Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-застосунку для пошуку роботи засобами React, Java та MySQL

Виконав: студент IV курсу, групи СНс-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Жаров Б.А.
(підпис) (прізвище та ініціали)

Керівник _____ Дуда О.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Голотенко О.С.
(підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_20_» _червня_ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп’ютерні науки
(шифр і назва спеціальності)

Студенту Жарову Богдану Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка веб-застосунку для пошуку роботи засобами React, Java та. MySQL

Керівник роботи : Дуда Олексій Михайлович, кандидат технічних наук, доцент кафедри КН
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «7» лютого 2023 року № 4/7-133

2. Термін подання студентом завершеної роботи 21.06 2023р.

3. Вихідні дані до роботи Літературні та інтернет джерела про розробку веб-застосунків засобами React, Java та MySQL

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Постановка задачі, формування вимог,обгрунтування технологій розробки для веб-застосунку для пошуку роботи. 1.1 Аналіз предметної області. 1.2 Формування вимог до веб-застосунку пошуку роботи. 1.3 Пошук актантів та варіантів використання веб-застосунку для пошуку роботи. 1.4.Опис ключових варіантів використання застосунку для пошуку. роботи. 1.5 Оцінка методів розв’язання поставленої задачі. 1.6 Вибір середовища розробки. 1.7 Обґрунтування використовуваних технологій. 1.8 Висновок до першого розділу кваліфікаційної роботи. Розділ 2. Проєктування та реалізація веб-застосунку для пошуку роботи. 2.1 Моделювання архітектури веб-застосунку для пошуку роботи. 2.2 Структура веб-застосунку для пошуку роботи. 2.3 Структура серверної частини веб-застосунку для пошуку роботи. 2.4 Проектування поведінки веб-застосунку для пошуку роботи. 2.5 Розробка моделей даних для веб-застосунку для пошуку роботи. 2.6 Програмна реалізація основних структурних елементів веб-застосунку для пошуку роботи. 2.7 Встановлення та налаштування веб-застосунку для пошуку роботи. 2.8 Тестування веб-застосунку для пошуку Висновки. Перелік джерел

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці		05.06.2023	0.06.2023

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про вебтехнології та бази даних	24.01.2023-26.01.2023	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо розробка веб-	01.02.2023-07.02.2023	*Виконано*
	застосунку для пошуку роботи засобами React, Java та		
	Mysql		
5.	Розроблення веб-застосунку для пошуку роботи	07.02.2023-08.02.2023	*Виконано*
	засобами React, Java та Mysql		
6.	Оформлення розділу «Постановка задачі, формування	08.02.2023-09.02.2023	*Виконано*
	вимог, обгрунтування технологій розробки для веб-		
	застосунку для пошуку роботи»		
7.	Оформлення розділу «Проєктування та реалізація	10.02.2023-12.02.2023	*Виконано*
	веб-застосунку для пошуку роботи»		
8.	Виконання завдання до розділу «Безпека	05.06.2023-08.06.2023	*Виконано*
	життєдіяльності, основи хорони праці»		
9.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	19.06.2023	

Студент _____ Жаров Б.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дуда О.М
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-застосунку для пошуку роботи засобами React, Java та Mysql// Кваліфікаційна робота освітнього рівня «Бакалавр» // Жаров Богдан Андрійович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-41 // Тернопіль, 2023 // С. – 46 , рис. – 23 , табл. – 0 , кресл. –13 , додат. – 4 , бібліогр. – 30

Ключові слова: сервер, кліент, java, база даних, react, spring, javascript

Кваліфікаційна робота освітнього рівня «Бакалавр» присвячена розробці веб-застосунку для пошуку роботи, який має широкий функціонал та можливості масштабування.

У першому розділі кваліфікаційної роботи було складені вимоги, яким повинен відповідати такий веб-застосунок, розглянуто рішення, що вже існують на ринку, що також пропонують послуги пошуку роботи, вивчені їх переваги та недоліки, проведено відбір інструментів розробки, обґрунтовано вибір інформаційних і комунікаційних технологій, які використані при створені веб-застосунку для пошуку роботи.

В другому розділі кваліфікаційної роботи проведено детальне моделювання архітектури веб-застосунку для пошуку роботи, визначена структура веб-застосунку, а також структура серверної частини, розроблені моделі даних для веб-застосунку та реалізовані основні структурні елементи. здійснено встановлення та налаштування веб-застосунку, проведено даного веб-застосунку тестування.

У розділі "Безпека життєдіяльності, основи хорони праці" розглядаються різні аспекти, пов'язані з тяжкістю ураження електричним струмом та вимогами до режимів праці і відпочинку при роботі з ВДТ або комп'ютерними моніторами.

ANNOTATION

Website Development for Job Lookup by Means of React, Java and MySQL // Qualification work of the educational level "Bachelor" // Zharov Bohdan// Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-41 // Ternopil, 2023 // // P.44, fig. – 23, tabl. – 0, draw. –13, annexes. – 4 references – 30.

Keywords: server, client, java, database, react, spring, javascript

The bachelor's degree thesis is devoted to the development of a web-based job search application that has wide functionality and scalability.

In the first chapter of the qualification work, the requirements that such a web application must meet were compiled, solutions already existing on the market that also offer job search services were considered, their advantages and disadvantages were studied, development tools were selected, and the choice of information and communication technologies used to create a web application for job search was justified.

In the second chapter of the qualification work, a detailed modeling of the architecture of the web application for job search was carried out, the structure of the web application and the structure of the server part were determined, data models for the web application were developed and the main structural elements were implemented. installation and configuration of the web application was carried out, and testing of the web application was carried out.

The section "Life safety, basics of work safety" discusses various aspects related to the severity of electric shock and requirements for work and rest regimes when working with Visual Displays Terminals or computer monitors.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – прикладний програмний інтерфейс.

CSS (англ. cascading style sheets) – каскадні таблиці стилів.

DOM (англ. Document Object Model) – це незалежний від платформи і мови програмний інтерфейс, що дає змогу програмам і скриптам отримати доступ до вмісту HTML-файлів.

HTML (англ. hyper text markup language) – мова розмітки гіпертекстових документів).

JSON (англ. JavaScript Object Notation) – це текстовий формат обміну даними між комп'ютерами.

JSX (англ. JavaScript Syntax Extension) – це розширення React синтаксису мови JavaScript.

JWT (англ. JSON Web Token) – це стандарт токена доступу на основі JSON.

ВДТ – візуальний дисплейний термінал.

ПК – персональний комп’ютер.

ЗМІСТ

ВСТУП

Актуальність теми. З розвитком технологій та віртуального середовища, пошук роботи став більш доступним та зручним. Багато компаній та рекрутингових агентств активно використовують соціальні мережі та спеціалізовані сайти, щоб знайти кандидатів для робочих місць. Крім того, існують застосунки та сервіси для пошуку роботи, що аналізують профіль користувача, його навички та досвід, щоб підібрати найбільш відповідні вакансії та запропонувати їх користувачеві. Деякі з цих застосунки можуть надавати можливості для розміщення резюме, підписки на нові вакансії, відстеження статусу заявки на роботу та сповіщення про нові можливості. Крім того, деякі з них також мають функції для зв'язку з роботодавцями та здійснення онлайн співбесід, що робить процес пошуку роботи більш швидким та зручним. Також варто зазначити, що багато людей працюють з дому, віртуальні робочі місця та проекти стали більш поширеними. Тому, розробка веб-застосунків для пошуку роботи з використанням інноваційних підходів до проектування програмного забезпечення є актуальним напрямком досліджень.

Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення проінформованості громадян про можливості пошуку роботи та організація високоякісних послуг що до взаємодії осіб що пропонують роботу та громадян що її шукають.

Для досягнення поставленої мети було сформовано ряд наступних завдань:

– провести аналіз предметної області;
– виконати проектування логіки роботи веб-застосунку;
– обрати найбільш ефективні технології ;
– провести розробку та тестування веб-застосунку для пошуку роботи.

Практичне значення одержаних результатів: в ході виконання кваліфікаційної роботи розроблено веб-застосунок для пошуку роботи проведено його тестування і встановлення.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ, ФОРМУВАННЯ ВИМОГ, ОБГРУНТУВАННЯ ТЕХНОЛОГІЙ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПОШУКУ РОБОТИ

1.1 Аналіз області пошуку роботи

В процесі виконання кваліфікаційної роботи буде розроблено веб-застосунок для пошуку роботи. Потенційними замовниками можуть стати компанії які хочуть автоматизувати власні процеси по пошуку нових працівників, так і компанії-посередники. В будь-якому випадку основними сутностями буде працедавець і працеємець.

1.2 Формування вимог до веб-застосунку пошуку роботи

Відповідно до побажань замовника в веб-застосунку для пошуку роботи повинні бути закладені наступні функціональні можливості:

- можливість автоматичної реєстрації нових користувачів в системі;
- можливість розподілу користувачів на ролі;
- можливість створення вакансій для ролі роботодавців;
- можливість пошуку вакансій за категоріями для усіх користувачів;
- можливість відгукнутись на вакансію для ролі працеємця;
- можливість створювати категорії для ролі адміністратора.

В застосунку для пошуку роботи повинні бути реалізовані наступні структурні особливості:

- застосунок для пошуку роботи повинен містити серверну частину;
- веб-застосунок для пошуку роботи повинен містити клієнтську веб-частину;
- в якості сховища даних необхідно використати реляційну базу даних.

Враховуючи наведені вище особливості веб-застосунку для пошуку роботи, було сформувано наступні вимоги до захисту:

– в системі не повинно бути функцій відновлення паролів. Дії з відновлення паролів повинні виконуватись користувачами з вищим рівнем доступу;

– при вході користувача в систему повинна відбуватись парольна аутентифікація користувача;

– забезпечення авторизації повинно реалізовуватися за допомогою спеціально згенерованого JWT-токену.

Про аналізувавши вимоги поставлені до веб-застосунку для пошуку роботи можна переходити до пошуку актантів і варіантів використання

1.3 Пошук актантів та варіантів використання веб-застосунку для пошуку роботи

1.3.1 Пошук актантів

Будь-яка програмна система працює у деякому контексті, що визначає зовнішнє оточення системи. Таке оточення формують користувачі (або актори) системи, якими можуть слугувати як люди, так і системи. Кожен з акторів взаємодіє з системою за своєю власною схемою та очікує від системи певної поведінки й реакції.

У застосунку для пошуку роботи фігуруватимуть наступні актори:

– незареєстрований користувач;

– працеємець;

– працедавець (далі компанія);

– адміністратор.

Таким чином діаграма акторів веб-застосунку для пошуку роботи має вигляд (див. рисунок 1.1).

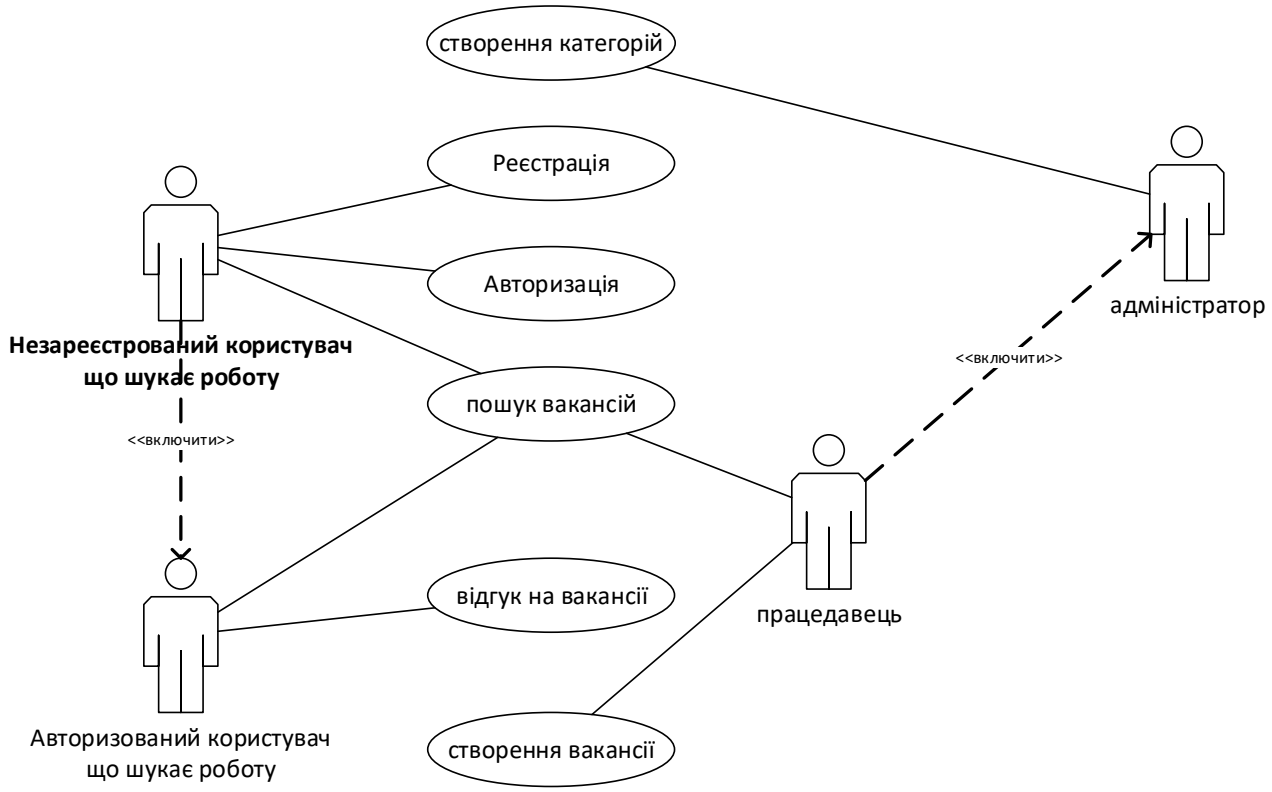

Рисунок 1.1 – Актори веб-застосунку для пошуку роботи

У даному випадку веб-застосунок для пошуку роботи є посередником в спілкуванні між компанією і працеємцем.

1.3.2 Варіанти використання веб-застосунку для пошуку роботи

Виходячи з опису предметної області та основних акторів можна сформувати ключові варіанти використання веб-застосунку. Їх короткий опис:

– усі користувачі (втому числі незареєстрований) – Пошук вакансій по категоріям;

– працеємець – можливість відгукнутись на вакансію;

– компанія – можливість створити вакансію, отримати список кандидатів які відгукнулися на вакансії компанії;

– адміністратор – можливість додавати категорії.

1.4 Опис ключових варіантів використання веб-застосунку для пошуку роботи

Оскільки головною причиною створення веб-застосунку для пошуку роботи це є взаємодія головно. сутністю, тобто вакансією, то ключовим варіантом використання є створення вакансії (див. рисунок 1.2).

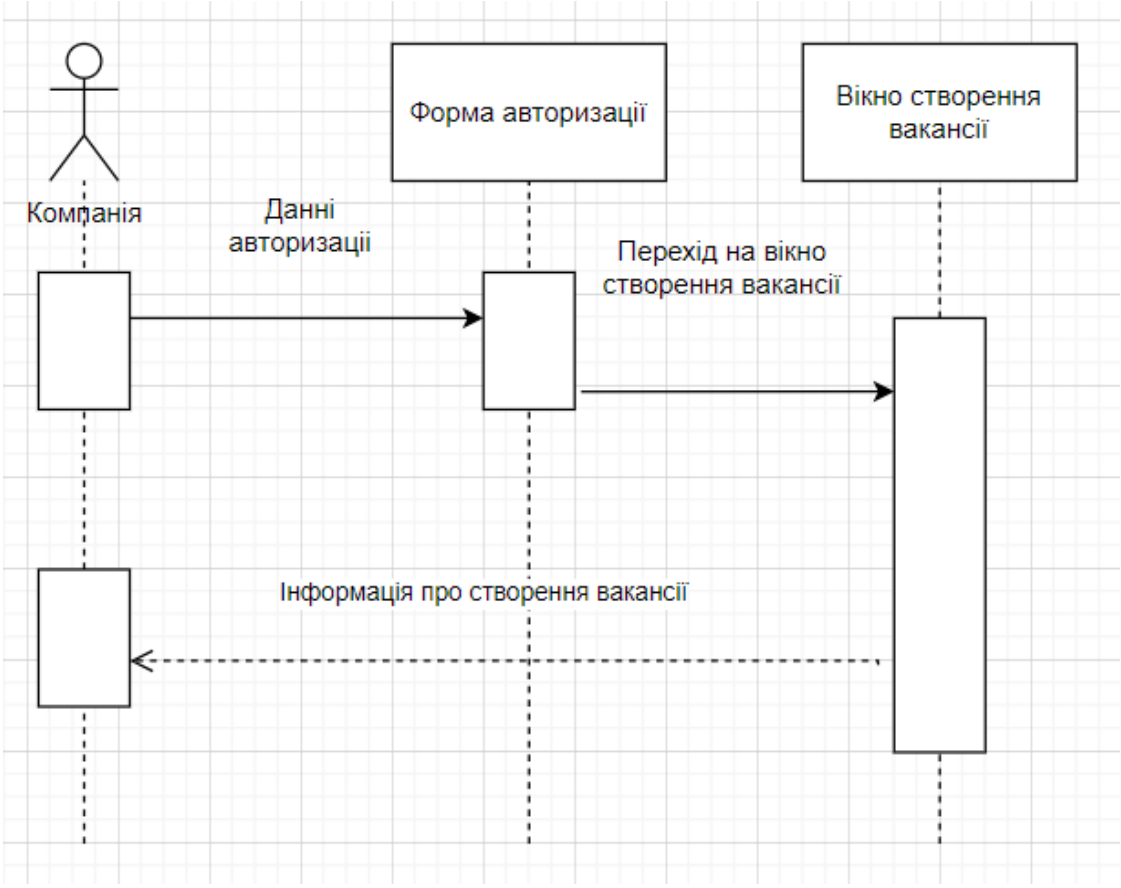

Рисунок 1.2 – UML-діаграма для процесу створення вакансії

Для створення вакансії компанії необхідно успішно авторизуватися і перейти до форми створення вакансії, і далі створити нову вакансію. Для того щоб відгукнутися на щойно створену вакансію, працеємець повинен також авторизуватися і знайти цю вакансію. Якщо користувач уже взаємодіяти з цією вакансією, сторінка покаже помилку (див. рисунок 1.3).

Рисунок 1.3 – UML-діаграма для процесу створення «відгуку» на вакансію

Після «відгуку» компанія-власниця вакансії отримає повідомлення з електронною поштою працеємця.

1.5 Оцінка методів створення веб-застосунку для пошуку роботи

Оцінка методів створення веб-застосунку для пошуку роботи залежить від різних факторів, таких як час, ресурси, складність і мета проекту. Ось кілька методів, які можна використовувати:

– розробка власного застосунку: Цей метод вимагає повноцінного розроблення веб-застосунку з нуля, включаючи фронтенд та бекенд. Він дає повний контроль над функціональністю та дизайном застосунку, але може зайняти багато часу та ресурсів. Використання популярних фреймворків, таких як React або Angular[1] для фронтенду та Spring або Django[2] для бекенду, може спростити розробку;

– використання готових платформ: Існують готові платформи для створення веб-застосунків, такі як WordPress [3] або Drupal [4]. Вони надають базовий функціонал та шаблони, які можна настроювати під свої потреби. Цей метод дозволяє швидко запустити веб-застосунок, але може бути обмежений в гнучкості та можливостях налаштування;

– використання готових рішень або API: Є веб-сервіси та API, які надають готові рішення, такі як Indeed або LinkedIn API. Вони дозволяють інтегрувати функціональність пошуку роботи безпосередньо в веб-застосунок. Цей метод дозволяє швидко отримати доступ до великої бази вакансій, але може бути обмежений в гнучкості та настроюванні;

– використання комбінації готових компонентів: Використання готових компонентів та бібліотек, таких як Bootstrap [5] або Material-UI, разом з фреймворком, таким як React або Vue.js [6], може допомогти в швидкому розробленні веб-застосунку для пошуку роботи. Цей метод комбінує переваги готових компонентів з можливістю настроювання та розширення.

Вибір методу залежить від ваших потреб, умов розробки та ресурсів, які ви готові вкласти. Важливо ретельно оцінити кожен метод відповідно до ваших конкретних вимог та можливостей, щоб забезпечити успішний результат проекту.

1.5.1 Вибір оптимального методу для розробки веб-застосунку пошуку роботи

Оскільки веб-застосунок для пошуку роботи є досить складною по функціоналу програмою недоцільно використовувати метод використання готових платформ адже від обмежує потенційний набір функцій які можуть бути імплементовані. Метод готових платформ не доцільно використовувати на ранніх етапах оскільки це значно збільшить вартість розробки. У випадку застосунку для пошуку роботи використовується комбінований метод розробки

серверна частина буде розроблена повністю самостійно, а клієнтська частина з використанням методу комбінації готових елементів.

1.5.2 Життєвий цикл веб-застосунку у для пошуку роботи

Життєвий цикл розробки програмного забезпечення [7] - представляє послідовність етапів розробки, які необхідні для створення проекту, починаючи від його початкової концепції й до випуску продукту на ринок та його подальшої підтримки. Є 7 основних фаз життєвого циклу розробки веб-рішень:

– Етап планування. Мета етапу: зрозуміти який продукт має бути на виході;

– Етап збору вимог. Мета етапу: зібрати та задокументувати вимоги до майбутнього програмного забезпечення.

– Етап проектування та прототипування. Мета етапу: втілити вимоги до розробки в дизайн.

– Етап розробки програмного забезпечення. Мета: створення реального програмного забезпечення.

– Етап тестування програмного забезпечення. Мета: забезпечити відповідність готового продукту узгодженим вимогам.

– Етап випуску готового продукту на ринок. Мета: надати готове програмне забезпечення кінцевим користувачам.

– Етап експлуатації та технічного обслуговування.

Останній етап означає, що підтримка відбувається готового продукту і його активна розробка закінчується.

1.6 Вибір середовища розробки веб-застосунку для пошуку роботи

Для розробки серверної частини веб-застосунку для пошуку роботи буде використано IntelliJ IDEA [8] (див. рисунок 1.7) – середовище розробки для різних мов програмування від компанії JetBrains.

Рисунок 1.7 – Ярлик програми IntelliJ IDEA

IntelliJ IDEA є потужним та повнофункціональним інструментом розробки. Він надає широкий набір функцій, таких як автодоповнення коду, рефакторинг, налагодження, система керування версіями та багато іншого. Екосистема плагінів: IntelliJ IDEA має широку екосистему плагінів, яка дозволяє розширювати його функціональність та налаштовувати під конкретні потреби розробки. Розробники можуть встановлювати плагіни для підтримки різних фреймворків, шаблонів коду, стилей форматування, інструментів аналізу та багато іншого. Це дозволяє налаштувати IntelliJ IDEA під свої потреби та стиль розробки.

Для розробки клієнтської частини веб-застосунку для пошуку роботи буде використано VS Code [9] (див. рисунок 1.8) – зручний текстовий редактор для створення сучасних веб застосунків і програм для хмарних систем.

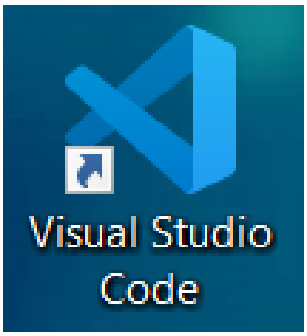

Рисунок 1.8 – Ярлик програми Visual Studio Code

VS Code відомий своєю простотою та інтуїтивним інтерфейсом користувача. Він має широкий набір функцій також надає розширені

можливості налаштування, дозволяючи користувачам налаштовувати його під свої потреби та робочий процес. VS Code має велику екосистему плагінів, які дозволяють розширювати його функціональність. Розробники можуть встановлювати плагіни для підтримки різних мов програмування, фреймворків, інструментів розробки та інших корисних функцій. Це дозволяє налаштувати VS Code під свої потреби та стиль роботи. Також VS Code має підтримку інтеграції з різними інструментами та сервісами розробки, такими як системи керування версіями (Git), системи автоматичної збирання (npm, Maven), сервіси хмарного розробки (Azure, AWS) та інші. Це дозволяє зручно працювати з різними інструментами та сервісами безпосередньо з VS Code.

Для роботи з базами даних можна використовувати звичайну консоль або програми клієнти які пропонуються розробниками баз даних.

1.7 Обґрунтування використовуваних інформаційних і комунікаційних технологій

Оскільки веб-застосунок для пошуку роботи поділений на модулі . Для серверного модуля обрано такий перелік технології:

– Java [10] – це об'єктно-орієнтована мова програмування. Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для певної платформи. Це означає, що ви можете розробляти веб-застосунки на Java і запускати їх на різних операційних системах, таких як Windows, macOS та Linux, не виконуючи для цього ніяких додаткових дій.

Java є високопродуктивною мовою програмування, особливо коли йдеться про обробку великих обсягів даних та високонавантажених веб-застосунків. Вона має ефективну систему управління пам'яттю та оптимізовані механізми виконання, що дозволяють добре масштабувати застосунки та забезпечувати швидку відповідь.

Також Java відома своїми вбудованими механізмами безпеки. Вона надає інструменти для автоматичного керування пам'яттю, відстеження помилок та

перехоплення винятків. Це допомагає зменшити ймовірність вразливостей, таких як помилки пам'яті та переповнення буферу, і забезпечує надійність веб-застосунків.

– Spring Framework [11] – це програмний каркас (фреймворк) з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java. Перевагами цього фреймворку є простота розробки ПЗ і наявність широкого спектру можливостей для роботи з базами даних і досить потужний механізм контейнерів Java-об’єктів. Spring пропонує механізм Інверсії управління, також відомий як контейнер IoC(Inversion of Control). Це дозволяє розробникам визначати залежності між компонентами застосунку, а контейнер IoC автоматично їх управляє. Це спрощує процес розробки та підтримки застосунків, а також дозволяє використовувати принципи розробки, такі як Dependency Injection (DI), для полегшення тестування та розширення коду.

Spring пропонує вбудований веб-фреймворк Spring MVC(Model-View-Controller), який надає потужні інструменти для розробки веб-застосунків. Spring MVC розділяє логіку застосунку на моделі, представлення та контролери, що сприяє розділенню відповідальностей та покращує підтримку керування станом обробки запитів та відображення даних.

Для клієнтського веб модуля обрано такий перелік технологій:

– React [12] – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. React базується на компонентній архітектурі, де весь інтерфейс користувача розбивається на невеликі незалежні компоненти. Це дозволяє зручно організувати код, забезпечуючи повторне використання компонентів, зрозумілість структури та легкість розширення.

React використовує віртуальний DOM, що дозволяє оптимізувати процес оновлення інтерфейсу. Замість прямого маніпулювання реальним DOM, React

працює з віртуальним DOM, який є легким та швидким для обробки. React автоматично визначає необхідні зміни в реальному DOM та ефективно оновлює його, забезпечуючи оптимальну продуктивність.

– Бібліотеки стилів і компонентів Tailwind CSS і Mantine. Головною перевагою є те що їх використання дозволяє не створювати власні CSS стилі, що пришвидшує розробку ПЗ Використання Tailwind CSS та Mantine в поєднанні дозволяє розробникам ефективно будувати стильні, гнучкі та функціональні веб-застосунки. Комбінація широкого набору готових стилів та компонентів з можливістю настроювання та розширення дозволяє швидко реалізувати проект з високою якістю та забезпечити потреби клієнтів.

– MySQL [13] – це відкрите програмне забезпечення для управління базами даних, яке використовує мову запитів SQL (Structured Query Language). Основне призначення MySQL полягає у зберіганні та управлінні великими обсягами даних, що використовуються веб-додатками, такими як веб-сайти та програми. Воно забезпечує широкі можливості для створення, модифікації та оптимізації баз даних. MySql обрано для розробки веб-застосунку для пошуку роботи, через такі особливості:

– висока продуктивність;
– доступність;
– масштабованість та гнучкість;
– надійна транзакційна підтримка;
– переваги інтернету та сховища даних;
– надійний захист даних;
– комплексна розробка застосунку;
– простота керування.

1.8 Висновок до першого розділу кваліфікаційної роботи

У першому розділі кваліфікаційної роботи було проведено докладний аналіз предметної області. Було визначено ключові аспекти, які впливають на

створення веб-застосунку для пошуку роботи. На основі цього аналізу були складені вимоги, які повинен відповідати такий веб-застосунок. Крім того, були розглянуто рішення, що вже існують на ринку, які також пропонують послуги пошуку роботи. Були вивчені їх переваги та недоліки. Після цього було проведено відбір інструментів розробки і обґрунтована обрані технології веб-застосунку для пошуку роботи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПОШУКУ РОБОТИ

2.1 Моделювання архітектури веб-застосунку для пошуку роботи

На основі проведеного аналізу області пошукуроботи було зроблено висновок про необхідніть побудови застосунку в стилі трирівневої [14] клієнт-серверної архітектури Доцільність її використання обґрунтовується необхідністю розподілу системи на рівні, кожен з яких матиме своє функціональне призначення. Отримання інформації про вакансію з використанням цієї архітектури показано на рисунку 2.1.

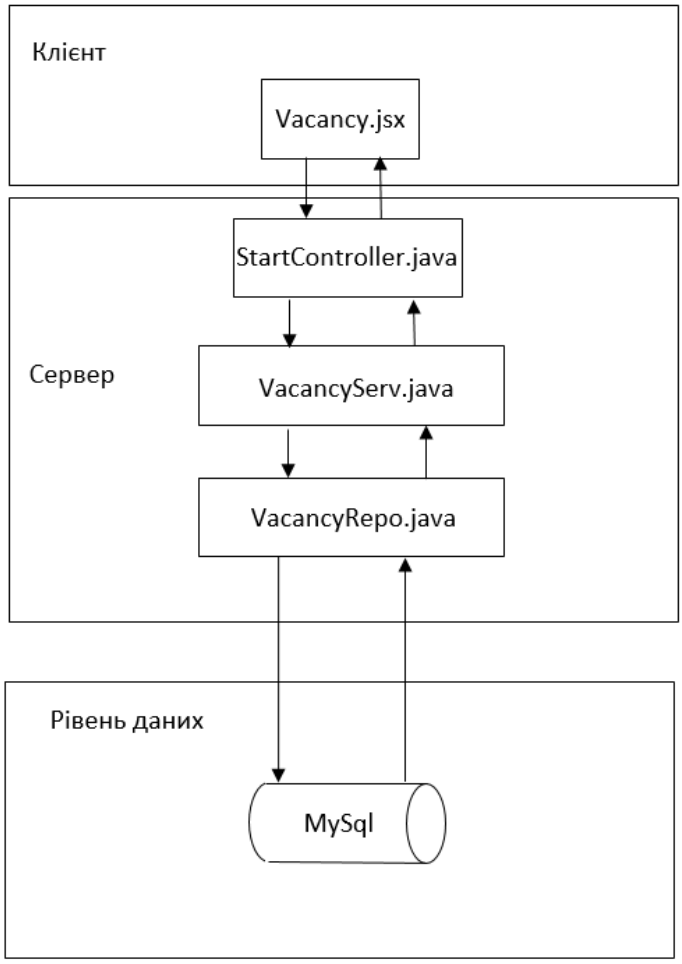

Рисунок 2.1 – Трирівнева архітектура веб-застосунку для пошуку роботи

Рівень клієнта представлений набором JSX-шаблонів, приклад VacancyPage.jsx показано в додатку А. Ці елементи являють собою об'єднання HTML - шаблонів , CSS - стилів і JavaScript файлів. Шляхом використання описаної клієнтської частини браузер отримує можливість надіслати дані вакансії, яку хоче оглянути користувач далі – на рівень сервера за допомогою HTTP запитів.

Рівень сервера представлений набором Java – класів, в яких описується логіка поведінки застосунку: спочатку дані опрацьовуються класом-контролером (див. додаток Б), який передає їх до класу-сервісу(див. додаток В), який передає інформацію до класу-репозиторію (див. додаток Г) де і відбувається запит до бази даних. Цей рівень є посередником між рівнем клієнта і рівнем даних.

2.2 Структура веб-застосунку для пошуку роботи

2.2.1 Структура клієнтської частини веб-застосунку для пошуку роботи

Структурна модель клієнта веб-застосунку для пошуку роботи включає наступні пакети:

– App.jsx і public/index.html – вхідна точка для застосунку;

– пакет «components» – найменші елементи в ієрархії застосунку. Зазвичай використовуються для відображення контенту який багаторазово повторюється. Наприклад в контексті застосунку для пошуку роботи це елементи, які відображують вакансії;

– пакет «layouts» – статичні елементи які беруть участь в формуванні головних сторінок. Цими елементами можуть бути header або footer;

– пакет «hooks» – власні написані хуки;

– пакет «services» – пакет з методами, які відправляють запити на серверну сторону;

– пакет «pages» – пакет з основними сторінками застосунку.

При розробці застосунків за допомогою React [15] слід використовувати саме таку структуру, адже при подальшому розширені програм відбудеться значне збільшення кількості файлів в яких буде складніше орієнтуватись.

2.3 Структура серверної частини веб-застосунку для пошуку роботи

Структурна модель серверної частини веб-застосунку для пошуку роботи розроблена за допомогою фреймворка Spring [16] і включає наступні пакети:

– пакет «controller» – пакет з класами, які відповідають за первинне отримання інформації від клієнта веб-застосунку для пошуку роботи;

– пакет «service» – пакет з класами, які є проміжним шаром між класами контролерами інтерфейсами репозиторіїв, можуть форматувати отриманні дані;

– пакет «repo» – пакет з інтерфейсами в яких формується безпосередні запити в базу даних;

– пакет «model» – пакет з класами сутностями;

– пакет «security» – пакет з класами, які налаштовують роботу JWT токену.

Така структура застосунку є рекомендованою оскільки дозволяє добре орієнтуватися в компонентах програми і без труднощів розширювати його функціонал.

2.4 Проектування поведінки веб-застосунку для пошуку роботи

Перелік компонентів застосунку для пошуку роботи:

– HomePage.jsx – показ головної сторінки;

– LoginPage.jsx – авторизація;

– RegistrationPage.jsx – реєстрація;

– SearchPage.jsx – пошук вакансій;

– VacancyPage.jsx – перегляд розширеної інформації про вакансію.

Функціональна схема взаємодії програмних модулів веб-застосунку для пошуку роботи зображена на рисунку 2.2. На ній зображені функціональні модулі, їхня взаємодія.

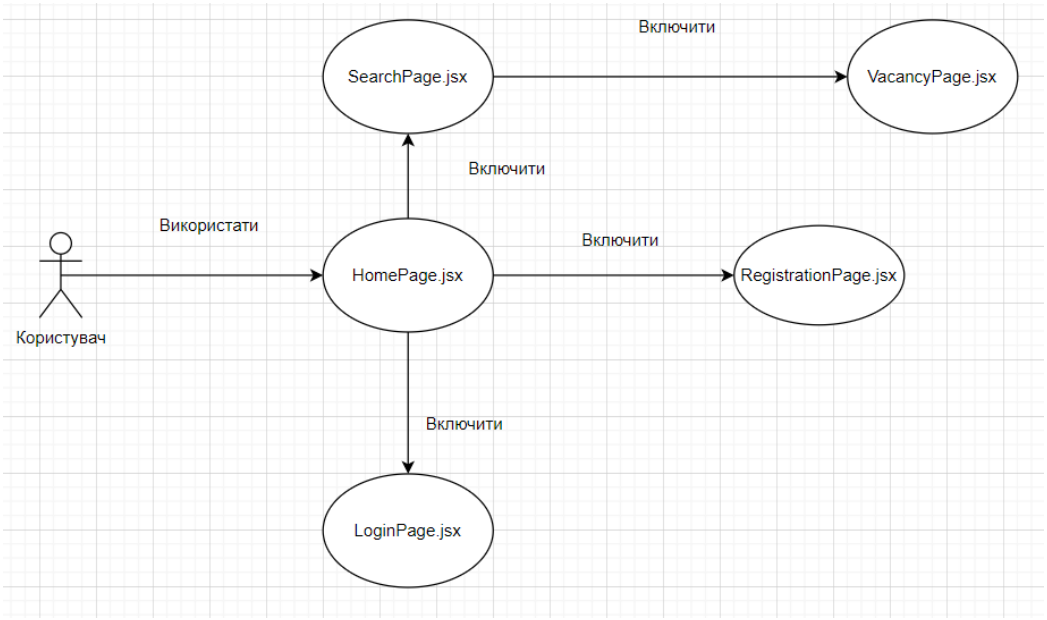

Рисунок 2.2 – Функціональна схема взаємодії програмних модулів

Першою сторінкою куди потрапляє користувач є Головна сторінка. Далі після переходу можна отримати різний функціонал, проте макет всіх сторінок є ідентичним (див. рисунок 2.3).

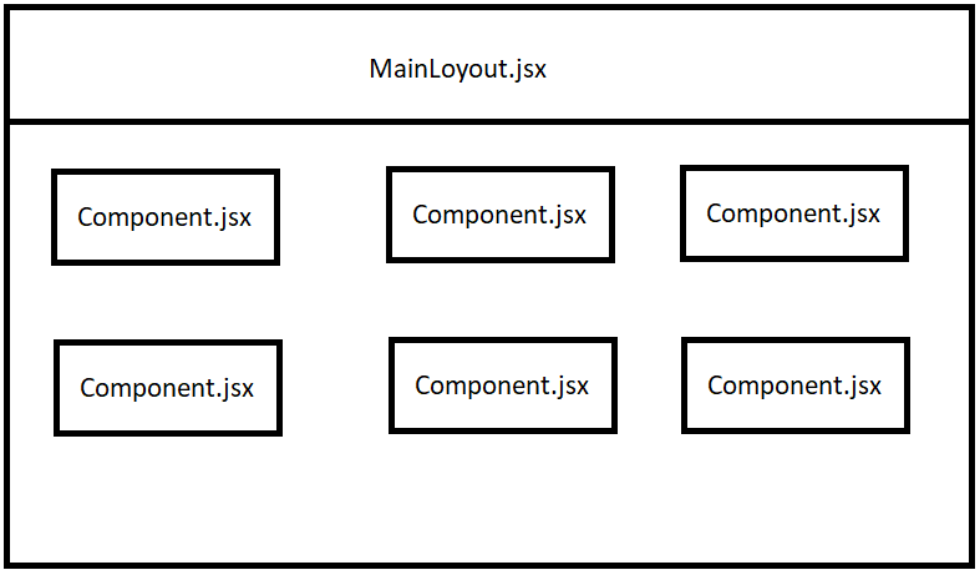

Рисунок 2.3 – Спрощений макет сторінки веб-застосунку для пошуку роботи

Кожна сторінка окрім логіна і реєстрації має хедер, який складається з власних компонентів.

2.5 Розробка моделей даних для веб-застосунку для пошуку роботи

Проаналізувавши предметну область та вимоги до інформаційної системи сформуємо перелік інформаційних сутностей:

– «accounts» – загальна сутність користувача;
– «client» – сутність з додатковою інформацією про працеємців;
– «company» – додаткова інформація про компанії;
– «vacancy» – загальна сутність вакансії;
– «category» – сутність категорії;
– «teamwork» – сутність, яка відповідає за співпрацю між працеємцем і компанією.

Визначившись з переліком інформаційних сутностей можемо перейти до проектування концептуальної моделі бази даних.

2.5.1 Проектування концептуальної моделі бази даних веб-застосунку для пошуку роботи

Для забезпечення взаємодії між різними сутностями веб-застосунку для пошуку роботи було введено декілька проміжних таблиць. Модель фінальної бази даних зображено на рисунку 2.4.

Модель містить 8 таблиць. Однією з найголовнішою сутністю є vacacncy яка містить містить інформацію про вакансію: назву ,опис і власника компенсацію. Таблиця account містить інформацію про користувачів. таблиця містить зв’язок один-до-одного з company якщо користувач є компанією, або до client, якщо він є працеємець. Teamwork дозволяє фіксувати співпрацю між компанією і працеємцем.

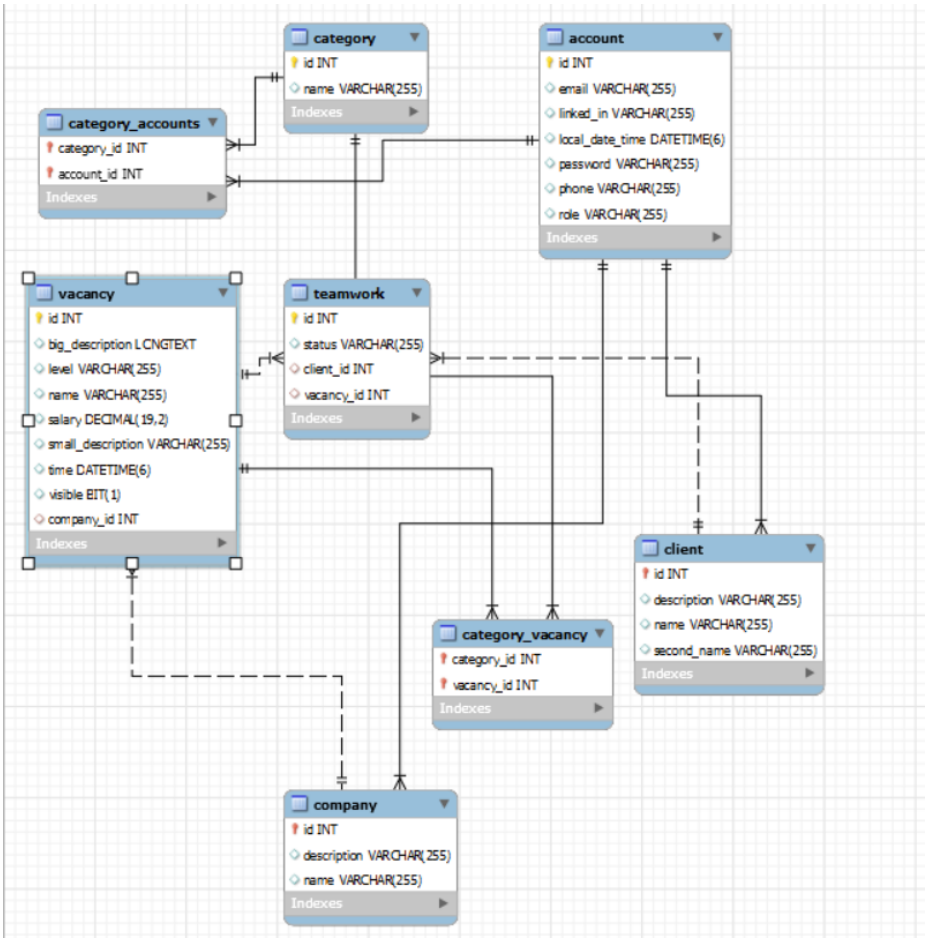

Рисунок 2.4 – Модель фінальної бази даних веб-застосунку для пошуку роботи

Проміжна таблиця category_accounts дає можливість надати в характеристиках акаунту декілька категорій. Схоже призначення і в таблиці category_vacany, тільки декілька категорій можна призначити для сутності вакансій.

2.6 Програмна реалізація основних структурних елементів веб-застосунку для пошуку роботи

Оскільки усі структурні елементи працюють за однаковим алгоритмом, для демонстрації реалізації буде взято функціонал переходу на сторінку обраної вакансії.

2.6.1 Програмна реалізація основних структурних модулів клієнтської частини веб-застосунку для пошуку роботи

У даному підрозділі буде опущено реалізацію серверної частини веб-застосунку. Перш за все на сторінці пошуку було натиснуте посилання яке перенаправляє на сторінку даної вакансії. Далі за допомогою хука useParam() [17] отримується id з url:

```
const { vacancyId } = useParams();
```

Після цього виконується хук життєвого циклу компонента useEffect() [18] зображений в лістингу 2.1, який перехоплює те що відбулася зміна id поточної вакансії:

Лістинг 2.1 – Хук useEffect

```
useEffect( () =>{
     getVacancyById(vacancyId)
          .then( (newVacancy) =>{
          setVacancy(newVacancy)
          })
     },[vacancyId]);
```

В тілі хука виконується метод – запит з сервісу vacancyService до серверної частини веб-застосунку для пошуку роботи. Метод використовує написаний хук useHttp(), який в свою чергу і відправляє запит на сервер. Реалізація методу

```
const getVacancyById =  async function(id){
     const response = await request(`${_URL}vacancies/${id}`);
     return  response;}
```

Після успішного запиту приходить інформація з серверу в форматі JSON яка перетворюється в об'єкт і призначається змінній Vacancy. Через створення змінної vacancy за допомогою хука useState()[19]:

```
const [vacancy,setVacancy] = useState({});
```

Змінну варто змінювати методом setVacancy. Після успішного присвоювання запускається процес формування HTML-сторінки. Результат успішного переходу на сторінку вакансії показано на рисунку 2.5.

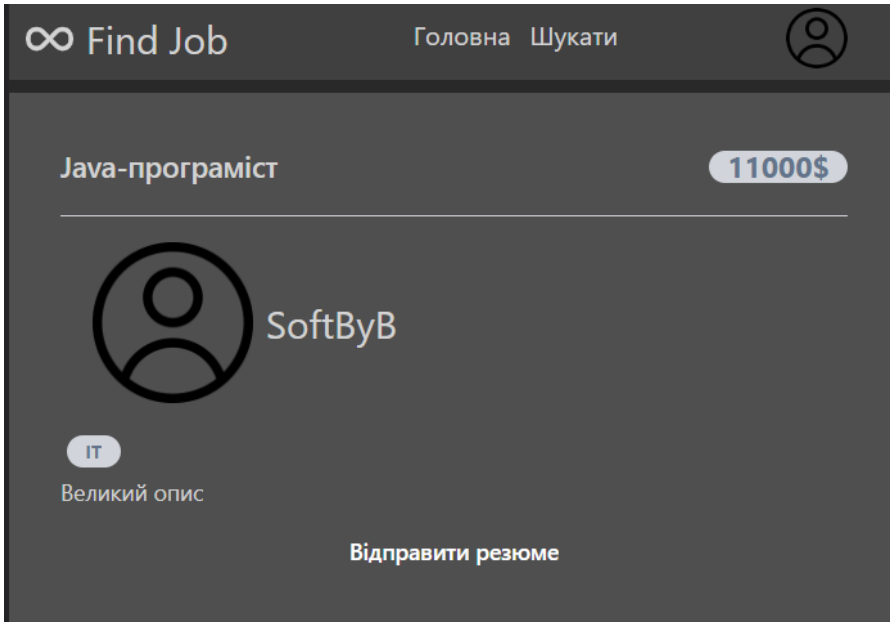

Рисунок 2.5 – Сторінка заданої вакансії

2.6.2 Програмна реалізація основних структурних модулів серверної частини веб-застосунку для пошуку роботи.

Після надсилання з веб-клієнтом запиту, першим буде його оброблювати клас StartController, а саме метод:

```
@GetMapping("/vacancies/{id}")
public VacDto getVacancyById(@PathVariable Integer id){
     return vacancyServ. getVacancyById}
```

Анотація @GetMapping [20] показує що це GET-метод. Анотація @PathVariable [21] дозволяє отримати id вакансії з запиту. Далі даний id

передається до сервісу vacancyServ. Далі сервіс викликає метод getVacancyById зображений на лістингу 2.2:

Лістинг 2.2 – Реалізація методу getVacancyById

```
public VacDto getCategoryById(Integer id){
     Vacancy vacancy = vacancyRepo.getById(id);
     VacDto dto = new VacDto();
     dto.setId(vacancy.getId());
     dto.getCompany().put("id",vacancy.getCompany().getId());
     dto.getCompany().put("name",vacancy.getCompany().getName())
     dto.getCompany().put("img",SERVER_URL+"files/"+
vacancy.getCompany().getId());
     return dto;
}
```

Спочатку виконується запит в базу даних vacancyRepo.getById(id) і далі відбується формування відповіді. Після того, як об’єкт відповіді сформований, він відправляється до класу-контролера. Далі цей файл надсилається клієнтську частину веб-застосунку у форматі JSON.

2.6.3 Розподіл доступу до застосунку за допомогою JWT токену

В залежності від ролі користувача зовнішній вигляд деяких елементів дещо відрізняється. Наприклад навігаційна панель незареєстрованого користувача буде мати вигляд показаний на рисунку 2.6.

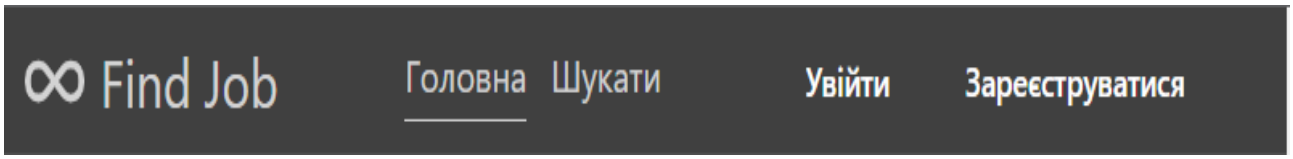

Рисунок 2.6 – Навігаційне вікно незареєстрованого

Проте для авторизованого користувача воно має інший вигляд (див. рисунок 2.7)

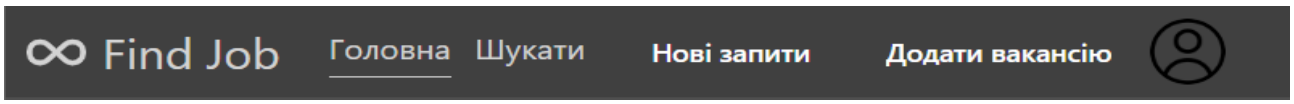

Рисунок 2.7 – Навігаційне вікно незареєстрованого

Таку поведінку веб-застосунку можна забезпечити за допомогою використання JWT-токену і React Context. JWT [22] – використовується для передачі даних для аутентифікації в клієнт-серверних програмах. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який надалі використовує цей токен для підтвердження своєї особи. Він має такий вигляд:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwib
mFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fw
pMeJf36POk6yJV_adQssw5c
```

React Context – дозволяє зберігати стан певних зміних і мати до них доступ з будь якої точки веб-застосунку. Для цього потрібно створити контекс у точці входу в веб застосунок і зберегти певний стан:

```
const LoginContext = React.createContext("");
<LoginContext.Provider value={{ isAuth, role, logout, login }}/>
```

Механізм розділення ролей такий:

– Користувач авторизується тим самим отримує JWT-токен і роль;

– В злежності від ролі показується певний набір функцій;

– Після завершення роботи користувач виходу з системи користувач втрачає токен і роль.

В серверній частині за оброблення запиту з токеном відповідає анотація:

```
@SecurityRequirement(name = "Bearer Authentication")
```

яка означає що перед виконанням запиту відбудеться перевірка токену на валідність. В разі помилки буде повернуто помилку 403[23].

2.7 Встановлення та налаштування веб-застосунку для пошуку роботи

Оскільки веб-застосунок для пошуку роботи є клієнт-серверною програмою, то з боку користувача для початку роботи необхідно мати лише доступ до інтернету і браузер. Для серверної частини веб-застосунку необхідно встановити останню версію Java. Також необхідно провести конфігурацію файлу application.properties[24], заповнити такі поля:

```
spring.datasource.url=
spring.datasource.username=
spring.datasource.password=
```

Які залежать від адреси бази даних, користувача бази даних і його паролю до бази даних відповідно. Наступним кроком буде додавання в файл конфігурації pom.xml плагіну[25] який дозволяє будувати jar-файли. Плагін подано в лістингу 2.3:

Лістинг 2.3 – Плагін для побудови jar-файлів

```
<build>
     <plugins>
          <plugin>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-maven-plugin</artifactId>
          </plugin>
     </plugins>

</build>
```

Далі необхідно побудувати jar-файл за допомогою команди maven [26]:

```
mvn install -skipTests
```

Отриманий файл необхідно завантажити на хостинг. Фінальним кроком налаштування є створення користувача в базі даних і надання йому права адміністратора, та створити бази даних.

2.8 Тестування веб-застосунку для пошуку роботи

В ході тестування ручного веб-застосунку для пошуку роботи не було виявлено суттєвих помилок, які б частково або повністю перешкоджали виконанню запланованих функцій. Для серверної частини застосунку для пошуку роботи використовувалася Java, тому у випадку використання некоректного синтаксису програма припиняє свою роботу. Для клієнтського застосунку було встановлено розширення ESLint [27].Вагомих помилок знайдено не було.

2.9 Експлуатація веб-застосунку для пошуку роботи

На початку роботи веб-застосунок для пошуку роботи показує його головну сторінку показану на рисунку 2.8.

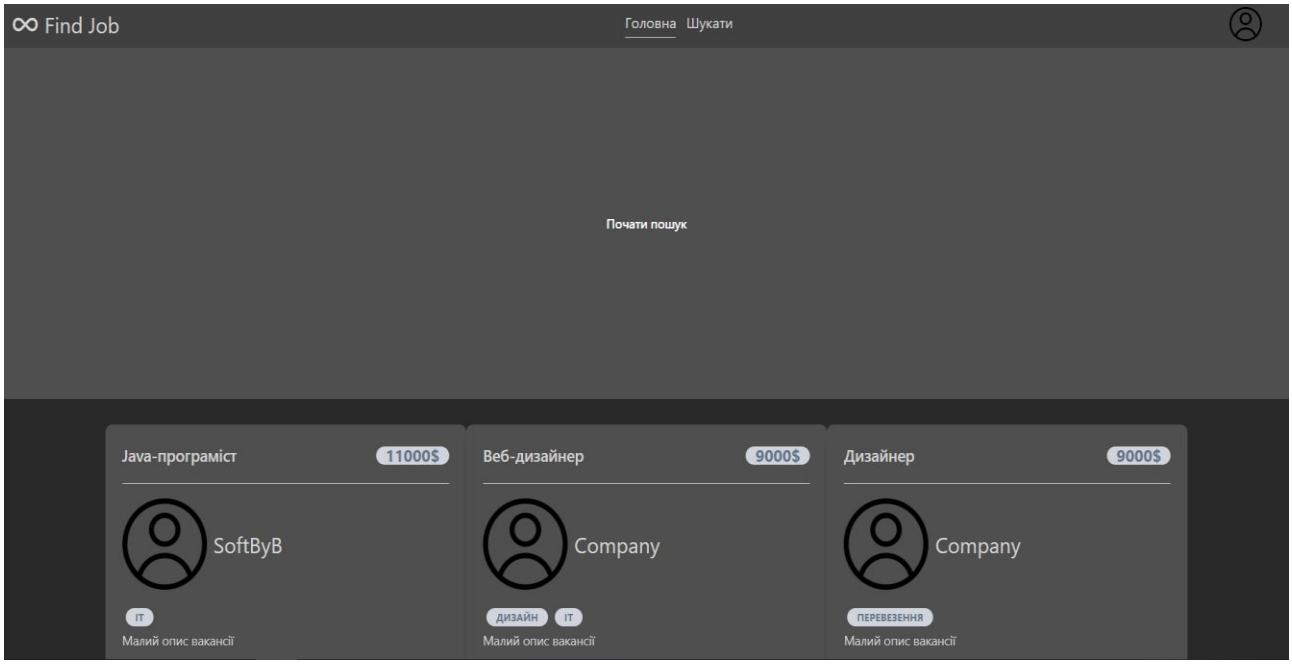

Рисунок 2.8 – Головна сторінка веб-застосунку для пошуку роботи

Далі для повноцінної роботи необхідно пройти процес авторизації або реєстрації. Сторінку авторизації веб-застосунку для пошуку роботи показано на рисунку 2.9. Авторизація проводиться під роллю компанії.

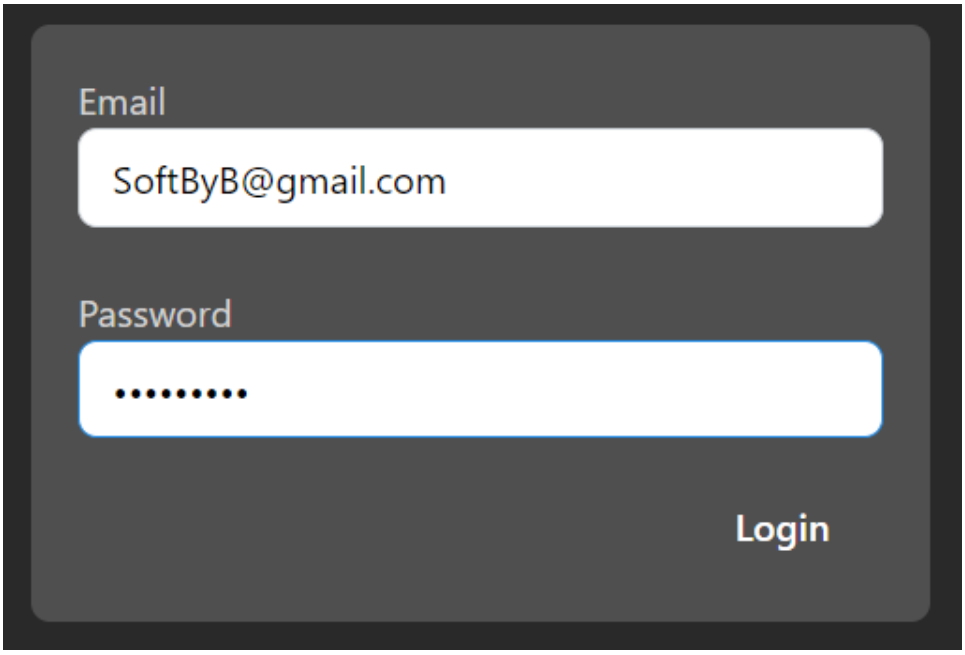

Рисунок 2.9 – Сторінка авторизації веб-застосунку для пошуку роботи

Після завершення авторизації можна починати пошук вакансій. Для цього у веб-застосунку для пошуку роботи було розроблену сторінку показану на рисунку 2.10.

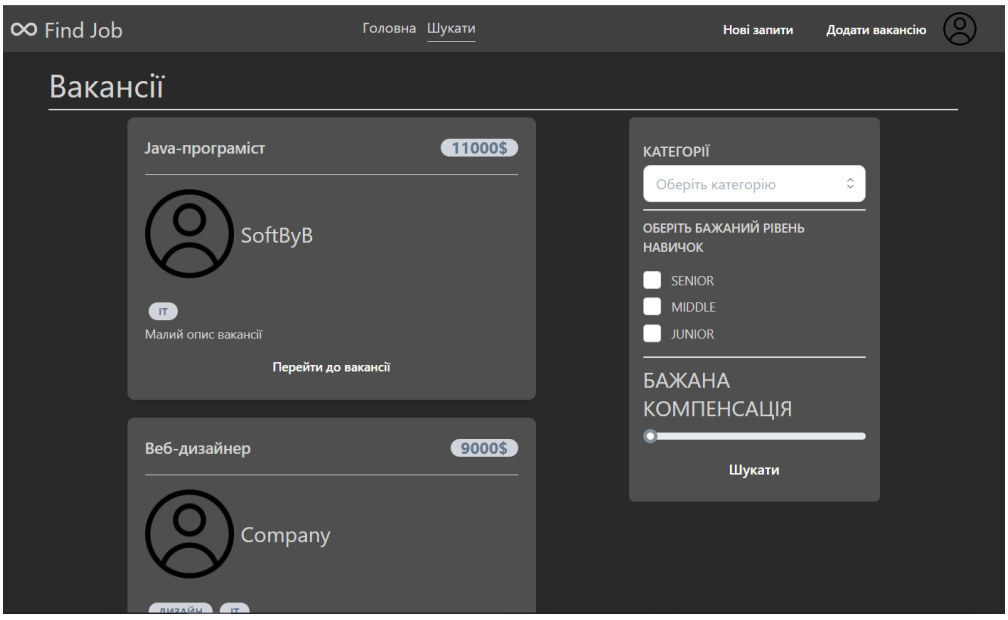

Рисунок 2.10 – Сторінка пошуку роботи

Ця сторінка містить форму фільтрації, яка відправляє запити в серверну частину застосунку. Користувач з роллю компанія має можливість створювати

вакансії. Форму створення вакансії з тестовими даними показано на рисунку 2.11.

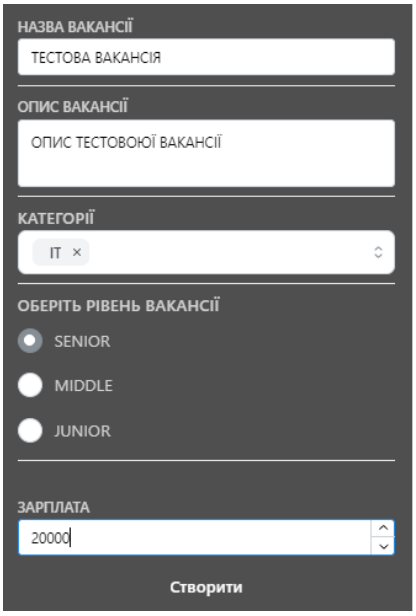

Рисунок 2.11 – Форма створення вакансії з тестовими даними

Результат виконання форми створення вакансії в веб-застосунку для пошуку роботи буде компонент з даними показані на рисунку 2.12.

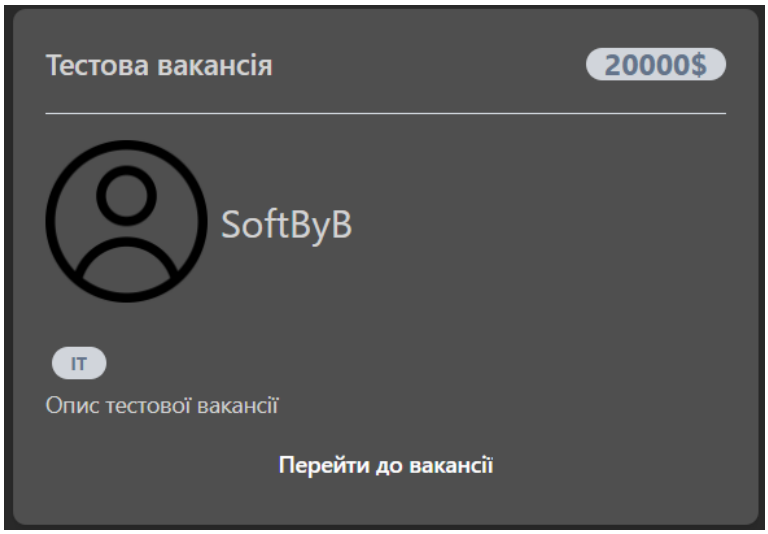

Рисунок 2.12 – Форма створення вакансії з тестовими даними

Наступний функціонал дозволить компанії-користувачу побачити, які працеємці відгукнулись на його вакансії. Результат показано на рисунку 2.13.

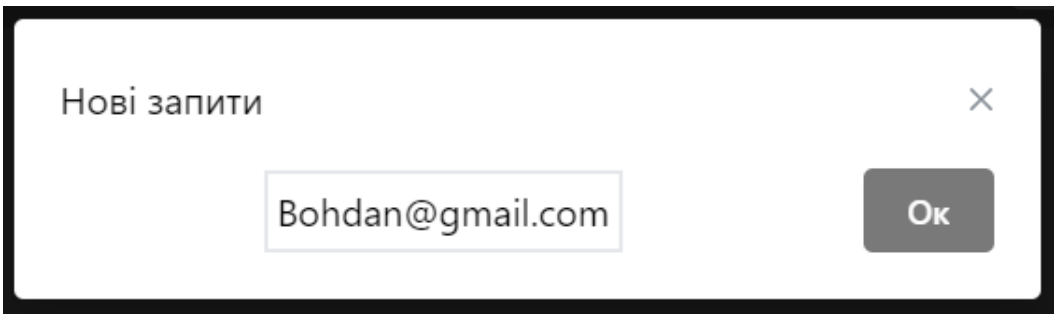

Рисунок 2.13 – Відгуки на вакансії в веб-застосунку для пошуку роботи

Наступною роллю, функціонал якої буде продемонстровано – адміністратор. В веб-застосунку для пошуку роботи адміністратор має можливість створювати категорії (див рисунок 2.14).

Рисунок 2.14 – Створення категорії

Останньою роллю в веб-застосунку для пошуку роботи є працеємець. Він має можливість відгукуватись на вакансію (див. рисунок 2.15).

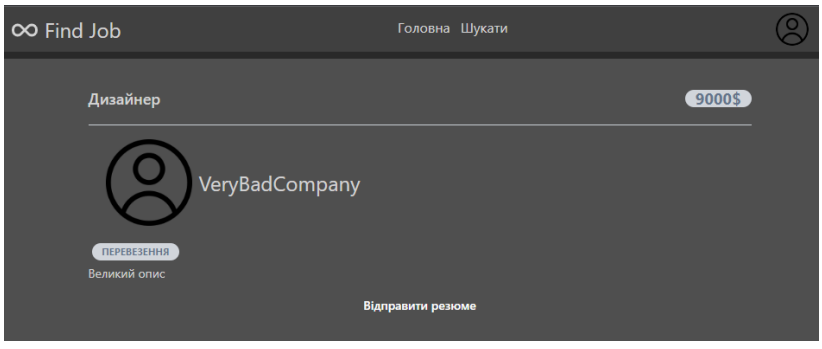

Рисунок 2.15 – Створення категорії

В разі того якщо відгук був уже надісланий веб-застосунок поверне помилку.

2.10 Висновок до другого розділу кваліфікаційної роботи

В ході виконання даної кваліфікаційної роботи було проведено детальне моделювання архітектури веб-застосунку для пошуку роботи. Була визначена структура веб-застосунку, а також структура серверної частини включаючи проектування поведінки веб-застосунку. Були розроблені моделі даних для веб-застосунку та реалізовані основні структурні елементи. Крім того, було проведено встановлення та налаштування веб-застосунку, а також здійснено його тестування.

В результаті проведених досліджень і роботи над проектом було створено функціональний та ефективний веб-застосунок для пошуку роботи. Всі поставлені завдання з моделювання архітектури, розробки, тестування та впровадження були успішно виконані. Розроблений веб-застосунок готовий до використання та задовольняє вимоги користувачів у пошуку роботи.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

3.1 Чинники, що впливають на тяжкість ураження електричним струмом.

Оскільки використання веб-застосунку для пошуку роботи відбувається за допомогою персональних комп’ютерів основною небезпекою для користувача є ураження електричним струмом. Чинники [28], що впливають на тяжкість ураження людини електричним струмом, діляться на три групи: електричного характеру, неелектричного характеру і чинники виробничого середовища. Основні чинники електричного характеру – це величина струму, що проходить крізь людину, напруга, під яку вона потрапляє, та опір її тіла, рід і частота струму. За характером дії на організм виділяють:

– відчутний струм – викликає при проходженні через організм відчутні подразнення;

– невідпускаючий струм – викликає при проходженні через організм непереборні судомні скорочення м'язів руки, в якій затиснуто провідник;

– фібриляційний струм – викликає при проходженні через організм фібриляцію серця.

Відповідно до наведеного вище:

– пороговий (найменше значення струму) відчутний струм для змінного струму частотою 50 Гц коливається в межах 0,6 – 1,5 мА і 5 – 7 мА - для постійного струму;

– пороговий невідпускаючий струм коливається в межах 10 – 15 мА для змінного струму і 50 – 80 мА - для постійного;

– пороговий фібриляційний струм знаходиться в межах 100 мА для змінного струму і 300 мА для постійного.

Граничнодопустимий струм, що проходить через людину при нормальному (неаварійному) режимі роботи електроустановки не повинен перевищувати 0,3 мА для змінного струму і 1 мА для постійного.

Електричний опір тіла людини. Шкіра є основним фактором, що визначає опір тіла людини в цілому. Опір шкіри різко знижується при ушкодженні її рогового шару, наявності вологи на її поверхні, збільшенні потовиділення, забрудненні. Опір тіла людини залежить від її статі і віку: у жінок він менший, ніж у чоловіків, у дітей менший, ніж у дорослих, у молодих людей менший, ніж у літніх. Спричиняється така залежність товщиною і ступенем огрубіння верхнього шару шкіри.

Частота і рід струму. Збільшення частоти прикладеної напруги супроводжується зменшенням повного опору тіла людини і, як наслідок, збільшенням струму через людину. Останнє дає підставу вважати, що тяжкість ураження електричним струмом має зростати зі збільшенням частоти. Але така закономірність спостерігається тільки в межах частот 0...50 Гц. Подальше збільшення частоти, незважаючи на зростання струму, що проходить через людину, не супроводжується зростанням небезпеки ураження. При частотах 450 –-500 кГц вірогідність загальних електротравм майже зникає, але зберігається небезпека опіків.

Основними чинниками неелектричного характеру є шлях струму через людину, індивідуальні особливості і стан організму людини, час, раптовість і непередбачуваність дії струму. Шлях струму через тіло людини суттєво впливає на тяжкість ураження. Особливо небезпечно, коли струм проходить через життєво важливі органи і безпосередньо на них впливає . Збільшення часу дії струму веде до зменшення опору тіла за рахунок зволоження шкіри від поту та електролітичних процесів в тканинах, поширюється пробій шкіри, підвищується вірогідність збігу максимального імпульсу струму через серце з фазою розслаблення серцевих м'язів, що призводить до більш тяжких уражень.

Чинниками виробничого середовища, які впливають на небезпеку ураження електричним струмом, є умови оточуючого середовища та схема включення людини в електричну мережу. За чинниками виробничого середовища ПУЕ [29] виділяють такі типи приміщень:

– гарячі, температура в яких впродовж доби перевищує 35°С;

– сухі, відносна вологість в яких не перевищує 60%, тобто знаходиться в межах оптимальної за гігієнічними нормативами;

– вологі, відносна вологість в яких не перевищує 75%, тобто знаходиться в межах допустимої за гігієнічними нормативами;

– сирі, відносна вологість в яких більше 75%, але менше вологості насичення;

– особливо сирі, відносна вологість в яких близька до насичення, спостерігається конденсація пари на будівельних конструкціях, обладнанні;

– запиленні, в яких пил проникає в електричні апарати та інші споживачі електроенергії, при цьому такі приміщення діляться на приміщення зі струмопровідним і неструмопровідним пилом;

– з хімічно агресивним або біологічним середовищем, що у вигляді плісняви утворюється на електрообладнанні, і призводить до порушення ізоляції.

Відповідно до ПУЕ, приміщення за небезпекою електротравм поділяються на три категорії:

– без підвищеної небезпеки;

– з підвищеною небезпекою;

– особливо небезпечні.

Категорія приміщення визначається наявністю в приміщенні чинників підвищеної або особливої небезпеки електротравм. До чинників підвищеної небезпеки належать:

– температура в приміщенні, що впродовж доби перевищує 35°С;

– відносна вологість більше 75%, але менше повного насичення

– струмопровідна підлога – металева, бетонна, цегляна, земляна тощо;

– струмопровідний пил;

– можливість одночасного доторкання людини до неструмовідних частин електроустановки і до металоконструкцій, що мають контакт із землею.

До чинників особливої небезпеки електротравм належать:

– відносна вологість близька до насичення (до 100%);

– хімічно агресивне або біологічне середовище, що пошкоджує ізоляцію.

При наявності одного з чинників підвищеної небезпеки, приміщення належить до приміщень підвищеної небезпеки електротравм. При наявності одночасно двох чинників підвищеної небезпеки або одного чинника особливої небезпеки, приміщення вважається особливо небезпечним.

3.2 Вимоги до режимів праці і відпочинку при роботі з ВДТ

При організації праці, пов'язаної з використанням ВДТ ПК для збереження здоров'я працюючих, запобігання професійним захворюванням і підтримки працездатності передбачаються внутрішньозмінні регламентовані перерви для відпочинку . Внутрішньозмінні режими праці і відпочинку містять додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення і зниження працездатності. При виконанні робіт, що належать до різних видів трудової діяльності, за основну роботу з ВДТ слід вважати таку, що займає не менше 50% робочого часу. Впродовж робочої зміни мають передбачатися:

– перерви для відпочинку і вживання їжі (обідні перерви);

– перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

За характером трудової діяльності розрізняють три професійні групи:

– розробники програм (інженери-програмісти) виконують роботу переважно з відеотерміналом та документацією при необхідності інтенсивного обміну інформацією з ПК і високою частотою прийняття рішень. Робота характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті верхніх кінцівок. Робота виконується в

режимі діалогу з ПК у вільному темпі з періодичним пошуком помилок в умовах дефіциту часу;

– оператори електронно-обчислювальних машин виконують роботу, пов'язану з обліком інформації, одержаної з ВДТ за попереднім запитом, або тієї, що надходить з нього, супроводжується перервами різної тривалості, пов'язана з виконанням іншої роботи і характеризується напруженням зору, невеликими фізичними зусиллями, нервовим напруженням середнього ступеня та виконується у вільному темпі;

– оператор комп'ютерного набору виконує одноманітні за характером роботи з документацією та клавіатурою і нечастими нетривалими переключеннями погляду на екран дисплея, з введенням даних з високою швидкістю. Робота характеризується як фізична праця з підвищеним навантаженням на кисті верхніх кінцівок на фоні загальної гіподинамії з напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням.

Правилами встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ПК при 8-годинній денній робочій зміні в залежності від характеру праці:

– для розробників програм із застосуванням ПК слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ВДТ;

– для операторів із застосуванням ПК слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за ВДТ.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години. При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-

годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин. Для зниження нервово-емоційного напруження, втомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які наведені у Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 [30].

3.3 Висновок до третього розділу

Загальний висновок з цих розділів полягає в тому, що робота з електричним струмом може бути небезпечною і потенційно шкідливою для людини. Тому важливо дотримуватись встановлених правил безпеки та рекомендацій щодо тяжкості ураження електричним струмом. Також, робота з ВДТ вимагає врахування вимог до режимів праці і відпочинку, щоб забезпечити здоров'я та благополуччя працівників. Це включає перерви для відпочинку, регулярні паузи для очей і виконання рухових вправ, які сприяють запобіганню втомі та знижують ризик розвитку професійних захворювань, пов'язаних з роботою перед ВДТ.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи освітнього рівня «Бакалавр» розроблено веб-застосунок для пошуку роботи, проведено його тестування і встановлення.. Основні особливості цього застосунку включають швидкий доступ до бази даних вакансій, моментальний пошук даних з урахуванням введених критеріїв та ефективне розмежування доступу до бази даних в залежності від користувача. Веб-застосунок може бути успішно використаний для ефективного пошуку роботи, як для звичайних користувачів, так і для роботодавців.

Для розробки веб-застосунку для пошуку роботи в першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– складені вимоги, яким повинен відповідати такий веб-застосунку.

– розглянуто рішення, що вже існують на ринку, які також пропонують послуги пошуку роботи, вивчені їх переваги та недоліки.

– проведено відбір інструментів розробки

– обґрунтовано обрані технології веб-застосунку для пошуку роботи.

В другому розділі кваліфікаційної роботи:

– проведено детальне моделювання архітектури веб-застосунку для пошуку роботи.

– визначена структура веб-застосунку, а також структура серверної частини

– розроблені моделі даних для веб-застосунку та реалізовані основні структурні елементи.

– проведено встановлення та налаштування веб-застосунку,

– здійснено його тестування.

У розділі «Безпека життєдіяльності, основи хорони праці» висвітлено чинники, що впливають на тяжкість ураження електричним струмом та подано вимоги до режимів праці і відпочинку при роботі з ВДТ

ПЕРЕЛІК ДЖЕРЕЛ

1. Singh, Ranveer, Suman Lata, and Harpreet Kaur. "Angular Js."

2. Vincent, William S. Django for Beginners: Build websites with Python and Django. WelcomeToCode, 2022.

3. Par, Leonardus, et al. "Development of a wordpress CMS-based school website as a medium of information and promotion for SMAN 1 Poco Ranaka, NTT." Community Empowerment 7.1 (2022): 88-95.

4. Goldfarb, Steven, et al. ATLAS public website: Evolution to Drupal 8. No. ATL-OREACH-PROC-2022-001. ATL-COM-OREACH-2021-010, 2022.

5. Морето, Сильвио. Bootstrap в примерах. Litres, 2022.

6. Bielak, Konrad, Bartłomiej Borek, and Małgorzata Plechawska-Wójcik. "Web application performance analysis using Angular, React and Vue. js frameworks." Journal of Computer Sciences Institute 23 (2022): 77-83.

7. Гарсія, Наталія Сеспедес, and Патрик Сеспедес Гарсія. "МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ." Молодий вчений 2 (114) (2023): 17-20.

8. Smirnov, Oleg, et al. "IntelliTC: automating type changes in IntelliJ IDEA." Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings. 2022.Strauss,

9. Dirk. "Working with Visual Studio 2022." Getting Started with Visual Studio 2022: Learning and Implementing New Features. Berkeley, CA: Apress, 2022. 65-163.

10. Farrell, Joyce. Java programming. Cengage Learning, 2022.

11. Walls, Craig. Spring in action. Simon and Schuster, 2022.

12. Wiguna, Tantra Agun, and Wahyu Eko Saputro. "Design a Digital Wedding Invitation app Using React Js & Express Js." Scientist: International Journal of Scientific Studies 1.1 (2022): 27-41..

13. Bell, Charles. "MySQL Database Service." MySQL Database Service Revealed: Running MySQL as a Service in the Oracle Cloud Infrastructure. Berkeley, CA: Apress, 2022. 137-195.).

14. Скідан, В. В., and Т. І. Демківська. "Аналіз архітектурних стилів при розробці WEB-додатків." Інформаційні технології в науці, виробництві та підприємництві (2022).

15. Lazuardy, Mochammad Fariz Syah, and Dyah Anggraini. "Modern Front End Web Architectures with React. Js and Next. Js." Research Journal of Advanced Engineering and Science 7.1 (2022): 132-141.

16. Srusti, Pranav, and Siddharth Bhorge. "Developing Complex Full Stack Java-Based Spring Cloud Applications." 2022 2nd Asian Conference on Innovation in Technology (ASIANCON). IEEE, 2022..

17. Хук useParam. reactrouter.com/en/main/hooks/use-params. Дата звернення 10.06 2023.

18. Хук useParam. ru.reactjs.org/docs/hooks effect.html .Дата звернення 10.06.2023.

19. Хук useState. https://ru.reactjs.org/docs/hooks effect.html. Дата звернення 11.06.2023.

20. Анотація @GetMapping. https://docs.spring.io/spring framework/docs/current/javadoc api/org/springframework/web/bind/annotation/GetMapping.html. Дата звернення 11.06.2023.

21. Анотація @PathVariable. https://www.baeldung.com/spring-pathvariable. Дата звернення 11.06.2023.

22. Saeed, Luqman, and Ghazy Abdallah. "Security with JWT." Pro Cloud Native Java EE Apps: DevOps with MicroProfile, Jakarta EE 10 APIs, and Kubernetes. Berkeley, CA: Apress, 2022. 293-308..

23. Wu, Huayao, et al. "Combinatorial testing of restful apis." Proceedings of the 44th International Conference on Software Engineering. 2022.

24. Конфігурування Spring boot застосунку. https://spring.io/quickstart Дата звернення 11.06.2023.

25. Spring Boot Maven Plugin/ https://docs.spring.io/spring-boot/docs/current/maven-plugin/reference/htmlsingle/ 10.06.2023

26. Команди maven. https://maven.apache.org/guides/getting-started/index.html. Дата звернення 11.06.2023.

27. ESLint. https://eslint.org/ Дата звернення 11.06.2023.

28. Гандзюк, М. П. Основи охорони праці [Текст] : підручник / М. П. Гандзюк, Є. П. Желібо, М. О. Халімовський ; за ред. М. П. Гандзюка ; МОН України. – 4-е видання. – К. : Каравела, 2008. –С. 254 – 260. – ISBN 966-8019-01-6.

29. Міністерство енергетики та вугільної промисловості України. Наказ № 476 Про затвердження Правил улаштування електроустановок. Чинний від 21.08.2017.

30. Міністерство охорони здоров'я України. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. Чинні від 10.12.1998. Міністерство охорони здоров'я України.

ДОДАТКИ

Додаток А

Jsx файл VacancyPage.jsx

```
import { useState,useEffect } from 'react';
import { useParams} from 'react-router-dom';
import {useServices} from '../services/vacancyService';
import { Card, Text, Badge, Button, Group, Divider, Center} from
"@mantine/core";
import { MainLayout } from '../layout/MainLayout';
import {LoginContext} from '../App';
import { useContext } from 'react';

export const VacancyPage = () =>{
    const [vacancy,setVacancy] =useState({});
    const { vacancyId } = useParams();
    const {getVacancyById,addTeamWorkRequest} = useServices();
    const {isAuth,role} = useContext(LoginContext);

    useEffect( () =>{
        getVacancyById(vacancyId)
            .then( (newVacancy) =>{
                console.log(newVacancy)
                setVacancy(newVacancy)
            })
    },[vacancyId]);

     const addTeamWork = () =>{
      addTeamWorkRequest({id:vacancyId}).then(result =>{
        let m;
        if(result){
          m ='Запит відправлено';
        }else{
          m = 'Ви уже відправляли запит';
        }
        alert(m);
      })
     }
    return(

      <MainLayout>
        <Center>
         <div className="w-full flex justify-center mt-2 p-8 max-
w-7xl bg-additional-color">
            <div class="container mx-auto">

              <Group position="apart" className="pb-4">
              <Text       className='text-text-color       text-lg'
weight={500}>{vacancy.name}</Text>
              <Badge  className='bg-gray-300  text-lg  text-slate-
500' >
                {vacancy.salary +"$"}
```

```
              </Badge>
            </Group>
            <Divider />
            <Card.Section  className='flex  items-center  px-5  pb-4
pt-4' component="a">
              <div className="w-[100px]  h-[100px] rounded-md">
              <img
                src={vacancy.company?.img}
                alt="Norway"
                className="w-full h-full object-cover rounded-md"
              />
              </div>

              <p  className="max-w-[140px]  text-2xl  text-text-color
pl-2 ">{vacancy.company?.name}</p>
            </Card.Section>

            {vacancy && vacancy.categories?.map( category =>{
              return(
                <Badge   key={category.id}   className='bg-gray-300
text-slate-500 m-1' >
                  {category.name}
                </Badge>
              )
            })}

            <Text size="sm" className='text-text-color'>
              {vacancy.bigDescription}
            </Text>
            {!isAuth  ? <Button  disabled> Для  подальшої  взаємодії
авторизуйтесь</Button>:null}
            {role  ==="COMPANY"  ? <Button  disabled> Для  подальшої
взаємодії змініть роль</Button>:null}
            {role  ==="ADMIN"  ?  <Button  disabled>  Для  подальшої
взаємодії змініть роль</Button>:null}
            {role ==="USER" ? <Button
            className='w-full       mt-2       hover:bg-hover-color
transition-all'
            onClick={addTeamWork}>
              Відправити резюме
            </Button>:null}
            </div>
      </div>
      </Center>
      </MainLayout>

    )
}
```

Додаток Б

Клас-контролер StartController.java

```
@RestController
@CrossOrigin("http://localhost:3000")

public class StartController {
    @Autowired
    private FilesStorageService storageService;
    @Autowired
    private AuthenticationService authenticationService;
    @Autowired
    private CompanyServ companyServ;
    @Autowired
    private UserServ userServ;
    @Autowired
    private VacancyServ vacancyServ;
    @Autowired
    private CategoryServ categoryServ;
    @Autowired
    private TeamWorkService teamWorkService;

    @GetMapping("/")
    public  List<VacDto> start(){
        return vacancyServ.getRandom();
    }

    @PostMapping("/auth/login")
    public   ResponseEntity<?>   login(@RequestBody   AuthRequest
request) {
        return
ResponseEntity.ok().body(authenticationService.login(request));
    }

    @GetMapping("/find")
    @SecurityRequirement(name = "Bearer Authentication")
    public  Map<String,Object> find(){

        Map<String,Object> hashMap = new HashMap<>();
        hashMap.put("levels", Level.values());
        hashMap.put("categories",categoryServ.getAllCategories());
        hashMap.put("minSalary",vacancyServ.getMinSalary());
        hashMap.put("maxSalary",vacancyServ.getMaxSalary());
        return hashMap;
    }

    @PostMapping("/upload")
    @SecurityRequirement(name = "Bearer Authentication")
    public ResponseEntity<String> uploadFile(@RequestParam("file")
MultipartFile file) {
        String message = "";
```

```
        try {
            storageService.save(file);

            message = "Uploaded the file successfully: " +
file.getOriginalFilename();
            return
ResponseEntity.status(HttpStatus.OK).body((message));
        } catch (Exception e) {
            message = "Could not upload the file: " +
file.getOriginalFilename() + "!";
            return
ResponseEntity.status(HttpStatus.EXPECTATION_FAILED).body((message
));
        }
    }

    @GetMapping("/files/{id}")
    @ResponseBody
    public                      ResponseEntity<InputStreamResource>
getFile(@PathVariable Integer id) throws IOException {
        Resource file = storageService.load(id);
        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION,
                        "attachment; filename=\""
                                + file.getFilename()
                                + "\"")
                .contentType(MediaType.IMAGE_PNG)
                .body(new
InputStreamResource(file.getInputStream()));
    }
    @GetMapping("/files/current")
    @SecurityRequirement(name = "Bearer Authentication")
    public                       ResponseEntity<Map<String,String>>
getCurrentUserImgPath() throws IOException {
        Map<String,String> map =new HashMap<>();
        map.put("imgPath",storageService.getCurrUserImgPath());
        return ResponseEntity.ok(map);
    }
    @PostMapping("/vacancies")
    public                               ResponseEntity<List<VacDto>>
findWithParam(@RequestBody FilterReq req){
        return
ResponseEntity.ok(vacancyServ.getFilter(req.getMinSalary(),req.get
Levels(),req.getCategories()));
    }
    @SecurityRequirement(name = "Bearer Authentication")
    @PostMapping("/addVacancy")
    public    ResponseEntity<?>  addVacancy(@RequestBody  AddVacDto
vacDto){
        return ResponseEntity.ok(vacancyServ.addVacancy(vacDto));
    }
    @GetMapping("/vacancies/{id}")
    public  VacDto getVacancyById(@PathVariable Integer id){
```

```
        return vacancyServ.getCategoryById(id);
    }

    @SecurityRequirement(name = "Bearer Authentication")
    @GetMapping("/vacancies/home")
    public List<VacDto> getVacanciesByCompany(){
        return vacancyServ.getVacanciesByCompany() ;
    }

    @SecurityRequirement(name = "Bearer Authentication")
    @PostMapping("/categories")
    public        ResponseEntity<?>        addCategory(@RequestBody
AddCategoryDto category){
        return
ResponseEntity.ok(categoryServ.addCategory(category.getName()));
    }

    @SecurityRequirement(name = "Bearer Authentication")
    @PostMapping("/teamwork")
    public        ResponseEntity<?>        addTeamWork(@RequestBody
AddTeamworkReq teamworkReq){
        return
ResponseEntity.ok(teamWorkService.addTeamWork(teamworkReq.getId())
);
    }
    @SecurityRequirement(name = "Bearer Authentication")
    @GetMapping("/teamwork/active")
    public ResponseEntity<?> getActiveTeamWork(){
        return
ResponseEntity.ok(teamWorkService.getActiveTeamWork());
    }
    @SecurityRequirement(name = "Bearer Authentication")
    @PutMapping("/teamwork/{email}")
    public  ResponseEntity<?>  updateTeamWork(@PathVariable  String
email){
        return
ResponseEntity.ok(teamWorkService.updateTeamWorkStatus(email));
    }
    @PostMapping("/registration")
    public       ResponseEntity<?>       registration(@RequestBody
RegistrationReq req){
        return
ResponseEntity.ok(authenticationService.registration(req));
    }

}
```

Додаток В

Клас-сервіс VacancyServ

```
@Service
public class VacancyServ {
    @Value("${server.name}")
    private  String SERVER_URL;
    @Autowired
    private AuthenticationService authenticationService;
    @Autowired
    private VacancyRepo vacancyRepo;
    @Autowired
    private CategoryRepo categoryRepo;
    @Autowired
    private ClientRepo clientRepo;
    @Autowired
    private CompanyRepo companyRepo;
    @Autowired
    private TeamworkRepo teamworkRepo;
    public BigDecimal getMinSalary(){
        return
vacancyRepo.findFirstByOrderBySalaryAsc().getSalary();
    }
    public BigDecimal getMaxSalary(){
        return
vacancyRepo.findFirstByOrderBySalaryDesc().getSalary();
    }
    public List<VacDto> getRandom(){
        return
vacancyRepo.findAll().subList(0,3).stream().map(this::convertToDTO
).collect(Collectors.toList());
    }
    public List<VacDto> getAll(){
        return
vacancyRepo.findAll().stream().map(this::convertToDTO).collect(Col
lectors.toList());
    }
    public  List<VacDto>  getFilter(BigDecimal  salary,  List<Level>
levels, List<Integer> categories){
        if (salary==null)salary = BigDecimal.valueOf(0);
        if (levels ==null|| levels.isEmpty()){
            levels = new ArrayList<Level>();

levels.addAll(Arrays.stream(Level.values()).collect(Collectors.toL
ist()));
        }
        List<Category> categoryList;
        if(categories==null || categories.isEmpty()){
            categoryList = (categoryRepo.findAll());;
        }
        else categoryList = categories.stream().map(id -> {
```

```
            Category c = new Category();
            c.setId(id);
            return c;
        }).collect(Collectors.toList());
        Set<Vacancy>list                                        =
vacancyRepo.findByCategoriesInAndLevelInAndSalaryGreaterThan(categ
oryList,levels,BigDecimal.valueOf(salary.intValue()-1));
        return
list.stream().map(this::convertToDTO).collect(Collectors.toList())
;
    }
    public VacDto getCategoryById(Integer id){
        Vacancy vacancy = vacancyRepo.getById(id);
        VacDto dto = new VacDto();
        dto.setId(vacancy.getId());
        dto.setName(vacancy.getName());
        dto.setBigDescription(vacancy.getBigDescription());
        dto.setSalary(vacancy.getSalary());
        dto.setCategories(vacancy.getCategories());
        dto.getCompany().put("id",vacancy.getCompany().getId());
      dto.getCompany().put("name",vacancy.getCompany().getName());
        dto.getCompany().put("img",SERVER_URL+"files/"+
vacancy.getCompany().getId());

        return dto;
    }

    public void save(Vacancy vacancy){
        vacancyRepo.save(vacancy);
    }
    public List<VacDto> getVacanciesByCompany(){
        Account                        account                  =
authenticationService.getCurrentAccount();
        List<Vacancy> vacancies = new ArrayList<>();
        List<Teamwork> list;
        switch (account.getRole()){
            case COMPANY:
                list =teamworkRepo.findByCompany(account.getId());

vacancies.addAll(list.stream().map(Teamwork::getVacancy).collect(C
ollectors.toList()));
                break;
            case USER:
                list =teamworkRepo.findByClient(account.getId());

vacancies.addAll(list.stream().map(Teamwork::getVacancy).collect(C
ollectors.toList()));
                break;
        }

        return
vacancies.stream().map(this::convertToDTO).collect(Collectors.toLi
st());
```

```
    }
    public boolean addVacancy(AddVacDto vacDto){
        Vacancy vacancy = new Vacancy();
        Company                        company                        =
companyRepo.getById(authenticationService.getCurrentAccount().getI
d());
        Set<Category>categories =
                vacDto.getCategories()
                        .stream().map(                          id->
categoryRepo.getById(id)).collect(Collectors.toSet());
        vacancy.setName(vacDto.getName());
        vacancy.setSmallDescription(vacDto.getSmallDescription());
        vacancy.setSalary(vacDto.getSalary());
        vacancy.setCategories(categories);
        vacancy.setCompany(company);
        vacancy.setLevel(vacDto.getLevel());
        vacancyRepo.saveAndFlush(vacancy);
        List<Category> categories1 = new ArrayList<>(categories);
        for (int i = 0; i < categories1.size(); i++) {
            Set<Vacancy>                vacancies                =
categories1.get(i).getVacancies();
            vacancies.add(vacancy);
            categories1.get(i).setVacancies(new
HashSet<Vacancy>(vacancies) {
            });
            categoryRepo.saveAndFlush(categories1.get(i));
        }

        return true;
    }
    private VacDto convertToDTO(Vacancy vacancy){
        VacDto dto = new VacDto();
        dto.setId(vacancy.getId());
        dto.setName(vacancy.getName());
        dto.setSmallDescription(vacancy.getSmallDescription());
        dto.setSalary(vacancy.getSalary());
        dto.setCategories(vacancy.getCategories());
        dto.getCompany().put("id",vacancy.getCompany().getId());

dto.getCompany().put("name",vacancy.getCompany().getName());
        dto.getCompany().put("img",SERVER_URL+"files/"+
vacancy.getCompany().getId());

        return dto;
    }
}
```

Додаток Г

Клас-репозиторій VacancyRepo.java

```
@Repository
public interface VacancyRepo extends
JpaRepository<Vacancy,Integer> {
    public
Set<Vacancy>findByCategoriesInAndLevelInAndSalaryGreaterThan(
            @Param("categories")List<Category>categories,
            @Param("levels")List<Level> levels,
            @Param("Salary")BigDecimal Salary
    );

    public Vacancy findFirstByOrderBySalaryAsc();

    public Vacancy findFirstByOrderBySalaryDesc();

    @Query("SELECT v FROM Vacancy v WHERE v.company.id = :id")
    public List<Vacancy> findByCompany(@Param("id") Integer id);


}
```