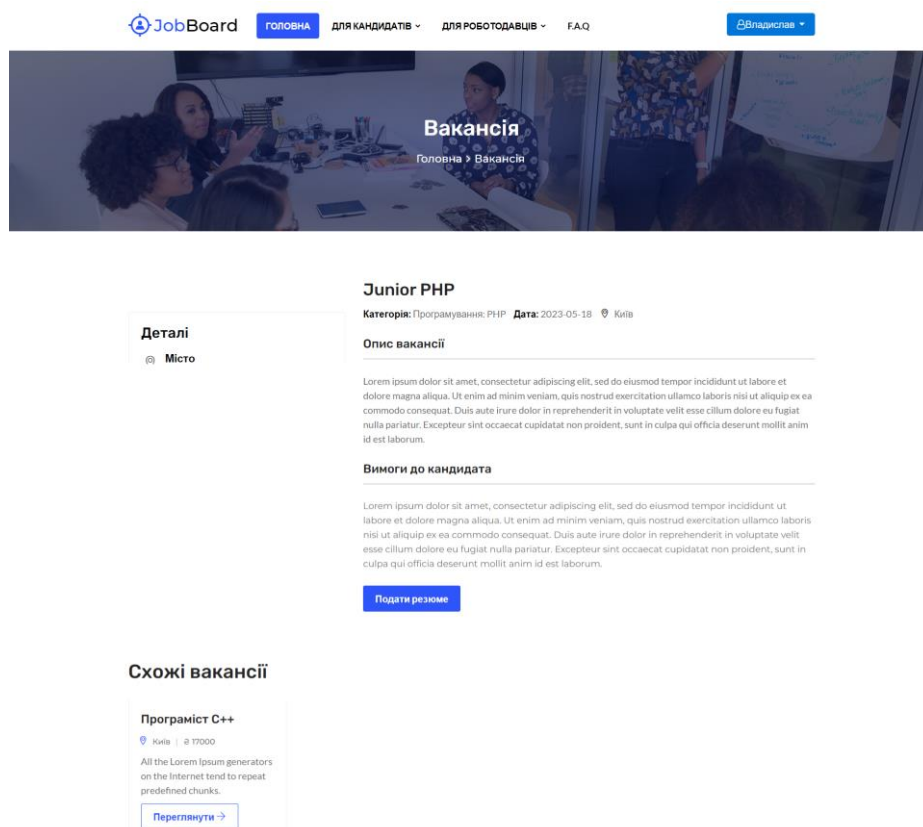


Рисунок 3.7 - Сторінка вакансії з детальною інформацією

Після того як пошукач ознайомився з вакансією, він може подати своє резюме, щоб отримати бажану роботу. Для цього користувач натискає на кнопку «Подати резюме». Буде здійснено перехід на сторінку з формою для подачі резюме. На формі розташовані поле select для вибору резюме із створених користувачем, поле для введення супровідного листа з потужним інтерфейсом для редагування тексту, створення таблиць, посилань тощо. Форма подачі резюме зображена на рисунку 3.8.

The screenshot shows the 'Подача заявки' (Application Submission) form on the JobBoard website. The form includes a header with navigation links: 'ГОЛОВНА', 'ДЛЯ КАНДИДАТІВ', 'ДЛЯ РОБОТОДАВЦІВ', and 'F.A.Q'. The user is logged in as 'Владислав'. The form itself has a title 'Подача заявки' and a text area for the resume. Below this is a section for the cover letter, which includes a rich text editor with various formatting options. At the bottom, there is a file upload section for the resume file, with a button 'Подати' (Submit).

Рисунок 3.8 - Форма подачі резюме

Якщо ж пошукач не може знайти собі підходящу вакансію – він може скористуватись пошуком вакансій по всьому українському сегменті. Для цього необхідно в панелі навігації перейти «Для кандидатів» - «Пошук по українським ресурсам». Буде здійснено перехід на сторінку з формою з двома полями: для введення назви посади та міста. Після введення даних та натискання на кнопку «Пошук» перед користувачем з'являться інші вакансії, яких немає на цьому сайті.

The screenshot shows the JobBoard website interface. At the top, there is a navigation bar with links: 'ГОЛОВНА', 'ДЛЯ КАНДИДАТІВ', 'ДЛЯ РОБОТОДАВЦІВ', 'F.A.Q', 'Реєстрація', and 'Вхід'. Below the navigation bar, there is a search form with two input fields: 'Посада:' (Job Position) and 'Місто:' (City). The 'Посада:' field contains 'Програміст PHP' and the 'Місто:' field contains 'Житомир'. A blue button labeled 'Искать' (Search) is to the right of the 'Місто:' field. Below the search form, there is a list of job results. The first result is 'DevOps Engineer' from 'dou.ua'. The second result is 'JS WebGL Developer Online' from 'notluffjobs.com'. The third result is 'Strong Junior Full-Stack Developer' from 'dou.ua'. The fourth result is 'Full-Stack Engineer (React+Node.js) (ID:580)' from 'dou.ua'. The fifth result is 'Senior Front-end Developer' from 'dou.ua'. The sixth result is 'Cloud Support Engineer' from 'dou.ua'. The seventh result is 'Senior Full-Stack Engineer (Python, Vue)' from 'dou.ua'.

Рисунок 3.9 - Пошук вакансій в українському сегменті Інтернету

Щоб створити резюме, пошукач натискає на своє ім'я в панелі навігації і обирає пункт меню «Створити резюме». Буде здійснено перехід на сторінку з формою для створення резюме. Користувач вводить свої якості, як кандидата в спеціальні поля (освіта, попередні місця роботи) та натискає на кнопку «Створити». Рисунок форми наведено в додатку.

Після того, як пошукач створив своє резюме, він може переглянути його. Для цього потрібно пройти шлях: кнопка з ім'ям користувача – «Мої резюме» - клацнути по резюме користувача. Варто зазначити, що під фотографією пошукача в резюме будуть розташовані кнопки для редагування та видалення резюме. Вони відображаються лише для власника резюме. Сторінка-резюме користувача зображена на рисунку 3.10.

 Олексій Валерійович Редагувати Видалити	Повне ім'я	Олексій Валерійович
	Email	vl.mykoliuk2001@gmail.com
	Телефон	+38(232)132-13-21
	Місто	Київ
Попереднє місце роботи		Php Programmer Lifecell (2022-05 - 2022-12)
Про себе		Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Рисунок 3.10 - Сторінка-резюме пошукача

Розглянемо тепер інтерфейс для користувача з правами «Роботодавець».

Роботодавець не може створювати резюме і подавати заявки на вакансію. Роботодавець зацікавлений в тому, щоб знайти людину, що відповідає вимогам вакансії. Для цього було розроблено базу даних з резюме пошукачів, зареєстрованих на сайті. Роботодавець може сам знайти людину, або ж створити вакансію і чекати заявок.

Щоб перейти до пошуку резюме згідно критеріїв на панелі меню обираємо «Для роботодавців» - «База резюме». Буде здійснено перехід на сторінку з резюме пошукачів, а також формою фільтрації. Сторінку «База резюме» зображено на рисунку 3.11.



Фахівець з продажів

Олена Володимирівна

Вінниця
Неповна зайнятість
Опубліковано 2023-05-12

€15000



C++ Programmer

Валерій Генадійович

Вінниця
Повна зайнятість
Опубліковано 2023-05-12

€25000



PHP Programmer

Олексій Валерійович

Київ
Повна зайнятість
Опубліковано 2023-05-12

€14000



Перекладач з китайської

Ганна Валімівна

Бер
Повна зайнятість
Опубліковано 2023-05-12

КЛЮЧОВІ СЛОВА

Продавець, юрист, фотограф

МІСТО

Місто

ТИП ЗАЙНЯТОСТІ

Повна зайнятість

Стажування / практика

Сезонна / тимчасова робота

Неповна зайнятість

Віддалена

Проектна робота

Позмінна робота

ОСВІТА

Вища

Незакінчена вища

Середньо-спеціальна

Середня

Рисунок 3.11 - «База резюме»

Якщо ж роботодавець не бажає шукати людину самотійно, він може створити вакансію та чекати заявок від пошукачів. Для цього необхідно в панелі-меню клацнути по кнопці з ім'ям (роботодавця) і обрати пункт меню «Створити вакансію». Після цього буде здійснено перехід на сторінку з формою для створення нової вакансії на сайті. Форма для створення вакансії зображена на рисунку 3.12.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		60

Заголовок для резюме

Web Designer

Тип зайнятості

Повна зайнятість

Категорія

Програміст \ Юрист \ Перукар

Місто

Місто

Заробітна плата

14000

Досвід роботи

1

Опис вакансії

Регіонарн

В І @ := = =

Вимоги

Регіонарн

В І @ := = =

Створити вакансію

Рисунок 3.12 - Форма для створення вакансії

Після того, як роботодавець створить вакансію всім пошукачам з резюме, що підходять під дану вакансію буде відправлено електронне повідомлення, про те, що з'явилась вакансія, що може їх зацікавити. А для роботодавця на електронну пошту будуть приходити повідомлення про заявки пошукачів.

Розглянемо інтерфейс для адміністратора.

Після авторизації для адміністратора здійснюється перехід головну сторінку адмін-панелі. Головна сторінка адмін панелі зображена на рисунку 3.13.

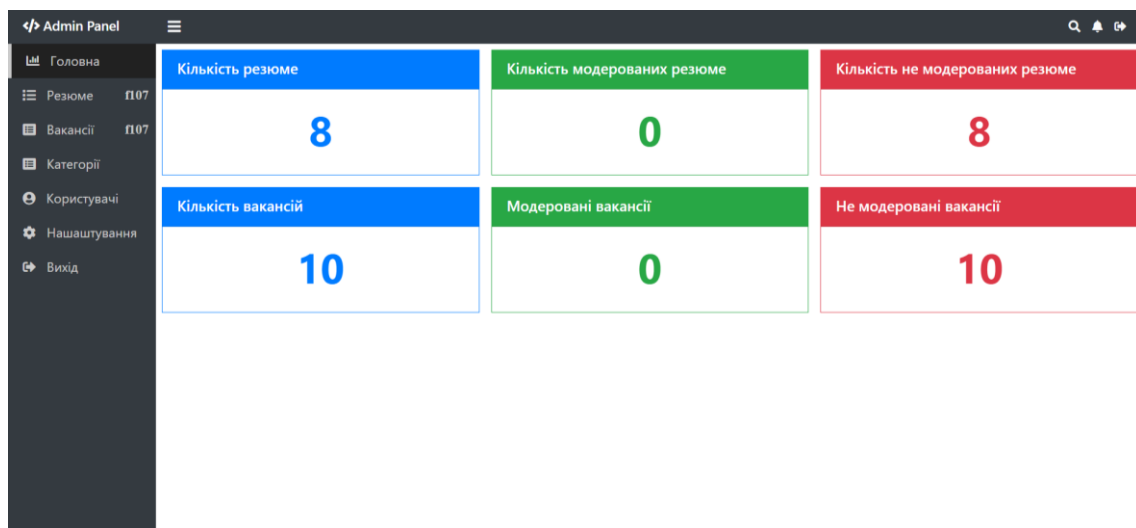


Рисунок 3.13 - Головна сторінка адмін-панелі

Обов'язками адміністратора є модерація вакансій та резюме. Адміністратор перевіряє чи заповнені вони вірно і відповідають вимогам. Для того щоб переглянути резюме в лівій панелі-меню потрібно клацнути по випадаючому меню «Резюме» та обрати «До модерації». На цій сторінці розташована таблиця з не модерованими резюме. Адміністратор має можливість переглянути резюме, а потім підтвердити його або відхилити. Сторінка Резюме до модерації зображена на рисунку 3.14.

Admin Panel		Show 10 entries	Search:			
Головна	Резюме 107	Вакансії 107	Категорії	Користувачі	Нашаштування	Вихід
Користувач	Заголовок	Підкатегорія	Дата створення	Статус	Дії	
[ID: 1]vl.mykoliuk2001@gmail.com	Фахівець з продажів	Продажі (клієнт-менеджмент)	2023-05-12 16:07:52	Не модеровано	Переглянути Підтвердити Відхилити	
[ID: 1]vl.mykoliuk2001@gmail.com	C++ Programmer	Програмування: C / C++	2023-05-12 16:10:36	Не модеровано	Переглянути Підтвердити Відхилити	
[ID: 1]vl.mykoliuk2001@gmail.com	PHP Programmer	Програмування: PHP	2023-05-12 16:11:27	Не модеровано	Переглянути Підтвердити Відхилити	
[ID: 1]vl.mykoliuk2001@gmail.com	Продавець магазину	Продавець-консультант Керівник магазину	2023-05-12 16:14:01	Не модеровано	Переглянути Підтвердити Відхилити	
[ID: 1]vl.mykoliuk2001@gmail.com	Менеджер по роботі з персоналом	Менеджер по роботі з персоналом	2023-05-12 16:14:38	Не модеровано	Переглянути Підтвердити Відхилити	
[ID: 3]mikulukn@gmail.com	Перекладач з китайської	Перекладачі	2023-05-12 16:12:44	Не модеровано	Переглянути Підтвердити Відхилити	
[ID: 3]mikulukn@gmail.com	Перекладач з німецької	Перекладач-асистент	2023-05-12 16:13:19	Не модеровано	Переглянути Підтвердити Відхилити	

Рисунок 3.14 - Резюме до модерації

Для того щоб переглянути вакансії в лівій панелі-меню потрібно клацнути по випадяючому меню «Вакансії» та обрати «До модерації». На цій сторінці розташована таблиця з не модерованими вакансіями. Адміністратор має можливість переглянути вакансію, а потім підтвердити її або відхилити. Сторінка Модерація вакансій зображена на рисунку 3.14.

Admin Panel		Модерація вакансій				
Головна		Show 10 entries		Search:		
Резюме	П07	Користувач	Заголовок	Підкатегорія	Дата створення	Статус
Вакансії	П07					
Категорії						
Користувачі						
Нашаштування						
Вихід						
		[ID: 1]vl.mykoliuk2001@gmail.com	Web Designer	Дизайн (Web, UI, Flash, 3D)	2023-03-28 09:57:33	Не модеровано
		[ID: 2]suzuki2001@ukr.net	Продавець-консультант	Продавець-консультант Керівник магазину	2023-05-18 19:11:44	Не модеровано
		[ID: 2]suzuki2001@ukr.net	Фінансовий консультант	Фінансовий консультант / Продаж фінансових продуктів	2023-05-18 19:13:34	Не модеровано
		[ID: 2]suzuki2001@ukr.net	Перукар	Косметологія / SPA / Перукарі	2023-05-18 19:14:00	Не модеровано
		[ID: 2]suzuki2001@ukr.net	Психолог	Психолог	2023-05-18 19:14:36	Не модеровано
		[ID: 2]suzuki2001@ukr.net	Інспектор відділу кадрів	Інспектор відділу кадрів	2023-05-18 19:15:11	Не модеровано

Рисунок 3.15 - Модерація вакансій

Для того щоб переглянути список користувачів – в лівій панелі-меню потрібно обрати пункт «Користувачі». Буде здійснено перехід на таблицю з користувачами. Сторінка користувачі зображена на рисунку 3.15.

Admin Panel		Користувачі			
Головна		Show 10 entries		Search:	
Резюме	П07	Користувач	ПІБ	Права	Дата реєстрації
Вакансії	П07				
Категорії					
Користувачі					
Нашаштування					
Вихід					
		[ID: 1]vl.mykoliuk2001@gmail.com	Миколюк Владислав	applicant	2023-02-25 22:19:37
		[ID: 2]suzuki2001@ukr.net	Фільваркова Ольга	employer	2023-05-16 20:54:57
		[ID: 3]mikulukn@gmail.com	Миколюк Назарій	applicant	2023-06-01 14:03:08
		Користувач	Заголовок	Підкатегорія	Дата створення
		Showing 1 to 3 of 3 entries			
				Previous	1 Next

Рисунок 3.16 - Сторінка «Користувачі»

3.3. Тестування роботи веб-сайту

Тестування програмного забезпечення (англ. Software Testing) — це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки.

Тестування програмного забезпечення - процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином [3].

Може оцінюватись:

- відповідність вимогам, якими керувалися проектувальники та розробники
- правильна відповідь для усіх можливих вхідних даних
- виконання функцій за прийнятний час
- практичність
- сумісність з програмним забезпеченням та операційними системами
- відповідність задачам замовника.

Після завершення розроблення веб-застосунку було проведено тестування на відповідність вимогам, поставленим у даній роботі. Тестування функціональних можливостей та роботи елементів веб-сайту проводилося за допомогою ручного тестування. Після даного тестування було виявлено деякі помилки та виправлено.

Для тестування веб-сайту пошуку роботи написаному на Laravel Framework було використано Laravel Debugger.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		64

«Laravel Debugbar» має 11 вкладинок:

- messages – повідомлення про помилки;
- Timeline – час завантаження сторінки та на які процеси він був витрачений;
- Views – які php файли з HTML розміткою були завантажені;
- Route – інформація про задіяний контролер;
- Queries – інформація про SQL запити, які були виконані для інтерпретації даної сторінки;
- Models – інформація про таблиці баз даних, з яких бралася вибірка;
- Mails;
- Gate;
- Session – інформація про дані, які занесені в сесію
- Request – інформація про дані, які передаються методами POST або GET

«Laravel Debugbar» також показує яка версія PHP використовується на сервері та скільки оперативної пам'яті знадобилось для інтерпретації веб-сторінки

Висновки до розділу 3

В цьому розділі була розроблена архітектура додатку. Завдяки проектуванню діаграм компонентів та розгортання було визначено основні об'єкти моделей в системі.

Було описано інтерфейс додатку. На основі опису інтерфейсу зрозумілі можливості користувача, а саме: для користувача-шукача можливість увійти в систему, створити резюме, отримати список рекомендованих вакансій та подати заявку на вакансію. Для користувача-роботодавця – створити вакансію, отримати доступ до бази резюме, а також відслідковування заявок. Для

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

модератора – це інструменти для модерації резюме та вакансій. А також доступ до даних користувачів.

Було проведено тестування додатку за допомогою спеціального програмного забезпечення. Виправлено незначні помилки. Веб-сайт пошуку роботи працює коректно та виконує поставлені задачі.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В результаті виконання випускної роботи бакалавра було спроектовано та реалізовано веб-сайт пошуку роботи. Під час розробки цього веб-проєкту було реалізовано наступні функції:

- поставлене завдання, щодо реалізації проєкту. Проаналізовані існуючі аналоги. Продумано можливий функціонал;
- був визначений аналіз основних вимог до веб-сайту. В нього ввійшли: вимоги користувачів, функціональні та не функціональні вимоги;
- за допомогою діаграми варіантів використання було відображено сценарії використання системи та користувачів системи, які використовують її функції;
- описано особливості фізичного представлення системи за допомогою діаграми компонентів;
- визначено архітектуру системи, що розробляється, встановлено залежності між програмними компонентами;
- проведено розробку загального і функціонального алгоритмів веб-додатку. Також наведено частково код і пояснено процес виконання роботи. Проведено тестування.

Проєкт має багато можливостей для подальшого вдосконалення, зокрема:

- оновлення інтерфейсу;
- інтеграція із сервісами штучного інтелекту для аналізу резюме та вакансій;
- авторизація через соціальні мережі;
- створення стрічки новин;

Розробка виконана у повному обсязі та відповідає вимогам, що зазначені в поставленій задачі. Завдяки обраній архітектурі подальша підтримка та вдосконалення продукту відбуватиметься без ускладнень.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MVC [Електронний ресурс] – Режим доступу до ресурсу: <https://brander.ua/technologies/mvc>.
2. Що таке нефункціональні вимоги: приклади, визначення, повне керівництво [Електронний ресурс] – Режим доступу до ресурсу: <https://visuresolutions.com/uk/blog/non-functional-requirements/>.
3. Що таке тестування програмного забезпечення? [Електронний ресурс] – Режим доступу до ресурсу: <https://qalight.ua/baza-znaniy/shho-take-testuvannya-programnogo-zabezpechennya/>.
4. Laravel Debugger [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel-news.com/laravel-debugbar>.
5. Laravel 5.7 Controllers [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/5.7/controllers>.
6. Eloquent: Getting Started [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/5.7/eloquent>.
7. Розуміння основ роботи API і REST API – короткий вступ [Електронний ресурс] – Режим доступу до ресурсу: <https://sebweo.com/rozuminnya-osnov-roboti-api-i-rest-api-korotkij-vstup/>.
8. Рейтинг сайтів з роботою в Україні [Електронний ресурс] – Режим доступу до ресурсу: <https://www.unian.net/economics/other/rejting-saytov-s-rabotoy-v-ukraine-12253728.html>.
9. Work.ua відгуки [Електронний ресурс] – Режим доступу до ресурсу: <https://www.otzyvua.net/uk/workua.html>.
10. Для чого потрібне веб-програмування? [Електронний ресурс] – Режим доступу до ресурсу: <https://indevlab.com/uk/blog-ua/dlya-chogo-potribne-veb-programuvannya/>.
11. Повне розуміння діаграми компонентів UML за допомогою легкого методу [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mindonmap.com/uk/blog/uml-component-diagram/>.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		69

12. Використання уніфікованої мови моделювання [Електронний ресурс] – Режим доступу до ресурсу: http://iwanoff.inf.ua/oop_kn/LabTraining05.html.
13. Діаграма послідовності [Електронний ресурс] – Режим доступу до ресурсу: <https://www.maxzosim.com/sequence-diagrams/>.
14. РНР [Електронний ресурс] – Режим доступу до ресурсу: <http://programming.in.ua/web-design/allphp/30-about-php.html>.
15. 17 переваг фреймворку Laravel [Електронний ресурс] – Режим доступу до ресурсу: <https://icstudio.online/post/17-perevag-frejmworku-laravel>.
16. Що таке Python і де він використовується [Електронний ресурс] – Режим доступу до ресурсу: <https://dan-it.com.ua/uk/blog/python-hto-jeto-za-jazyk-programmirovanija-i-gde-ego-ispolzujut/>.
17. Мова програмування Ruby: чи варто вивчати, плюси і мінуси [Електронний ресурс] – Режим доступу до ресурсу: <http://teg.com.ua/mova-programyvannia-ruby-chi-var-to-vivchati-plusi-i-minysi/3/>.

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

Код ResumeController.php

```

<?php

namespace App\Http\Controllers\Resume;

use App\Models\Category;
use App\Models\City;
use App\Models\Resume;
use App\Models\Subcategory;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;
class ResumeController extends Controller
{
    public $category;
    public $city;
    public $subcategory;

    public $resume;

    public function __construct(Category $category, City $city, Subcategory
$subcategory, Resume $resume)
    {
        $this->category = $category;
        $this->city = $city;
        $this->subcategory = $subcategory;
        $this->resume = $resume;
    }

    public function index(Request $request)
    {
        $data = [
            'categories' => $this->category->all(),
            'cities' => $this->city->all()
        ];

        $resumes = $this->resume->newQuery();

        if ($request->has('type_employment')) {
            foreach ($request->input('type_employment') as $type_employment) {
                $resumes = $resumes->orWhere('job_type', $type_employment);
            }
        }

        if ($request->has('education')) {
            foreach ($request->input('education') as $education) {
                $resumes = $resumes->orWhere('education', $education);
            }
        }

        if ($request->has('city')) {
            $city_id = $this->city->where('name', $request->input('city'))->first();
            if ($city_id) {
                $city_id = $city_id->id;
            }
        }
    }
}

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

        $resumes = $resumes->where('city_id', $city_id);
    }

    $resumes = $resumes->get();

    if ($request->has('key_words')) {
        $key_words = $request->input('key_words');

        $results = [];

        foreach ($resumes as $resume) {
            similar_text($key_words, $resume->title, $percent);

            if ($percent > 50) {
                $results[] = $resume;
            }
        }

        if (count($results) > 0) {
            $data['resumes'] = $results;
        } else {
            $data['resumes'] = $resumes;
        }

        $data['request'] = $request;

        return view('resumes.index', $data);
    }

    public function create()
    {
        $data =
        [
            'cities' => $this->city->all(),
            'subcategories' => $this->subcategory->all()
        ];

        return view('resumes.create', $data);
    }

    public function show($id)
    {
        $crud_approval = false;

        if(Auth::check() and Auth::user()->id == $this->resume->where('id',
        $id)->first()->user->id)
            $crud_approval = true;

        $data =
        [
            'resume' => $this->resume->where('id', $id)->first(),

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'crud_approval' => $crud_approval
    ];
    return view('resumes.show', $data);
}

public function store(Request $request)
{
    $data = $request->all();
    unset($data['_token']);

    if($this->city->where('name', 'like', $request->input('city'))->first())
        $data['city_id'] = $this->city->where('name', 'like', $request->input('city').'%')->first()->id;
    else
        return redirect()->back()->withErrors(['msg' => "Оберіть місто із запропонованого списку"]);

    unset($data['city']);

    if($this->subcategory->where('name', 'like', $request->input('subcategory').'%')->first())
        $data['subcategory_id'] = $this->subcategory->where('name', 'like', $request->input('subcategory').'%')->first()->id;
    else
        return redirect()->back()->withErrors(['msg' => "Оберіть категорію із запропонованого списку"]);

    unset($data['subcategory']);

    $data['photo'] = $this->storePhoto($request, Resume::count()+1);

    $data['user_id'] = Auth::user()->id;
    Resume::create($data);

    return redirect()->back();
}

public function storePhoto(Request $request, $id)
{
    $filename = null;

    if ($request->hasFile('photo')) {
        $file = $request->file('photo');
        $filename = time() . '_' . $file->getClientOriginalName();
        $file->storeAs('/public/resume-photos/'.$id, $filename);
        $filename = "resume-photos/".$id."/".$filename;
    }

    return $filename;
}

public function my_resumes()
{
    $data =
    [
        'resumes' => $this->resume->where('user_id', Auth::user()->id)->get()
    ];
    return view('resumes.my_resumes', $data);
}

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

    }

    public function delete($id)
    {
        if(Auth::user()->id == $this->resume->where('id', $id)->first()->user-
>id)
        {
            $resume = $this->resume->where('id', $id)->first();
            $message = [
                'message' => "Резюме ".$resume->title." було успішно видалено"
            ];
            $resume->delete();

            return redirect()->route('resume.my_resumes')->withErrors($message);
        }else{
            return redirect()->back();
        }
    }

    public function edit($id)
    {
        if($this->resume->where('id', $id)->exists() && Auth::user()->id ==
$this->resume->where('id', $id)->first()->user->id)
        {

            $data =
            [
                'resume' => $this->resume->where('id', $id)->first(),
                'crud_approval' => true,
                'subcategories' => $this->subcategory->all(),
                'cities' => $this->city->all()
            ];
            return view('resumes.edit', $data);

        }else{
            return redirect()->back();
        }

    }

}

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

Код JobsController.php

```

<?php

namespace App\Http\Controllers\Jobs;
use Symfony\Component\DomCrawler\Crawler;
use Goutte\Client;
use App\Models\City;
use App\Models\Job;
use App\Models\Region;
use App\Models\Subcategory;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use App\Http\Controllers\Parser\ParserController;

class JobsController extends Controller
{
    public $region;

    public function __construct(Region $region, Subcategory $subcategory, City $city, Job $job)
    {
        $this->region = $region;
        $this->subcategory = $subcategory;
        $this->city = $city;
        $this->job = $job;
    }

    public function index()
    {
        $parserController = new ParserController();

        /*
            $data =
            [
                'jobs' => $parserController->parseWorkUa()
            ];*/

        $data =
        [
            'jobs' => $this->job->orderBy('created_at', 'DESC')-
>paginate(10)
        ];
        return view('jobs.index', $data);
    }
    public function byRegion()
    {
        $data =
        [
            'regions' => $this->region->all()
        ];
        return view('jobs.by-region', $data);
    }

    public function create()
    {

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

        $data =
        [
            'subcategories' => $this->subcategory->all(),
            'cities' => $this->city->all()
        ];
        return view('jobs.create', $data);
    }

    public function store(Request $request)
    {
        $data = $request->all();
        unset($data['_token']);

        if($this->city->where('name', 'like', $request->input('city'))->first())
            $data['city_id'] = $this->city->where('name', 'like', $request->input('city').'%')->first()->id;
        else
            return redirect()->back()->withErrors(['msg' => "Оберіть місто із запропонованого списку"]);

        unset($data['city']);

        if($this->subcategory->where('name', 'like', $request->input('subcategory').'%')->first())
            $data['subcategory_id'] = $this->subcategory->where('name', 'like', $request->input('subcategory').'%')->first()->id;
        else
            return redirect()->back()->withErrors(['msg' => "Оберіть категорію із запропонованого списку"]);

        unset($data['subcategory']);

        $data['user_id'] = Auth::user()->id;

        Job::create($data);

        return redirect()->back();
    }

    public function storeCompanyLogo(Request $request, $id)
    {
        $filename = null;

        if ($request->hasFile('photo')) {
            $file = $request->file('photo');
            $filename = time() . '_' . $file->getClientOriginalName();
            $file->storeAs('/public/company-logos/'.$id, $filename);
            $filename = "company-logos/".$id."/".$filename;
        }

        return $filename;
    }

    public function show($id)
    {
        $data =
        [
            'job' => $this->job->where('id', $id)->first()
        ];
        return view('jobs.show', $data);
    }

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		77

```
}

public function showApplyForm(Request $request)
{
    $data = [
        'resumes' => Auth::user()->resumes->all()
    ];

    return view('jobs.apply', $data);
}
}
```

Код WishListController.php

```

<?php

namespace App\Http\Controllers\WishList;

use App\Models\Job;
use App\Models\WishList;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;

class WishListController extends Controller
{
    public $wishList;
    public $job;
    public function __construct(WishList $wishList, Job $job)
    {
        $this->wishList = $wishList;
        $this->job = $job;
    }

    public function store(Request $request)
    {
        $new_wishlist = $this->wishList;

        WishList::create(
            [
                'user_id' => Auth::user()->id,
                'job_id' => $request->input('job_id')
            ]
        );

        return redirect()->back()->withInput()->with('scrollTo', 'job-'.
        $request->input('job_id'));
    }

    public function delete(Request $request)
    {
        $old_wishlist = $this->wishList->where('id', $request-
        >input('wishlist_id'))->first();
        $old_wishlist->delete();

        return redirect()->back();
    }

    public function index()
    {
        $current_user_id = Auth::user()->id;

        $data =
            [
                'wishlists' => $this->wishList->where('user_id',
        $current_user_id)->orderBy('created_at', 'DESC')->get()
            ];
    }
}

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

/*      $data = [
        'wishlists' => $this->job->all()
    ];*/

    return view('wishlist.index', $data);

}

}

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

Код RecommendationController.php

```

<?php

namespace App\Http\Controllers\Recommendation;

use App\Models\Job;
use App\Models\Resume;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;

class RecommendationController extends Controller
{
    public $job;
    public function __construct(Job $job)
    {
        $this->job = $job;
    }

    public function getRecommendationsByResumes($user_id)
    {
        $userResumes = Resume::where('user_id', $user_id)->get(); // Дістаємо
резюме

        $recommendations = $this->job;

        // Збираємо підкатегорії, які цікавлять користувача
        $subcategory_ids = [];

        foreach($userResumes as $resume)
        {
            $subcategory_ids[] = $resume->subcategory_id;
        }

        $recommendations = $recommendations->whereIn('subcategory_id',
$subcategory_ids);

        // Відсіємомо категорії згідно міст
        $city_ids = [];

        foreach($userResumes as $resume)
        {
            $city_ids[] = $resume->city_id;
        }

        $recommendations = $recommendations->whereIn('city_id', $city_ids);

        // Додавання фільтрації за зарплатою
        $recommendations = $recommendations->where(function ($query) use
($userResumes) {
            foreach ($userResumes as $resume) {
                $minSalary = $resume->salary * 0.8;
                $query->orWhere('salary', '>=', $minSalary);
            }
        });

        // Додавання фільтрації за досвідом роботи

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $recommendations = $recommendations->where(function ($query) use
($userResumes) {
            foreach ($userResumes as $resume) {
                $totalExperience = $this->calculateTotalExperience($resume);
                $query->orWhere('experience', '>=', $totalExperience);
            }
        });

        $recommendations = $recommendations->get();
        return $recommendations;
    }

    private function calculateTotalExperience($resume)
    {
        $totalExperience = 0;

        for ($i = 1; $i <= 3; $i++) {
            $startKey = "work_start{$i}";
            $finishKey = "work_finish{$i}";

            if (isset($resume->$startKey) && isset($resume->$finishKey)) {
                $startDate = strtotime($resume->$startKey);
                $finishDate = strtotime($resume->$finishKey);
                $experience = ($finishDate - $startDate) / (365 * 24 * 60 * 60);
                // Вираховуємо рік досвіду

                if ($experience < 1) {
                    $experience = 1; // Округлюємо до 1 року, якщо менше
                }

                $totalExperience += $experience;
            }
        }

        return $totalExperience;
    }

    public function index()
    {
        $recommendations = $this->getRecommendationsByResumes(Auth::user()->id);
        dd($recommendations);
    }
}

```

Код resumes/index.blade.php

```

@extends('layouts.app')

@section('content')
    <!-- header END -->
    <!-- Content -->
    <div class="page-content bg-white">
        <!-- inner page banner -->
        <div class="dez-bnr-inr overlay-black-middle" style="background-
image:url('{{"images/banner/bnr1.jpg"}}');">
            <div class="container">
                <div class="dez-bnr-inr-entry">
                    <h1 class="text-white">База Резюме</h1>
                    <!-- Breadcrumb row -->
                    <div class="breadcrumb-row">
                        <ul class="list-inline">
                            <li><a href="{{ route('welcome')
}}">Головна</a></li>
                            <li>База Резюме</li>
                        </ul>
                    </div>
                    <!-- Breadcrumb row END -->
                </div>
            </div>
        <!-- inner page banner END -->
        <div class="content-block">
            <div class="section-full bg-white browse-job content-inner-2">
                <div class="container">
                    <div class="row">
                        <div class="col-xl-9 col-lg-8">

                            @if(isset($resumes))
                                <ul class="post-job-bx">

                                    @foreach($resumes as $resume)
                                        <li>
                                            <a href="{{ route('resume.show', ['id' =>
$resume->id]) }}">
                                                <div class="d-flex m-b30">
                                                    <div class="job-post-company">
                                                        <span>photo) }}" /></span>
                                                    </div>
                                                    <div class="job-post-info">
                                                        <h4>{{ $resume->title }}</h4>
                                                        <ul>
                                                            <li style = "color:black">{{
$resume->name }}</li>
                                                            <li><i class="fa fa-map-
marker"></i> {{ $resume->city->name }}</li>
                                                            <li><i class="fa fa-
usd"></i> {{ $resume->job_type }}</li>
                                                            <li><i class="fa fa-clock-
o"></i> Опубліковано {{ \Carbon\Carbon::createFromFormat('Y-m-d H:i:s', $resume-
>created_at)->format('Y-m-d') }}</li>
                                                        </ul>

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

        </div>
    </div>
    <div class="d-flex">

        <div class="salary-bx">
            <span>@{{ $resume->
salary}}</span>

        </div>
        <form action = "{{
route('addToFavorites') }}" method = "POST">
            @csrf
            <input type = "hidden" name =
"job_id" value = "{{ $resume->id }}">
            <button class="post-like fa fa-
heart-o" type = "submit"></button>

        </form>

    </div>

</a>

</li>
@endforeach
</ul>
@else
    <h1>Немає резюме. Перевірте ваші параметри</h1>
@endif

</div>
<div class="col-xl-3 col-lg-4">
    <div class="sticky-top">
        <form action = "{{ route('resume.index') }}"
method = "GET">

            <div class="clearfix m-b30">
                <h5 class="widget-title font-weight-700
text-uppercase">Ключові слова</h5>
                <div class="">
                    <input type="text" class="form-control"
placeholder="Продавець, юрист, фотограф" name = "key_words" value="{{ $request->input('key_words') }}">
                </div>
            </div>

            <div class="clearfix m-b10">
                <h5 class="widget-title font-weight-700 m-t0
text-uppercase">Місто</h5>
                <input type="text" class="form-control m-
b30" placeholder="Місто" list="cities" name = "city" id="city" value="{{
$request->input('city') }}">
                <datalist id = "cities">
                    @foreach($cities as $city)
                        <option value="{{ $city->name
}}"></option>
                    @endforeach
                </datalist>
            </div>

            <!-- Тип зайнятості -->
            <div class="clearfix m-b30">

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

<h5 class="widget-title font-weight-700
text-uppercase">Тип зайнятості</h5>
<div class="row">
  <div class="col-lg-12 col-md-12 col-sm-
12 col-12">
    <div class="product-brand">
      <div class="custom-control
custom-checkbox">
        <input type="checkbox"
class="custom-control-input" id="check1" name="type_employment[]" value = "Повна
зайнятість">
        <label class="custom-
control-label" for="check1">Повна зайнятість</label>
      </div>
      <div class="custom-control
custom-checkbox">
        <input type="checkbox"
class="custom-control-input" id="check2" name="type_employment[]" value =
"Стажування / практика">
        <label class="custom-
control-label" for="check2">Стажування / практика</label>
      </div>
      <div class="custom-control
custom-checkbox">
        <input type="checkbox"
class="custom-control-input" id="check3" name="type_employment[]" value =
"Сезонна / тимчасова робота">
        <label class="custom-
control-label" for="check3">Сезонна / тимчасова робота</label>
      </div>
      <div class="custom-control
custom-checkbox">
        <input type="checkbox"
class="custom-control-input" id="check4" name="type_employment[]" value =
"Неповна зайнятість">
        <label class="custom-
control-label" for="check4">Неповна зайнятість</label>
      </div>
    </div>
    <div class="custom-control
custom-checkbox">
        <input type="checkbox"
class="custom-control-input" id="check5" name="type_employment[]" value =
"Віддалена">
        <label class="custom-
control-label" for="check5">Віддалена</label>
      </div>
    <div class="custom-control
custom-checkbox">
        <input type="checkbox"
class="custom-control-input" id="check6" name="type_employment[]" value =
"Проектна робота">
        <label class="custom-
control-label" for="check6">Проектна робота</label>
      </div>
    <div class="custom-control
custom-checkbox">
        <input type="checkbox"
class="custom-control-input" id="check7" name="type_employment[]" value =
"Позмінна робота">

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<label class="custom-
control-label" for="check7">Позмінна робота</label>
</div>

</div>
</div>
</div>
</div>
<!-- Тип зайнятості END-->

<!-- Освіта -->
<div class="clearfix m-b30">
<h5 class="widget-title font-weight-700
text-uppercase">Освіта</h5>

<div class="row">
<div class="col-lg-12 col-md-12 col-sm-
12 col-12">

<div class="product-brand">
<div class="custom-control
custom-checkbox">

<input type="checkbox"
class="custom-control-input" id="check8" name="education[]" value = "Вища">
<label class="custom-
control-label" for="check8">Вища</label>

</div>
<div class="custom-control
custom-checkbox">

<input type="checkbox"
class="custom-control-input" id="check9" name="education[]" value = "Незакінчена
вища">

<label class="custom-
control-label" for="check9">Незакінчена вища</label>
</div>
<div class="custom-control
custom-checkbox">

<input type="checkbox"
class="custom-control-input" id="check10" name="education[]" value = "Середньо-
спеціальна">

<label class="custom-
control-label" for="check10">Середньо-спеціальна</label>
</div>
<div class="custom-control
custom-checkbox">

<input type="checkbox"
class="custom-control-input" id="check11" name="education[]" value = "Середня">
<label class="custom-
control-label" for="check11">Середня</label>

</div>
</div>
</div>
</div>
</div>
<!-- Освіта End -->

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

                                @foreach($categories as $category)
                                <option value = "{{ $category->id
}}">{{ $category->name }}</option>
                                @endforeach
                                </select>
                                </div>-->

                                <div class = "clearfix">
                                <input type = "submit" value = "Пошук" class
= "btn btn-primary">
                                </div>
                                </form>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                <!-- Content END-->
@endsection

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

Код jobs/index.blade.php

```

@extends('layouts.app')

@section('content')
    @if(session('scrollTo'))
        <script>
            window.onload = function() {
                var element = document.getElementById('{{ session('scrollTo')
            }}');

            if (element) {
                element.scrollIntoView();
            }
        }
    </script>
    @endif

    <!-- Content -->
    <div class="page-content bg-white">
        <!-- inner page banner -->
        <div class="dez-bnr-inr overlay-black-middle" style="background-
image:url('{{ "/images/banner/bnr1.jpg" }}');">
            <div class="container">
                <div class="dez-bnr-inr-entry">
                    <h1 class="text-white">Список вакансій</h1>
                    <!-- Breadcrumb row -->
                    <div class="breadcrumb-row">
                        <ul class="list-inline">
                            <li><a href="#">Головна</a></li>
                            <li>Список вакансій</li>
                        </ul>
                    </div>
                    <!-- Breadcrumb row END -->
                </div>
            </div>
        </div>
        <!-- inner page banner END -->
        <!-- contact area -->
        <div class="content-block">
            <!-- Browse Jobs-->
            <div class="section-full bg-white browse-job content-inner-2">
                <div class="container">
                    <div class="row">
                        <div class="col-xl-9 col-lg-8">
                            <h5 class="widget-title font-weight-700 text-
uppercase">Останні вакансії</h5>
                            <ul class="post-job-bx">

                                @foreach($jobs as $job)
                                    <li>
                                        <a href="{{ route('jobs.show', ['id' =>
$job->id]) }}" id = "job-{{$job->id}}">

                                        <div class="d-flex m-b30">
                                            <div class="job-post-company">
                                                <span></span>
</div>
<div class="job-post-info">
  <h4>{{ $job['title'] }}</h4>
  <ul>
    <li><i class="fa fa-map-
marker"></i> {{ $job->city->name }}</li>
    <li><i class="fa fa-
bookmark-o"></i> {{ $job->job_type }}</li>
    <li><i class="fa fa-clock-
o"></i> Опубликовано {{ \Carbon\Carbon::createFromFormat('Y-m-d H:i:s', $job-
>created_at)->format('Y-m-d') }}</li>
  </ul>
</div>
</div>
<div class="d-flex">
  <div class="job-time mr-auto">
    <span>{{ $job->job_type
</div>
  <div class="salary-bx">
    <span>@{{ $job->salary }}</span>
  </div>
</div>

@if(\Illuminate\Support\Facades\Auth::user()->wishlists->contains('job_id',
$job->id))
  <form action = "{{
route('wishlist.delete') }}" method = "POST">
    @csrf
    <input type = "hidden" name
= "wishlist_id" value = "{{ \Illuminate\Support\Facades\Auth::user()->wishlists-
>where('job_id', $job->id)->first()->id }}">
    <button class="post-like fa
fa-heart" type = "submit"></button>
  </form>

@else
  <form action = "{{
route('wishlist.store') }}" method = "POST">
    @csrf
    <input type = "hidden" name
= "job_id" value = "{{ $job->id }}">
    <button class="post-like fa
fa-heart-o" type = "submit"></button>
  </form>
@endif

</div>
</a>
</li>

@endforeach

</ul>
<div class="pagination-bx m-t30">

```

```

        <ul class="pagination">
            <li class="previous"><a href="#"><i
class="ti-arrow-left"></i> Prev</a></li>
            <li class="active"><a href="#">1</a></li>
            <li><a href="#">2</a></li>
            <li><a href="#">3</a></li>
            <li class="next"><a href="#">Next <i
class="ti-arrow-right"></i></a></li>
        </ul>
    </div>
</div>
<div class="col-xl-3 col-lg-4">
    <div class="sticky-top">
        <div class="clearfix m-b30">
            <h5 class="widget-title font-weight-700
text-uppercase">Ключові слова</h5>
            <div class="">
                <input type="text" class="form-control"
placeholder="Ключові слова">
            </div>
        </div>
        <div class="clearfix m-b10">
            <h5 class="widget-title font-weight-700 m-t0
text-uppercase">Місто</h5>
            <input type="text" class="form-control m-
b30" placeholder="Місто">
        </div>
        <div class="clearfix m-b30">
            <h5 class="widget-title font-weight-700
text-uppercase">Тип зайнятості</h5>
            <div class="row">
                <div class="col-lg-12 col-md-12 col-sm-
12 col-12">
                    <div class="product-brand">
                        <div class="custom-control
custom-checkbox">
                            <input type="checkbox"
class="custom-control-input" id="check1" name="example1">
                            <label class="custom-
control-label" for="check1">Повна зайнятість</label>
                        </div>
                        <div class="custom-control
custom-checkbox">
                            <input type="checkbox"
class="custom-control-input" id="check2" name="example1">
                            <label class="custom-
control-label" for="check2">Стажування / практика</label>
                        </div>
                        <div class="custom-control
custom-checkbox">
                            <input type="checkbox"
class="custom-control-input" id="check3" name="example1">
                            <label class="custom-
control-label" for="check3">Сезонна / тимчасова робота</label>
                        </div>
                        <div class="custom-control
custom-checkbox">
                            <input type="checkbox"
class="custom-control-input" id="check4" name="example1">
                            <label class="custom-
control-label" for="check4">Неповна зайнятість</label>
                        </div>

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

Код layouts/app.blade.php

```

@include('layouts.components.header')
@include('layouts.components.footer')

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

<head>
    <meta charset="utf-8">

    <!-- FAVICONS ICON -->
    <link rel="icon" href="images/favicon.ico" type="image/x-icon"/>
    <link rel="shortcut icon" type="image/x-icon" href="images/favicon.png"/>

    <!-- PAGE TITLE HERE -->
    <title>JobBoard - HTML Template</title>

    <!-- MOBILE SPECIFIC -->
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!--[if lt IE 9]>
    <script src="{{asset('js/html5shiv.min.js')}}"></script>
    <script src="{{asset('js/respond.min.js')}}"></script>
    <![endif]-->

    <!-- STYLESHEETS -->
    <link rel="stylesheet" type="text/css" href="{{asset('/css/plugins.css')}}">
    <link rel="stylesheet" type="text/css" href="{{asset('/css/style.css')}}">
    <link rel="stylesheet" type="text/css"
href="{{asset('/css/template.css')}}">
    <link class="skin" rel="stylesheet" type="text/css"
href="{{asset('/css/skin/skin-1.css')}}">
    <link rel="stylesheet" href="{{asset('plugins/datepicker/css/bootstrap-
datepicker.min.css')}}">
    <!-- Revolution Slider Css -->
    <link rel="stylesheet" type="text/css"
href="{{asset('plugins/revolution/revolution/css/layers.css')}}">
    <link rel="stylesheet" type="text/css"
href="{{asset('plugins/revolution/revolution/css/settings.css')}}">
    <link rel="stylesheet" type="text/css"
href="{{asset('plugins/revolution/revolution/css/navigation.css')}}">
    <!-- Revolution Navigation Style -->
</head>
<body id="bg">
<!--<div id="loading-area"></div-->
<div class="page-wrapper">
    <!-- header -->
    @yield('header')
    @yield('content')
    <!-- Footer -->
    @yield('footer')
    <!-- Footer -->

</div>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		92

```

<!-- JAVASCRIPT FILES ===== -->
<script src="{{asset('js/jquery.min.js')}}"></script><!-- JQUERY.MIN JS -->
<script src="{{asset('plugins/wow/wow.js')}}"></script><!-- WOW JS -->
<script src="{{asset('plugins/bootstrap/js/popper.min.js')}}"></script><!--
BOOTSTRAP.MIN JS -->
<script src="{{asset('plugins/bootstrap/js/bootstrap.min.js')}}"></script><!--
BOOTSTRAP.MIN JS -->
<script src="{{asset('plugins/bootstrap-select/bootstrap-
select.min.js')}}"></script><!-- FORM JS -->
<script src="{{asset('plugins/bootstrap-touchspin/jquery.bootstrap-
touchspin.js')}}"></script><!-- FORM JS -->
<script src="{{asset('plugins/counter/waypoints-min.js')}}"></script><!--
WAYPOINTS JS -->
<script src="{{asset('plugins/counter/counterup.min.js')}}"></script><!--
COUNTERUP JS -->
<script src="{{asset('plugins/imagesloaded/imagesloaded.js')}}"></script><!--
IMAGESLOADED -->
<script src="{{asset('plugins/masonry/masonry-3.1.4.js')}}"></script><!--
MASONRY -->
<script src="{{asset('plugins/masonry/masonry.filter.js')}}"></script><!--
MASONRY -->
<script src="plugins/owl-carousel/owl.carousel.js"></script><!-- OWL SLIDER -->
<script src="plugins/rangeslider/rangeslider.js"></script><!-- Rangeslider -->
<script src="js/custom.js"></script><!-- CUSTOM FUCTIONS -->
<script src="js/dz.carousel.js"></script><!-- SORTCODE FUCTIONS -->
<script src='js/recaptcha/api.js'></script> <!-- Google API For Recaptcha -->
<!--<script src="js/dz.ajax.js"></script> CONTACT JS -->
<script src="plugins/paroller/skrollr.min.js"></script><!-- PAROLLER -->
<!-- Go to www.addthis.com/dashboard to customize your tools -->

</body>

</html>

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		93