МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

«Веб-застосунок для пошуку робочих місць з IT спеціальностей»

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Прикладне програмування»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи ПП-41
_____ Нікітенко О.О._____
(прізвище та ініціали)
Керівник _____Ващіліна О.В._____
(прізвище та ініціали)
_____ к.ф.-м.н., доц._____
(науковий ступінь, звання)

Унікальність тексту 93%

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *прикладних інформаційних систем*
Протокол № від р.
зав. кафедри _____ Плескач В.Л.

Київ – 2023

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вваліфікаційної роботи бакалавра	14.10.2022	виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	24.10.2022	заява
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	31.10.2022	виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	01.11.2022	виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	08.11.2022	виконано
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	21.12.2022	виконано
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	31.01.2023	виконано
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	30.03.2023	виконано
9.	Подання роботи у першому варіанті	28.04.2023	виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2023	виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**22.05.2023**	виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	26.05.2023	виконано
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	12.06.2023	виконано
14.	Захист кваліфікаційної роботи бакалавраибір теми та наукового керівника к	26.06.2023	

Здобувач вищої освіти _____

(підпис)

Керівник _____

(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	3
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	16
2	21
3	16
Висновки	2
Перелік використаних джерел	8
Додатки	4

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Нікітенко О.О.			Відомість дипломної роботи	Лист	Листів
Керівн.	Ващіліна О.В.					80
Н/контр.	Макаренко С.А.		26.05.2023			
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

Дипломна робота: 80 с., 7 рис., 4 табл., 66 джерел, 3 дод.

Ця дипломна робота присвячена проектуванню та розробленню веб-застосунку для пошуку робочих місць з ІТ спеціальностей.

Метою дипломної роботи є ефективна організація пошуку робочих місць з ІТ спеціальностей у формі веб-застосунку.

Для досягнення поставленої мети треба вирішити такі **завдання**:

- дослідити функціональні можливості та особливості побудови веб-застосунків для пошуку робочих місць з ІТ спеціальностей;
- здійснити аналіз програмно-технологічних рішень для створення веб-застосунків з пошуку вакансій;
- на основі проведених досліджень здійснити проектування і розробку веб-застосунку для пошуку робочих місць з ІТ спеціальностей.

Об’єкт дослідження.

Процеси роботи веб-застосунків з пошуку вакансій.

Предмет дослідження.

Процеси працевлаштування ІТ-фахівців за допомогою інформаційних технологій.

Методи дослідження.

Аналіз літературних джерел для вивчення теоретичних аспектів програмних систем з пошуку вакансій, включаючи методи проектування та розробки систем пошуку вакансій, емпіричний аналіз та синтез систем пошуку вакансій, метод порівняння, що застосовано для аналізу наявних ресурсів та програмних систем пошуку вакансій, зокрема, порівняння з існуючими веб-застосунками пошуку робочих місць з ІТ спеціальностей, для визначення переваги та недоліків розроблюваної системи.

Ключові слова: веб-застосунок, інтернет, ІТ, пошук роботи, вакансії.

ANNOTATION

Thesis: 80 p., 7 figures, 4 tables, 66 sources, 3 appendix.

This thesis is devoted to the design and development of a web application for finding jobs in IT specialties.

The purpose of the thesis is to effectively organize the search for jobs in IT specialties in the form of a web application.

To achieve this goal, the following **tasks** need to be solved:

- to study the functionality and features of building web applications for job search in IT specialties;

- to analyze software and technological solutions for creating web applications for job search;

- based on the research, to design and develop a web application for job search in IT specialties;

Object of research.

Processes of web-based job search applications.

The subject of research.

Programmatic, technical, organizational foundations, principles, approaches to building a web application for finding jobs in IT specialties.

Research methods.

The study of literature sources on the theoretical aspects of job search software systems will help in the development of a web application for finding jobs in IT specialties. The study will take into account the methods of designing and developing job search systems, conduct an empirical analysis and synthesis of such systems, as well as a comparison with existing web applications for finding jobs in IT specialties to determine the advantages and disadvantages of the system under development.

Keywords: web application, Internet, IT, job search, vacancies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

IDE (Integrated Development Environment) - Інтегроване середовище розробки

CRUD (Create Read Update Delete) - 4 основні функції управління даними «створення, читання, оновлення і вилучення»

JPA (Java Persistence API) - стандартизований інтерфейс для Java ORM фреймворків

ORM (Object-Relational Mapping) - Об'єктно-реляційна проекція

JPQL (Java Persistence Query Language) - платформо-незалежна об'єктно-орієнтована мова запитів

JDBC (Java DataBase Connectivity) - з'єднання з базами даних на Java

ВСТУП

В останні роки, з розвитком нових технологій, все більшою популярністю користується веб-застосунок для пошуку робочих місць з ІТ спеціальностями. Цей застосунок допомагає людям, які шукають роботу в галузі інформаційних технологій, знайти відповідні пропозиції працевлаштування. Користувачі можуть шукати вакансії на веб-застосунках роботодавців безпосередньо або через альтернативні сервіси. У залежності від бажаної спеціальності та рівня кваліфікації, користувачі можуть знайти відповідну роботу у великих компаніях, маленьких стартапах або навіть працювати віддалено.

Актуальність цієї теми зумовлена тим, що сучасний ринок праці відкриває безліч можливостей для спеціалістів у сфері ІТ. Однак, знайти відповідну вакансію може бути складно, особливо для початківців. Веб-застосунки для пошуку робочих місць у сфері ІТ допомагають спеціалістам знайти відповідну роботу швидко та ефективно, не обмежуючи їх географічно. Багато компаній, які шукають фахівців у сфері ІТ, публікують вакансії в інтернеті, що дозволяє кандидатам бути в курсі всіх актуальних пропозицій та подавати свої резюме в будь-який зручний для них час.

Метою кваліфікаційної роботи бакалавра є ефективна організація пошуку робочих місць з ІТ спеціальностей у формі веб-застосунку.

Завдання дослідження:

- дослідити функціональні можливості та особливості побудови веб-застосунків для пошуку робочих місць з ІТ спеціальностей;
- здійснити аналіз програмно-технологічних рішень для створення веб-застосунків з пошуку вакансій;
- на основі проведених досліджень здійснити проектування і розробку веб-застосунку для пошуку робочих місць з ІТ спеціальностей;

Об’єктом дослідження кваліфікаційної роботи бакалавра є процеси роботи веб-застосунків з пошуку вакансій.

Предметом дослідження засоби побудови веб-застосунку для пошуку робочих місць з IT спеціальностей.

Методи дослідження: дослідження літературних джерел щодо теоретичних аспектів програмних систем з пошуку вакансій допоможе в розробці веб-застосунку для пошуку робочих місць з IT спеціальностей. У дослідженні буде враховано методи проектування та розробки систем з пошуку вакансій, проведено емпіричний аналіз та синтез таких систем, а також порівняння з існуючими веб-застосунками для пошуку робочих місць з IT спеціальностей, щоб визначити переваги та недоліки розроблюваної системи.

Практичне значення одержаних результатів розробки веб-застосунку для пошуку робочих місць з IT спеціальностей полягає в тому, що він може стати корисним та зручним інструментом для людей, які шукають роботу в сфері IT. Розроблений веб-застосунок може допомогти швидко та ефективно знайти робоче місце, взаємодіяти з роботодавцями та підвищувати шанси на успішне працевлаштування. Крім того, веб-застосунок може допомогти роботодавцям швидко та зручно знаходити кваліфікованих працівників в сфері IT, що сприятиме розвитку цієї галузі.

Структура роботи:

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, розподілених на підрозділи, та висновку.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПОШУКУ РОБОЧИХ МІСЦЬ З ІТ СПЕЦІАЛЬНОСТЕЙ ТА ПРОГРАМНИХ СИСТЕМ З ПОШУКУ ВАКАНСІЙ

1.1. Основні поняття з пошуку роботи

1.1.1. Процес пошуку роботи

Процес пошуку роботи складається з декількох етапів, і необхідно почати його зі збору інформації. Після того, як кандидат знайде цікаву роботу, йому необхідно отримати якомога більше інформації про роботодавця і все, що пов'язано з відкритою вакансією. Коли він отримує інформацію заздалегідь, буде легше подати заявку на цю посаду.

Коли людина знайшла цікаву вакансію, їй необхідно уважно прочитати оголошення про роботу. Бажано знайти відповіді принаймні на наступні питання, пов'язані з відкритою вакансією:

- Які основні обов'язки та особливі вимоги на цій посаді?
- Чи потрібна певна освіта або певний досвід роботи?
- Чи відповідають завдання компетенції кандидата?
- Що можна дізнатися з вимог вакансії?
- Яку людину шукають на цю посаду?

Перед тим, як подавати заяву про працевлаштування, також потрібно ознайомитися з роботодавцем. Відвідати веб-застосунок організації та дізнатися, принаймні, про галузь, розмір і структуру організації, а також її бачення, цінності та цілі.

Після ознайомлення з вакансією та роботодавцем, якщо кандидат все ще хоче подати заявку на цю посаду, слід подумати над наступним кроком. Чи зв'яжеться кандидат з роботодавцем перед тим, як відправити заявку на роботу? Чи допоможе дзвінок або візит на робоче місце, або відправлення електронного листа написати заяву? Іноді в оголошенні про вакансію вказуються контактні дані особи, яка бере участь у процесі підбору

персоналу, а також час, коли можна звернутися до роботодавця. Якщо дійсно є питання про вакансію і кандидат вважає, що буде корисно зв'язатися з роботодавцем, можна зробити це до того, як подати заяву про прийом на роботу. Однак слід ретельно обміркувати, який варіант є найбільш прийнятним у конкретній ситуації, і діяти відповідно до нього.

1.1.2. Документи для подачі заявки на працевлаштування

Варто витратити трохи часу і зусиль на написання листа-заявки, оскільки те, що кандидат напише, може допомогти йому пройти співбесіду. Мета листа-заявки - привернути увагу рекрутера, щоб саме цього кандидата запросили на співбесіду.

У листі-заявці потрібно описати:

- Чому кандидат претендує на цю роботу;
- Як кандидат відповідає критеріям відбору на цю посаду;
- Чому саме цей кандидат повинен бути обраний на дану роботу.

Лист-заявка повинен створити позитивне враження про кандидата. Він має бути стислим і адресованим саме роботодавцю, на посаду якого людина претендує. У резюме потрібно описати свій досвід та історію роботи, але в листі-заявці основна увага приділяється майбутньому. Необхідно уникати складання списків. У листі-заявці можна описати себе і свої компетенції в більш неформальній формі. На додаток до предметної компетенції, також потрібно підкреслити інші свої сильні сторони, які можуть бути корисними для посади, на яку претендує кандидат. Окрім їх опису, в листі-заявці важливо пояснити, що кандидат може запропонувати організації.

Якщо кандидат претендує на посаду, на якій у нього немає попереднього досвіду роботи, йому необхідно пояснити, як його компетенція відповідає цій посаді, і підкреслити свою мотивацію, гарне ставлення до роботи та здатність до навчання.

Працюючи над резюме, потрібно зробити його вичерпним, але водночас чітким. Можна проявити творчість при складанні резюме, але не варто занадто відходити від загальноприйнятої структури. Потрібно

висловлюватися лаконічно. Зрештою, дуже важливо, щоб рекрутер легко і швидко знаходив потрібну інформацію, оскільки на кожне окреме резюме не завжди виділяється багато часу. Хороше резюме є візуально цікавим, чітким і легким для сприйняття. Воно швидко показує, чи відповідає кандидат вимогам вакансії.

Бажано описувати досвід роботи та інформацію про освіту в хронологічному порядку, починаючи з найновішого досвіду. Коротко описувати кожне місце роботи. Також можна описати, що входить у обов'язки і чого людина навчилася в процесі роботи. Окрім досвіду роботи, також можна описати свої мовні та IT-навички, а також надати список рекомендацій. Потрібно переконатися, що резюме містить контактну інформацію. Профілі LinkedIn не замінили традиційні резюме, але сервіс може підтримати резюме в процесі пошуку роботи.

Кандидат також може написати лист-заявку в полі електронного повідомлення. Подумати над текстом у своєму електронному повідомленні так само уважно, як якщо б він писав традиційний лист-заявку.

- У полі "Тема" написати "Заявка" і назву посади, на яку він претендує.
- Завантажити своє резюме як додаток до повідомлення.
- Почати з дружнього привітання.
- У текстовому полі написати текст листа-заявки. Не обов'язково дотримуватися макету традиційного листа-заявки.
- Пояснити, що резюме прикріплене до електронного листа.
- Закінчити лист заключною відповіддю і вказати свою контактну інформацію.

Власні електронні сервіси пошуку роботи роботодавців мають свої особливості. Наприклад, рекрутери можуть здійснювати на них пошук за словами. Іншими словами, потрібно знайти в оголошенні про вакансію ключові слова, що описують характер завдання і характеристики претендента, і використовувати їх у власній заявці. Подаючи заявку в онлайн-сервісі роботодавця, потрібно уважно прочитати інструкцію. Бажано

спочатку написати текст за допомогою програми для редагування тексту, а потім скопіювати його у форму.

Дедалі частіше роботодавці вимагають від претендентів відеорезюме. Відеоматеріал полегшує роботодавцю попередній відбір кандидатів. За допомогою відео можна створити більш достовірне уявлення про себе, ніж у традиційній заявці на роботу. Можна зробити коротке, 1-3-хвилинне відео і завантажити його на відеосервіс, надіслати посилання роботодавцю.

Портфоліо - це колекція найкращих і найважливіших робіт або досягнень кандидата. Можна зібрати портфоліо по-різному. Це може бути папка, портфоліо, демонстрація, план, малюнок або колекція фотографій. Найчастіше портфоліо використовують у креативному секторі, але воно добре працює і в багатьох інших сферах. Наприклад, портфоліо шеф-кухаря може включати його особисті рецепти, фотографії страв та відгуки клієнтів. Для кожної нової заявки слід збирати нове портфоліо. Однак не потрібно робити портфоліо занадто широким за обсягом. Портфоліо може включати:

- сертифікати, рекомендації та оцінки
- зразки різних робочих завдань у вигляді брошур, плакатів, програм, журнальних статей тощо
- все, що допоможе отримати роботу.

Можна надіслати своє портфоліо роботодавцю або взяти його з собою на співбесіду і представити там. Якщо портфоліо можна знайти в Інтернеті, необхідно вказати посилання на нього у своєму резюме.

1.1.3. Співбесіда на роботу

Для роботодавця співбесіда - це спосіб перевірити відповідність претендента відкритій вакансії та робочому колективу. Претендент також знайомиться з роботодавцем і обмірковує свою відповідність посаді та зацікавленість у тому, щоб стати частиною організації. Коли кандидат буде надсилати виклик інтерв'юеру і задавати питання про роботодавця або посаду, він буде створювати враження, що він мотивований і покаже, що він

дійсно зацікавлен в цій посаді. Водночас кандидат отримує інформацію про те, чи підходить йому ця посада.

Перед початком співбесіди слід зробити наступне:

- Дізнатися більше про роботодавця.
- Згадати посадові обов'язки і те, що вказано в оголошенні про вакансію.
- Знати свої компетенції та бути готовим коротко описати їх.
- Подумати над питаннями, які цікавлять про роботу і потенційного роботодавця.

Позитивне перше враження має велике значення при першій зустрічі.

Під час співбесіди рекрутер оцінить, чи справді кандидат зацікавлений в цій посаді і чи підходять його навички та компетенції для неї. Перш за все, співбесіда виявить навички взаємодії та ставлення до роботи. Якщо на співбесіді присутні кілька інтерв'юерів, необхідно приділити кожному з них однакову кількість уваги. Зазвичай співбесіда проходить у три етапи:

- На початку часто обговорюються загальні питання. Мета - створити загальне уявлення про співбесідника.
- В середині співбесіди інтерв'юери ставлять запитання, щоб з'ясувати, наскільки кандидат мотивован і чи підходить на цю посаду. Питання також стосуватимуться кар'єри та змін у ній. Крім того, інтерв'юери захочуть дізнатися, що кандидат за людина, які його цінності та світогляд.
- На завершальному етапі співбесіди основна увага приділяється більш практичним питанням, пов'язаним з посадою, таким як заробітна плата, робочий час і дата початку роботи. Інтерв'юер також часто описує, як буде продовжуватися процес розгляду заявки. За необхідності також можна задати питання про наступні етапи процесу.

Якщо кандидата не обрали на посаду, варто запитати роботодавця або інтерв'юера, на які фактори зверталася увага при відборі, і які були причини, чому кандидата не обрали на цю роботу.

1.2. Поняття веб-застосунку для пошуку роботи

Веб-застосунок для пошуку роботи це платформа, на якій роботодавці розміщують свої вакансії та шукають відповідних кандидатів. З іншого боку, кандидати розміщують свої резюме та шукають бажані вакансії. Крім того, веб-застосунок з працевлаштування - це всесвітня пошукова система. Вона об'єднує вакансії з багатьох інших служб зайнятості та розміщує їх на платформі.

Для того, щоб краще зрозуміти, потрібно знати про переваги такого веб- застосунку для працівників та роботодавців:

- Онлайн-резюме.

Це дуже зручно для роботодавців, оскільки їм не потрібно сортувати тони резюме вручну. Крім того, є можливість встановлювати різні фільтри для кожної опублікованої вакансії, щоб було зручніше керувати надісланими резюме. Наприклад, більшість відомих веб-застосунків з пошуку роботи, такі як Glassdoor або Ladders, надають роботодавцям ATS. Це система відстеження кандидатів. Функціонує як основний фільтр. Вона автоматично видаляє всі невідповідні резюме, враховуючи ключові слова та загальний формат резюме кандидата. Це дозволяє компаніям не сортувати тони нерелевантних резюме. Таким чином, компанії економлять час і роблять процес підбору персоналу більш ефективним.

- Дошки оголошень про роботу.

Компанії користуються перевагами платних і безкоштовних дошок оголошень. Таким чином вони розміщують оголошення про свої вакансії та орієнтуються на кандидатів з певними навичками та досвідом роботи в певній сфері.

- Дослідження роботодавців.

Дозволяє шукачам побачити профіль компанії, прочитати відгуки та переглянути рейтинги інших претендентів.

Враховуючи обсяг ринку веб-застосунків з працевлаштування, він дійсно величезний. Окрім Indeed, існують такі гіганти у сфері працевлаштування, як Glassdoor, Monster, JobisJob, Simplyhired, ZipRecruiter,

Lensa, CareerBuilder та Jooble. Однак, за даними Statista, Indeed є одним з найбільш відвідуваних веб-застосунків для пошуку роботи в усьому світі.

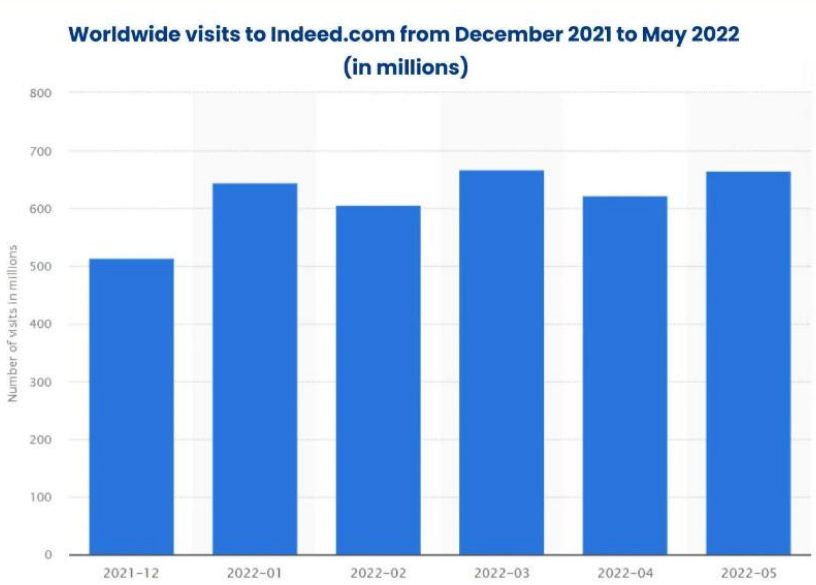

Рисунок 1.1 – Графічне представлення кількості відвідувань веб-застосунка Indeed за останні роки

Зокрема, Indeed увійшла до списку найбільш інноваційних компаній світу за версією Fast Company 2022. У цьому списку представлені підприємства, які мають значний вплив на власний бізнес та суспільство.

1.3. Критерії унікальності веб-застосунків пошуку роботи.

Як зазначив Давіде Новеллі, директор з міжнародних онлайн-операцій, є три основні ключі до успіху платформи для пошуку роботи:

- Нарощування досвіду роботи на місцевому ринку. Це означає, що компанія, яка займається пошуком роботи, наймає тільки тих фахівців, які знають специфіку кожного регіонального ринку.
- Забезпечення релевантних результатів завдяки контекстному перекладу. Платформі потрібно забезпечувати триступеневий контекстний переклад, щоб охопити всі культурні та місцеві потреби користувачів.

- Розробка досвіду пошуку роботи для кожної локації. Кандидати мають можливість шукати будь-яку посаду та локацію з будь-якого куточку світу. Система пошуку не прив'язана до місцезнаходження, що значно покращує користувацький досвід пошукачів.

1.4. Основні виклики при створенні веб-застосунки для пошуку роботи

Після того, як розробник вирішив створити платформу для пошуку роботи, йому потрібно зрозуміти основні виклики, які можуть виникнути. Можна зіткнутися з ними до або після запуску онлайн-бізнесу з пошуку роботи. Розуміння цих проблем допоможе їх уникнути.

1.4.1. Невідповідність кандидатів

Претендент може виглядати як хороший кандидат для компанії. Але потім виявляється, що він занадто кваліфікований або немає необхідного набору професійних здібностей. З іншого боку, вакансія може виявитися не релевантною для кандидата навіть після проходження співбесіди.

Насамперед, це пов'язано з алгоритмами, які підбирають результати на основі SEO (пошукової оптимізації). Наприклад, ці алгоритми можуть бути застарілими. Якщо кандидат використовує неправильні або недостатню кількість "правильних ключових слів", може виникнути проблема невідповідності. Одним з хороших рішень проблеми невідповідності є використання технологій штучного інтелекту. Наприклад, такий гігант, як IBM, впроваджує програмне забезпечення на основі ШІ, яке підбирає резюме кандидатів. Принцип роботи досить простий і потужний. Алгоритм відстежує як резюме кандидатів, так і їхні профілі в соціальних мережах, щоб зібрати більше даних про кандидата. Після цього програма зіставляє навички претендента з його вміннями, щоб запропонувати найбільш релевантну вакансію. Однак впровадження програмного забезпечення на основі штучного інтелекту є складним і досить дорогим процесом. У багатьох

випадках стартапери можуть обійтися без цього або використовувати альтернативні рішення.

1.4.2. Достовірність оголошень про вакансії

Багато претендентів, які використовують платформи з працевлаштування, скаржаться на застарілі оголошення. Деякі веб-застосунки з пошуку роботи нерегулярно оновлюють базу роботодавців/працівників. Це призводить до появи застарілих вакансій або вже найнятих кандидатів. Не актуальні публікації з'являються, коли модерація веб-застосунку не працює на достатньому рівні. Щоб вирішити цю проблему, необхідно подумати про те, як можна відстежувати, чи є публікації якісними та актуальними. Хороша ідея - автоматизувати процес моніторингу. Щоб програма контролювала дату публікації вакансії, її статус, а також користувача/компанію, яка опублікувала вакансію. Ще одне гарне рішення - запровадити на платформі плату за розміщення вакансій. Роботодавці більше не будуть зацікавлені в тому, щоб тримати вакансію застарілою або неактуальною.

1.4.3. Вразливість даних користувача

В умовах віртуального спілкування дуже важко довести достовірність наданих даних. Як пошукачі, так і роботодавці не можуть повністю довіряти інформації до особистої співбесіди. До речі, цей факт призводить до того, що щодня з'являється велика кількість шахраїв, які збирають персональні дані або обманюють користувачів. Відвідувачі веб-застосунку з працевлаштування часто страждають від спамерів, які перехоплюють їхню електронну пошту або надсилають тони повідомлень на мобільні телефони. Тому власнику платформи варто подумати про захист даних своїх користувачів. Підвищена безпека підвищить довіру до веб-застосунку з працевлаштування.

Існують правові норми, які захищають приватні дані, це робить GDPR. Це Загальний регламент захисту даних в Європейському Союзі. Щоб підвищити безпеку платформи пошуку роботи варто слідувати наступним правилам:

- Не відображати контактну інформацію користувачів. Можна реалізувати внутрішньо платформний чат, щоб дати користувачам можливість спілкуватися.
- Запобігати копіюванню інформації. Інтегрувати анти скрепери, які не дозволяють іншим пошуковим системам скопіювати дані з веб-застосунку з пошуку роботи.
- Повідомляти користувачів про всі правила та політику. Попросити як роботодавців, так і пошукачів уважно прочитати політичні документи і підтвердити, що вони ознайомилися з матеріалом.

1.5. Кроки створення веб-застосунку для пошуку роботи

Нижче наведені кроки з інформацією, як створити веб-застосунок для пошуку роботи, який буде корисним як для компаній, так і для пошукачів:

- Проведення дослідження ринку та підбір найбільш підходящої бізнес-модель.
- Підбір необхідного функціоналу.
- Додавання кількох функцій, які будуть відрізняти платформу від конкурентів.
- Створення MVP агрегатора веб-застосунків з пошуку роботи.
- Розробка повноцінного веб-застосунку для пошуку роботи.
- Підбір правильної команди розробників.
- Визначення вартості запуску веб-застосунку з пошуку роботи.

1.5.1. Проведення маркетингового дослідження ринку та вибір бізнес-моделі

Без дослідження ринку навіть потенційно блискуча бізнес-ідея приречена на провал. Можна вкласти багато грошей, а потім виявити, що продукт чи послуга насправді нікому не потрібні.

Дослідження ринку допоможе уникнути такої невдачі. Людина отримує важливі дані про цільовий ринок, потенційних клієнтів і конкурентів. Проаналізувавши цю інформацію, можна створити щось більше, ніж клон Indeed або GlassDoor. Розробник дізнається, як створити веб-застосунок з пошуку роботи, який принесе виняткову цінність клієнтам. Основні типи веб-застосунків з пошуку роботи наведені в таблиці нижче.

Таблиця 1.1 – Типи веб-застосунків з пошуку роботи

Класифікація	Типи веб-застосунків для пошуку роботи	Приклади
За місцем розташування	Місцеві, міжнародні	WNYJobs (перший регіональний веб-застосунок з працевлаштування в Західному Нью-Йорку), Totaljobs (Великобританія), JobisJob, Indeed
За рівнем кваліфікації	Навички вищого рівня, будь-які	Ladders, GlassDoor
За спеціалізацією	Загальні, спеціалізовані	Jooble, CareerBuilder, Health eCareers (healthcare), K12JobSpot

		(education)

Вибір найбільш підходящої бізнес-моделі стає першочерговим завданням для створення веб-застосунків з пошуку роботи, який буде приносити стабільний прибуток, вирішіть, як брати плату з користувачів за доступ до платформи і для цього існує декілька способів:

- Розміщення вакансій. Ця модель заробітку передбачає оплату з роботодавців за публікацію їхніх вакансій на платформі. Наприклад, Dice бере 495 доларів за одне оголошення, яке буде відображатися протягом 30 днів.
- Оплата за клік. Можна дозволити компаніям розміщувати свої вакансії безкоштовно і брати плату, коли потенційний працівник виявляє певний інтерес до вакансії.
- Підписка. Сплачуючи щомісячну оплату, роботодавці отримують можливість публікувати необмежену кількість вакансій протягом місяця.
- Різні плани та пакети. Можна запропонувати користувачам різні пакети, які відрізняються за функціональністю. Пропонуючи кілька варіантів, розробник надає користувачам гнучкість у виборі рішення, яке найбільше відповідає їхньому бюджету та бізнес-цілям.
- Спонсорована вакансія. Якщо роботодавцю потрібен кращий показ його вакансії, можна виділити його пропозицію за додаткову плату. Крім того, можна брати плату за просування вакансії в соціальних мережах або на веб-застосунках партнерів.
- Плата за кар'єрні послуги. Можна запропонувати додаткові кар'єрні послуги за додаткову плату. Вони можуть включати написання резюме або надання корисних порад щодо успішного проходження співбесіди.
- Плата за доступ до резюме. Можна брати з роботодавців плату за доступ до резюме пошукачів. Ця бізнес-модель може бути дуже

прибутковою за умови, що на веб-застосунку є велика кількість претендентів.

1.5.2. Вибір обов'язкових функцій для веб-застосунку з пошуку роботи

Отже, після визначення, як монетизувати веб-застосунок з пошуку роботи. Наступний крок - вирішити, який функціонал буде запропонований користувачам. Традиційно веб-застосунки з пошуку роботи орієнтовані як на роботодавців, так і на пошукачів. Тому варто обирати функції, які допоможуть зрозуміти, як створити веб-застосунок з пошуку роботи, що задовольняє потреби обох сторін. Перераховані нижче функції підходять для створення як повноцінного веб-застосунку з пошуку роботи, так і його MVP-версії.

- Особистий профіль. Можна дозволити кандидатам створювати особистий кабінет на веб-застосунку пошуку роботи. Потрібно переконатися, що вони можуть заповнити свою основну інформацію. Сюди входять особисті дані, освіта, навички, досвід роботи та мови. Також можна надати пошукачам можливість завантажити або створити своє резюме.
- Пошук вакансій та фільтри. Широка система фільтрів допоможе кандидатам точніше сформулювати свій запит на роботу.
- Профіль роботодавця. Тут шукачі роботи можуть ознайомитися з основною інформацією про компанію. Також вони можуть переглянути список вакансій, подивитися фотографії та перевірити посилання на соціальні мережі. Але найголовніше - це можливість прочитати чесні відгуки від співробітників та інших претендентів.
- Конструктор резюме. Ця функція буде корисною як для кандидатів, які шукають свою першу роботу, так і для досвідчених професіоналів, які хочуть заощадити свій час. Тут їм буде доступна анкета, в яку вони зможуть внести основну інформацію про свою освіту, досвід роботи та навички, щоб створити професійне резюме.

- Можливість подати заявку на роботу. Коли претендент нарешті вибрав конкретну компанію або роботодавця, цей функціонал дозволяє прикріпити резюме або супровідний лист і відправити їх потенційному роботодавцю.
- Функціонал для розміщення вакансій. Компанії повинні мати можливість створити вакансію за лічені секунди. Для цього необхідно надати їм форму, де роботодавці можуть вказати ключові деталі, пов'язані з пропозицією про роботу. До них відносяться назва посади, вимоги, обов'язки та умови праці.
- Профіль компанії. Цей функціонал дозволяє роботодавцям вказати основну інформацію про свою компанію. Серед ключових даних - галузь, місцезнаходження, розмір компанії. Тут організації також можуть запропонувати посилання на свій веб-застосунок або акаунти в соціальних мережах.
- Аналітична панель. Цей функціонал дозволяє рекрутерам або менеджерам з персоналу переглядати статистику, пов'язану з пропозицією про роботу, наприклад, перегляди, кліки або заявки. Крім того, роботодавці можуть робити нотатки про кожного кандидата і встановлювати певний статус (відхилений, пройшов відбір або найнятий).
- Пошук резюме та фільтри. Роботодавці також можуть скористатися розширеною системою фільтрації. Вказуючи рівень освіти або професійні навички майбутнього працівника, вони збільшують свої шанси знайти найбільш релевантного кандидата.

1.5.3. Вибір особливих функцій для веб-застосунку з пошуку роботи

На сьогодні не вдасться досягти успіху за допомогою стандартного веб-застосунку з пошуку роботи. Щоб зробити його великим, потрібно мислити нестандартно, створювати нові способи задовольнити запити клієнтів. Надійний спосіб досягти цієї мети - запровадити додаткові функції, яких немає у конкурентів. Пропонуючи користувачам унікальний, нестандартний

спосіб вирішення їхніх проблем, можна завоювати визнання клієнтів. Як результат, веб-застосунок для пошукачів та роботодавців отримає необхідний поштовх, щоб приносити стабільний дохід.

Нижче розглянуті деякі "особливі" функції, які можна впровадити на своєму веб-застосунку з працевлаштування, щоб він відрізнявся від десятків подібних платформ.

- Відео-співбесіди. Спалах Covid-19 змусив багато компаній перейти на віддалену роботу. У цих нових умовах функція відео-інтерв'ю може стати дуже зручним способом відбору кандидатів для рекрутерів.
- Система відстеження заявок. За допомогою цієї функції рекрутери можуть легко відстежувати прогрес кандидатів на шляху до працевлаштування. Зокрема, рекрутери можуть перевірити, чи пройшов кандидат скринінг, чи надав необхідні документи.
- Оцінювання. Оцінювання може підвищити ефективність процесу найму. Перевіряючи результати оцінки навичок, надані кандидатом, рекрутери можуть визначити, чи підходить ця людина для конкретної вакансії.
- Фільтрація резюме. Цей інструмент може заощадити рекрутерам багато часу. За допомогою скрейпінгу резюме вони автоматично переглядають десятки резюме, щоб зіставити навички, згадані в них, з навичками, необхідними для конкретної вакансії.

1.5.4. Вибір підходу до розробки

Після визначення з моделлю доходу та функціоналом майбутнього веб-застосунку з пошуку роботи, настає найцікавіша частина: вибір підходу до розробки. Багато експертів рекомендують обирати його з розумом, оскільки він вплине на терміни реалізації проекту та кінцеву вартість готового веб-застосунку з пошуку роботи. Нижче розглянуті три найпопулярніші варіанти:

- Системи управління контентом.

Існує кілька популярних систем управління контентом, які можна використовувати, якщо потрібно створити веб-застосунок для пошуку роботи якнайшвидше. Це Drupal, Joomla, Magento та WordPress.

У WordPress, ключовою перевагою систем управління контентом є велика кількість шаблонів, тем і плагінів, які дають достатню свободу дій. Однак цей варіант не бездоганний. По-перше, якщо трапиться збій або помилка, знадобиться хоча б мінімальна технічна підготовка, щоб її виправити. Крім того, не можна використовувати плагіни або теми з інших платформ. Також можна зіткнутися з проблемами масштабування. Це найбільша проблема для клієнтів, які використовують CMS для розробки веб-застосунків.

- Готові програмні рішення.

Можна обрати програмне забезпечення для дошки оголошень SaaS, наприклад, Recooty або Smartjobboard. Як і в попередньому варіанті, готове програмне забезпечення підійде, якщо потрібно швидко запустити дошку оголошень протягом декількох днів. З цими продуктами технічні навички не потрібні. Крім того, не доведеться турбуватися про збої, помилки або падіння продуктивності. Постачальник програмного забезпечення подбає про будь-яку технічну проблему.

Що стосується недоліків програмного забезпечення для дошки оголошень SaaS, то слід згадати відсутність контролю та нульову кастомізацію. Ну буде змоги використовувати власний шаблон або додати нову функціональність.

- Розробка програмного забезпечення на замовлення.

Вибравши цей підхід, розробник отримає унікальне масштабоване рішення. Воно буде враховувати всі специфічні вимоги. Крім того, завжди можна додавати нові функції, щоб забезпечити користувачам найкращий досвід. Отриманий продукт матиме виразний дизайн, який може виділитися серед конкурентів.

1.5.5. Початок розробки з MVP веб-застосунку дошки оголошень про вакансії

Якщо розробник обрав індивідуальний підхід до розробки програмного забезпечення, рекомендується почати з мінімально життєздатного продукту. Є кілька причин запустити MVP-версію веб-застосунку з пошуку роботи, перш ніж створювати повнофункціональне програмне рішення.

Перша - це економічна ефективність. За допомогою концепції мінімально життєздатного продукту можна протестувати ринковий попит на обране рішення, не вкладаючи надто багато грошей. Немає необхідності створювати складну функціональність. Базового набору функцій, який показує, як продукт вирішуватиме проблеми цільової аудиторії, буде достатньо.

Друга причина - швидкий запуск. Створення програмного продукту з обмеженим функціоналом не займе багато часу. Отже, можна буде випустити його на ринок досить швидко. Власник веб-застосунку швидше отримає цінний зворотній зв'язок. Як наслідок, може швидко покращити свою пропозицію. Крім того, зможе раніше протестувати свої маркетингові стратегії. Таким чином, можна просувати свій бізнес з розумом. Як тільки розробник отримає зворотній зв'язок від користувачів-першопрохідців, він зможе удосконалити існуюче рішення.

Отже, у ході розгляду теоретичних основ з побудови веб застосунку з пошуку роботи з ІТ спеціальності, були досліджені основні поняття з пошуку роботи, процеси пошуку роботи, основні документи для подачі заявки на працевлаштування та процес співбесіди на роботу. З’ясовані поняття веб-застосунку для пошуку роботи. Вивчені критерії унікальності веб-застосунків пошуку роботи з ІТ спеціальностей. Переглянуті та засвоєні основні виклики при створенні веб-застосунку для пошуку роботи ІТ спеціальностей. Розглянуті основні кроки з розробки веб застосунку

РОЗДІЛ 2
АНАЛІЗ ПРОГРАМНО-ТЕХНОЛОГІЧНИХ РІШЕНЬ ДЛЯ СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПОШУКУ РОБОЧИХ МІСЦЬ З ІТ СПЕЦІАЛЬНОСТЕЙ

2.1. Технологічний стек для розробки додатків для веб-застосунків з працевлаштування

Технологічний стек - один з найважливіших аспектів успіху будь-якого додатку. Всі інвестори хочуть знати найкращі але лише деякі з них усвідомлюють важливість наявності надійного технологічного стеку.

Хоча новітні технології можуть застаріти, надійні технології не будуть застарівати ще багато років. Нижче наведені основні бібліотеки та технології, які сертифіковані фахівці використовують для створення крос-платформного порталу вакансій додаток.

Для платформи Android:

- Фронтенд розробка на Java та Kotlin.
- Розробка з підтримкою: Python, Laravel, C++, JavaScript
- Ul: Android Ul, Jetpack Compose

- Інструменти розробки: Android Studio.

Для платформи iOS:

- Front-end розробка: Swift та Objective C
- Розробка з підтримкою: Python, C#
- Ulkit та Swift Ul
- Інструменти розробки: XCode та AppCode

Для крос-платформи:

- Front-end розробка: Flutter, React Native
- Розробка з підтримкою: C#, JavaScript, Python
- Ul: Ionic API
- Appcelerator та PhoneGap - інструменти для розробки

2.2. Важливі аспекти у створенні веб-застосунку для пошуку робочих місць з IT спеціальностей

2.2.1. Безпека

Користувачі, швидше за все, зберігатимуть свої персональні дані на платформі, включаючи контактну інформацію, таку як адреси електронної пошти та номери телефонів. Навіть якщо власник надасть користувачам засоби зв'язку на веб-застосунку, є ймовірність, що деякі з них віддадуть перевагу особистому спілкуванню. Забороняти його повністю може бути нерозумно. Необхідно створити веб-застосунок з функціоналом для приховування конфіденційної інформації від незареєстрованих або непідтверджених користувачів, а також від користувачів з низьким рейтингом. Також впровадити технології, які захистять веб-застосунок від хакерських атак.

2.2.1.1. Як захистити свій веб-сервіс від хакерів

Кожна компанія усвідомлює, що кібератаки становлять серйозну цифрову загрозу для її діяльності. Однак багато компаній не знають, які

заходи безпеки слід вживати, щоб захистити веб-застосунки від хакерів, нижче наведені деякі з них:

- Встановлення SSL та плагінів безпеки. Одним з основних кроків для захисту веб-застосунку від злому є встановлення SSL і плагінів безпеки. В результаті інформація, що надсилається з веб-застосунку, буде зашифрована, і її отримують тільки цільові одержувачі.
- Встановлення найновішого програмного забезпечення для захисту. Власники веб-застосунків повинні завжди перевіряти, чи встановлено найновіше програмне забезпечення для забезпечення безпеки. Ця порада особливо важлива для власників веб-застосунків, які використовують у своїй діяльності CMS, такі як WordPress, що мають багато плагінів. Оновлення містять спеціальні виправлення та функції безпеки, призначені для усунення нових загроз і, таким чином, захищають веб- веб-застосунки від хакерів.
- Використання надійного паролю. Власнику необхідно перевірити, чи використовує він надійний пароль. Надійний пароль не повинен бути відстежуваним і повинен містити спеціальні символи, цифри та літери.
- Використання протоколу https. HTTPS шифрується з метою підвищення безпеки передачі даних. Це особливо важливо, коли користувачі передають конфіденційні дані.
- Контроль даних, які користувачі завантажують на веб-застосунок. Якщо користувачам необхідно завантажувати певні файли, то слід вказати, які розширення дозволені, а також максимально допустимий розмір файлу. Також потрібно робити сканування завантажених файлів, оскільки вони можуть містити шкідливе програмне забезпечення. Завантажені файли слід зберігати окремо, щоб навіть якщо вони містять якесь шкідливе програмне забезпечення, воно не зашкодило функціонуванню веб-застосунку та безпеці його даних.

- Створення резервних копій веб-застосунку. Варто створювати резервні копії веб-застосунку, якщо його зламали, можна буде відновити всі дані і повернути веб-застосунок до нормального функціонування.
- Вибір надійних хостинг-провайдерів. Необхідно обирати надійних хостинг-провайдерів, які регулярно перевіряють логи на наявність доступу відомих зловмисників і надають часті резервні копії. У разі кібератаки відповідальний постачальник послуг негайно співпрацюватиме з власником веб-застосунку для фільтрації трафіку. Може бути доцільно перевірити історію інцидентів з безпекою, пов'язаних з хостинг-провайдером.
- Використання лише необхідних плагінів. Варто використовувати лише належним чином підтримувані плагіни, які дійсно потрібні у діяльності веб-застосунку. Якщо плагін не підтримувався роками або містить відомі вразливості, не можна його використовувати.

2.2.2. Алгоритми пошуку

Щоб зіставити пропозиції роботи з профілями кандидатів, розробнику знадобляться алгоритми зіставлення. Традиційні алгоритми працюють на основі ключових слів, але це не надійний метод. Іноді схожі вакансії можуть називатися по-різному, і пошук за ключовими словами пропустить кілька релевантних профілів просто тому, що слова не збігаються. Хороший алгоритм зіставлення працює над вирішенням таких проблем.

Сьогодні все більше веб-застосунків, які займаються будь-якими видами пошуку (наприклад, сервіси знайомств), використовують штучний інтелект і машинне навчання для покращення результатів. Самонавчальні системи можуть розширювати або звужувати результати пошуку за допомогою даних, які вони збирають від користувачів.

2.3.2.1. Що таке алгоритм пошукової системи?

Алгоритм пошукової системи - це набір формул, які визначають якість і релевантність конкретного оголошення або веб-сторінки запиту користувача. Як повідомляється, Google змінює свій алгоритм сотні разів

щороку. Хороша новина: тільки великі зміни (або оновлення) можуть суттєво вплинути на SEM-кампанії. Одна з найбільших помилок маркетологів полягає в тому, що вони зосереджують усі свої зусилля на гарячковому пристосуванні кампанії до цих формул замість того, щоб дивитися на картину в цілому.

2.3.2.2. Як працюють алгоритми пошукових систем

Пошукові системи роблять користувацький досвід головним пріоритетом. Google вдалося стати найпопулярнішою пошуковою системою на планеті завдяки створенню складних алгоритмів, які покращують процес пошуку за допомогою витончених тактик, що надають користувачам інформацію, яку вони шукають. Алгоритм працює з усіма видами деталей для контексту, від очевидних підказок, таких як сприйняття якості контенту, до історії спаму власника веб-застосунку. Загалом, Google використовує понад 200 факторів ранжування, визначаючи, які результати показувати і в якому порядку.

2.3.2.3. Типи оновлень алгоритмів пошукових систем

Практично неможливо відстежувати всі оновлення, які випускає Google, і при цьому встигати зосередитися на маркетинговій стратегії.

1. Основні оновлення. Ці оновлення виходять нечасто і часто стосуються конкретної проблеми пошукового алгоритму. Наприклад, нещодавнє оновлення Core Web Vitals вирішує проблеми, пов'язані з користувацьким досвідом на веб-сторінках. Пошукові системи зазвичай випускають їх один або два рази на рік.
2. Широкомасштабні оновлення. Оновлення цієї категорії спрямовані на таргетування неякісних сторінок. Зазвичай вони коригують важливість кількох факторів ранжування. Наприклад, вони можуть вирішити, що швидкість завантаження сторінки тепер важливіша за загальну кількість зворотних посилань. Такі оновлення зазвичай відбуваються раз на 4-5 місяців.

3. Невеликі оновлення. Ці оновлення зазвичай не спричиняють значних видимих змін у продуктивності та аналітиці веб-застосунку. Часто це незначні зміни, які покращують роботу пошукачів і не впливають на ранжування якісних веб-застосунків. Незначні оновлення можна впроваджувати щодня або щотижня.

2.3 Огляд аспектів розробки бізнес-логіки веб-застосунку для пошуку робочих місць з IT спеціальностей

2.3.1 Java мова програмування для створення бізнес-логіки веб-застосунку

Java - це дуже популярна мова програмування, яка відома своєю високорівневою, об'єктно-орієнтованою природою. Вона була створена для того, щоб мінімізувати залежності при реалізації та надавати універсальну платформу для розробників. Java має принцип "напиши один раз, виконуй будь-де", який дозволяє скомпільованому Java-коду працювати на різних платформах без перекомпіляції. Програми на Java зазвичай компілюються в байт-код, що дозволяє їм працювати на будь-якій віртуальній машині Java (JVM), незалежно від архітектури комп'ютера. Хоча Java має схожість з C та C++ з точки зору синтаксису, вона пропонує менше низькорівневих можливостей. Однак вона компенсує це динамічними можливостями, такими як рефлексія та модифікація коду під час виконання, які зазвичай відсутні у традиційних мовах, що компілюються.

Java дуже популярна в сфері клієнт-серверних веб-додатків, а за оцінками, 9 мільйонів розробників використовують цю мову. Вона була створена Джеймсом Гослінгом у Sun Microsystems і вперше представлена у травні 1995 року як основний компонент платформи Java від Sun. Спочатку Sun випускала компілятори Java, віртуальні машини та бібліотеки класів під власними ліцензіями. Однак, починаючи з травня 2007 року, Sun почала переліцензувати більшість своїх технологій Java лише під ліцензією GPL-2.0, відповідно до специфікацій Процесу спільноти Java.

У той час як Oracle надає свою віртуальну машину Java HotSpot, OpenJDK JVM є офіційною еталонною реалізацією, широко прийнятою розробниками і слугує JVM за замовчуванням для більшості дистрибутивів Linux. Станом на березень 2023 року Java 20 є останньою версією, а Java 17, 11 та 8 - версіями з довгостроковою підтримкою (LTS).

Java також знайома своїми різноманітними бібліотеками, які дозволяють програмістам створювати програми швидше та ефективніше. Наприклад, бібліотека Swing дозволяє створювати графічні інтерфейси користувача, а бібліотека JavaFX дозволяє створювати інтерактивні веб-додатки. Крім того, Java має різноманітні фреймворки, такі як Spring та Hibernate, які спрощують розробку веб-додатків та зменшують кількість коду.

Java також має велику спільноту розробників, яка активно спілкується між собою та допомагає один одному розв'язувати проблеми. Ця спільнота не тільки допомагає новачкам зрозуміти мову, але й працює над розробкою нових бібліотек та фреймворків для поліпшення екосистеми Java.

В цілому, Java є дуже потужною мовою програмування, яка має велику кількість можливостей та функцій. Вона знайома своїми високорівневими можливостями, що дозволяє програмістам швидше створювати програми. Принцип "напиши один раз, виконуй будь-де" дозволяє створювати програми, які можна використовувати на різних платформах, а різноманітні бібліотеки та фреймворки дозволяють розробникам створювати програми швидше та ефективніше.

2.3.2 Історія мови програмування Java

Проект мови Java був ініційований у червні 1991 року Джеймсом Гослінгом, Майком Шеріданом і Патріком Нотоном. Спочатку мова була призначена для розважального телебачення, однак виявилася занадто розвиненою для того часу. У процесі розробки проекту мова мала кілька назв: Oak, Green та Java. Кожне з цих ім'я було обране з різних причин, але Java була інспірована індонезійською кавою Ява. Java була спроектована з

синтаксисом, схожим на C/C++, щоб зробити її звичною для системних і прикладних програмістів.

Перша публічна реалізація Java, випущена Sun Microsystems, була Java 1.0 у 1996 році. Однією з головних переваг Java стало впровадження концепції "напиши один раз, запускай будь-де" (WORA), що дозволяє запускати Java-додатки на різних платформах без додаткових витрат на час виконання. Java також надала безпеку, включаючи конфігуровані обмеження доступу для мережевих і файлових операцій. Основні веб-браузери швидко інтегрували підтримку запуску аплетів Java на веб-сторінках, що призвело до швидкого впровадження Java. У 1997 році Sun Microsystems співпрацювала з такими органами стандартизації, як ISO/IEC JTC 1 та Ecma International, щоб формалізувати Java, але врешті-решт вийшла з цього процесу.

Пізніше, у 2006 році, з маркетингових міркувань Sun перейменувала версії Java EE, Java ME та Java SE з Java 2. Наразі вони відомі як Java Platform, Enterprise Edition (Java EE), Java Platform, Micro Edition (Java ME) та Java Platform, Standard Edition (Java SE), відповідно. Після придбання компанією Oracle компанії Sun Microsystems, Oracle позиціонувала себе як розпорядника технології Java та активно сприяє розвитку спільноти, заснованої на участі та прозорості. Тим не менш, Oracle подала позов проти Google за використання Java в Android SDK, що спричинило ряд проблем у спільноті.

Програмне забезпечення Java може працювати на широкому спектрі пристроїв, від ноутбуків і центрів обробки даних до ігрових консолей і наукових суперкомп'ютерів. Oracle та інші експерти галузі наполегливо рекомендують видалити застарілі та не підтримувані версії Java через невирішені проблеми з безпекою, наявні у старих версіях. Java залишається однією з найпопулярніших мов програмування у світі, а її спільнота продовжує активно розвиватися.

2.3.3 Фреймворк Spring для створення бізнес-логіки веб-застосунку

Spring Framework є одним з найпопулярніших фреймворків додатків та контейнерів управління зворотними залежностями для платформи Java. Він надає широкий набір функцій, які можуть бути використані в будь-якому Java-додатку, а також має розширення для розробки веб-додатків на платформі Java EE (Enterprise Edition). Хоча Spring не обмежується певною моделлю програмування, він став популярним серед Java-розробників як доповнення до моделі Enterprise JavaBeans (EJB). Важливою особливістю Spring є його відкритий вихідний код.

Spring спрощує розробку корпоративних додатків на Java, надаючи необхідні інструменти для використання мови Java в корпоративному середовищі. Він підтримує Groovy та Kotlin як альтернативні мови для JVM і дозволяє гнучко створювати різноманітні архітектури залежно від потреб програми. Починаючи з версії Spring Framework 6.0, для його використання вимагається Java 17 або новіша версія.

Spring підтримує різноманітні сценарії застосування. У великих підприємствах додатки часто працюють протягом тривалого періоду та повинні бути сумісними з різними версіями JDK та серверів додатків. Інші додатки можуть бути упаковані в один виконуваний файл з вбудованим сервером, а можуть бути розгорнуті у хмарному середовищі. Деякі додатки можуть бути автономними, не вимагаючи сервера, і використовуватися для пакетної обробки або інтеграційних завдань.

Spring має велику та активну спільноту розробників, що сприяє постійному зворотному зв'язку та підтримці. Ця спільнота базується на реальних випадках використання та надає цінні відгуки та рекомендації. Благодаря такій активній спільноті, Spring успішно розвивається та покращується протягом тривалого періоду часу.

Spring Framework дозволяє розробникам Java зосередитись на логіці додатку, забезпечуючи потужні інструменти для управління залежностями, роботи з базами даних, взаємодії з веб-сервісами та іншими розширеними

функціями. Він також сприяє використанню різних проектних шаблонів та добрим практикам, що полегшують розробку високоякісних додатків.

У підсумку, Spring Framework є потужним інструментом для розробки Java-додатків, особливо в корпоративному середовищі. Його відкритий вихідний код та активна спільнота роблять його популярним серед розробників. Завдяки своїм багатим можливостям та гнучкості, Spring став важливим компонентом в екосистемі розробки на платформі Java.

2.3.4 Історія Spring Framework

Род Джонсон написав початкову версію Spring Framework і представив її разом зі своєю книгою "Expert One-on-One J2EE Design and Development" у жовтні 2002 року. Спочатку фреймворк був випущений під ліцензією Apache 2.0 у червні 2003 року. Перший офіційний реліз, версія 1.0, став доступний у березні 2004 року. Цей реліз отримав визнання, вигравши як нагороду за продуктивність Jolt, так і нагороду за інновації JAX у 2006 році.

Наступні версії Spring Framework були випущені у жовтні 2006 року: Spring 2.0, Spring 2.5 у листопаді 2007 року, Spring 3.0 у грудні 2009 року, Spring 3.1 у грудні 2011 року та Spring 3.2.5 у листопаді 2013 року. Зокрема, Spring Framework 4.0, випущений у грудні 2013 року, приніс значні покращення, такі як підтримка Java SE 8, Groovy 2, деяких функцій Java EE 7 та WebSocket.

Spring Boot, фреймворк для створення автономних Spring-додатків, дебютував з версією 1.0 у квітні 2014 року. Він спростив процес розробки, надавши продумані налаштування за замовчуванням та автоматичну конфігурацію.

Spring Framework 4.2.0 було випущено 31 липня 2015 року, а 1 вересня 2015 року вийшла версія 4.2.1. Ця версія зберегла сумісність з Java 6, 7 і 8, зосередившись при цьому на вдосконаленні ядра та сучасних веб-можливостях.

10 червня 2016 року було випущено Spring Framework 4.3 з обіцянкою підтримки до 2020 року. Це була остання велика версія в рамках загальних вимог Spring 4, яка включала Java 6+ та Servlet 2.5+.

Було оголошено, що Spring 5 буде побудовано на Reactor Core, сумісному з Reactive Streams, що додасть фреймворку можливості реактивного програмування.

Останній великий випуск, Spring Framework 6.0, став доступний 16 листопада 2022 року. У ньому було прийнято базову лінію Java 17+ та включено стандарти Jakarta EE 9+ у просторі імен jakarta. Основна увага в цьому випуску була приділена використанню нещодавно випущених API-інтерфейсів Jakarta EE 10, включаючи Servlet 6.0 і JPA 3.1.

2.3.5 Принцип інверсії контролю у Spring

В основі Spring Framework лежить контейнер інверсії управління (IoC), який пропонує послідовний підхід до конфігурації та управління об'єктами Java за допомогою рефлексії. Контейнер бере на себе відповідальність за управління життєвими циклами конкретних об'єктів, включаючи їх створення, виклик методів ініціалізації та конфігурацію за допомогою проводки.

Керовані об'єкти або біни - це об'єкти, створені контейнером. Контейнер можна конфігурувати за допомогою XML-файлів або шляхом виявлення певних анотацій Java у класах конфігурації. Ці джерела конфігурації містять визначення бобів, які надають необхідну інформацію для створення бобів.

Об'єкти можна отримати за допомогою пошуку залежностей або ін'єкції залежностей. Пошук залежностей відбувається за шаблоном, коли користувач запитує об'єкт з певним ім'ям або типом з контейнера. Ін'єкція залежностей, з іншого боку, передбачає надання контейнером об'єктів за іменами іншим об'єктам через конструктори, властивості або фабричні методи.

Хоча Spring Framework пропонує інші компоненти, які можна використовувати без контейнера, використання контейнера може значно підвищити легкість конфігурації та кастомізації програми. Контейнер Spring слугує послідовним механізмом для конфігурації додатків і легко інтегрується з різними середовищами Java, від невеликих до великих корпоративних додатків.

Завдяки проекту Pitchfork контейнер можна перетворити на частково сумісний з Enterprise JavaBeans (EJB) 3.0 контейнер. Деякі критики стверджують, що Spring Framework відхиляється від стандартів. Однак SpringSource розглядає відповідність EJB 3 як другорядну мету і стверджує, що фреймворк і його контейнер дозволяють створювати більш потужні моделі програмування. Замість того, щоб безпосередньо створювати об'єкти, програмісти описують, як вони повинні бути створені, у файлі конфігурації Spring. Аналогічно, сервіси та компоненти не викликаються безпосередньо, а визначаються у конфігураційному файлі Spring, щоб вказати, які з них слід викликати. Така інверсія управління має на меті покращити простоту обслуговування та полегшити тестування.

2.3.6 Фреймворк що розширює Spring

Spring Boot - це фреймворк, який розширює можливості Spring Framework, спрощуючи створення незалежних, готових до виробництва додатків на основі Spring. Його основна мета - спростити процес розробки за рахунок використання підходу "конвенції над конфігурацією", що дозволяє розробникам швидко налаштовувати та розгортати Spring-додатки з мінімальними зусиллями.

Використовуючи автоматичну конфігурацію, Spring Boot усуває необхідність ручного налаштування. Він автоматично налаштовує Spring-додатки на основі залежностей, виявлених у шляху класів проекту. Крім того, Spring Boot надає повну колекцію стартових залежностей, які є попередньо сконфігурованими залежностями, пристосованими для конкретних випадків використання, таких як веб-додатки, інтеграція з базами даних та безпека. Ці

стартові залежності дозволяють розробникам без зусиль включати необхідні залежності у свої проекти без зайвого клопоту з індивідуальним управлінням.

Підтримка вбудованих серверів - ще один важливий аспект Spring Boot. Він включає вбудовані контейнери, такі як Tomcat, Jetty та Undertow, що полегшує пакування додатків у вигляді окремих виконуваних JAR-файлів. Ці JAR-файли можуть виконуватися безпосередньо без необхідності розгортання на окремому сервері, що спрощує процес розгортання.

Spring Boot використовує архітектуру мікросервісів і пропонує функції, які підтримують її розвиток. Наприклад, він полегшує зовнішнє конфігурування, дозволяючи легко керувати конфігураціями поза кодом програми. Крім того, Spring Boot надає надійну підтримку для створення RESTful API та легко інтегрується з популярними фреймворками та бібліотеками. Наприклад, Spring Data для ефективного доступу до баз даних, Spring Security для аутентифікації та авторизації та Spring Cloud для побудови розподілених систем.

2.3.7 Різниція Spring Boot та Spring Framework

Якщо порівнювати Spring Boot з Spring Framework, то основними перевагами використання Spring Boot є його простота та прискорення процесу розробки. Завдяки цим перевагам, можна заощадити значну кількість часу та зусиль, що зазвичай витрачаєте на розробку додатку з використанням Spring Framework.

Хоча робота безпосередньо з Spring Framework пропонує більшу гнучкість в теорії, на практиці цей компроміс, як правило, виправдовує себе, якщо тільки не потрібна дуже індивідуальна конфігурація. Однак, за допомогою Spring Boot все ще можливо використовувати широко розповсюджену систему анотацій Spring Framework, що дозволяє без особливих зусиль додавати додаткові залежності до додатку, які можуть бути не охоплені Spring Starters.

Більше того, можна зберегти доступ до всіх можливостей Spring Framework, включаючи зручну обробку подій, валідацію, прив'язку даних,

перетворення типів та вбудовані можливості безпеки та тестування. За допомогою Spring Boot, можливо швидко розгорнути прості додатки, які можуть бути налаштовані та розширені з використанням великої кількості додаткових бібліотек та інструментів. По суті, якщо проект може бути виконаний навіть одним Spring Starter, Spring Boot можна значно спростити робочий процес розробки та допомогти створити більш потужний та розширюваний додаток.

2.4 Огляд аспектів розробки система управління базами даних веб-застосунку для пошуку робочих місць з IT спеціальностей.

2.4.1 Визначення баз даних

База даних — це організований набір даних, які зберігаються та доступні в електронному вигляді. Невеликі бази даних можна зберігати у файловій системі, тоді як великі бази даних розміщуються на комп'ютерних кластерах або хмарних сховищах. Дизайн бази даних включає формальні методи та практичні міркування, включаючи моделювання даних, ефективне представлення та зберігання даних, мови запитів, безпеку та конфіденційність конфіденційних даних, а також питання розподіленого обчислення, включаючи підтримку одночасного доступу та відмовостійкість.

Система керування базами даних (СУБД) — це програмне забезпечення, яке взаємодіє з кінцевими користувачами, програмами та самою базою даних для збору та аналізу даних. Програмне забезпечення СУБД також містить основні засоби, передбачені для керування базами даних. Сукупність баз даних, СУБД і відповідних додатків можна назвати системою баз даних. Загалом, термін «база даних» також використовується в широкому сенсі для позначення будь-якої СУБД, системи баз даних або пов'язаної з базою даних програми.

Інформатики можуть класифікувати системи керування базами даних відповідно до моделі бази даних, яку вони підтримують. У 1980-х роках домінували реляційні бази даних. Вони моделюють дані як серію рядків і

стовпців у таблицях і переважно використовують SQL для запису та запиту даних. У 2000-х роках нереляційні бази даних (загально відомі як NoSQL) стали популярними, оскільки вони використовували інші мови запитів.

2.4.2 Типи баз даних

Існує кілька типів баз даних, кожна з яких призначена для певних цілей і задовольняє різні потреби в управлінні даними. Відомі такі типи баз даних:

- Реляційна база даних: Реляційні бази даних базуються на реляційній моделі і організовують дані в таблиці з рядками і стовпчиками. Вони використовують мову структурованих запитів (SQL) для маніпулювання та пошуку даних. Прикладами є MySQL, Oracle Database та Microsoft SQL Server.
- База даних NoSQL: Бази даних NoSQL (не тільки SQL) призначені для обробки великих обсягів неструктурованих і напівструктурованих даних. Вони забезпечують гнучкий дизайн схем і горизонтальну масштабованість. Прикладами є MongoDB, Cassandra та Redis.
- Об'єктно-орієнтована база даних: Об'єктно-орієнтовані бази даних (OODB) зберігають дані у вигляді об'єктів, які складаються з полів даних і пов'язаних з ними методів або функцій. Вони особливо підходять для додатків зі складними структурами даних або об'єктно-орієнтованими мовами програмування. Прикладами є ObjectDB та Versant.
- Графові бази даних: Графові бази даних зберігають і представляють дані за допомогою графових структур, що складаються з вузлів (сутностей) і ребер (зв'язків). Вони відмінно справляються з управлінням сильно взаємопов'язаними даними і ефективні для складних запитів на зв'язки. Прикладами є Neo4j, Amazon Neptune та JanusGraph.
- Ієрархічна база даних: Ієрархічні бази даних організовують дані у деревоподібний структурі зі зв'язками "батько-нащадок", де кожен нащадок має лише одного батька. Вони підходять для зберігання

ієрархічних даних, таких як організаційні структури або файлові системи. Система управління інформацією (IMS) від IBM є прикладом ієрархічної бази даних.

- Мережеві бази даних: Мережеві бази даних також організовують дані у деревоподібний структурі, але на відміну від ієрархічних баз даних, вони дозволяють кожному дочірньому об'єкту мати декількох батьків. Вони підходять для моделювання складних взаємозв'язків і часто використовуються в застарілих системах. Інтегроване сховище даних (IDS) є прикладом мережевої бази даних.
- База даних часових рядів: Бази даних часових рядів оптимізовані для обробки даних з позначкою часу або часових рядів, таких як дані датчиків, дані фінансового ринку або журнали. Вони забезпечують ефективне зберігання та пошук точок даних у часі. Приклади включають InfluxDB і Prometheus.
- Стовпчикова база даних: Стовпчикові бази даних організовують дані у стовпці зі значеннями, що відповідають конкретному атрибуту. Це підходить для ситуацій, коли необхідно швидко здійснювати пошук і аналіз даних. Прикладами є Apache Cassandra та Google Bigtable.
- База даних документів: База даних документів зберігає дані у вигляді документів, які можуть містити структуровані та неструктуровані дані. Вони забезпечують гнучкий дизайн та підтримку запитів, які орієнтовані на документи. Прикладами є MongoDB, Couchbase та Apache CouchDB.
- Просторова база даних: Просторові бази даних спеціалізуються на зберіганні та запиті просторових або географічних даних, таких як карти, координати або геометричні фігури. Вони пропонують просторову індексацію та підтримку просторових операцій. PostGIS (розширення для PostgreSQL) і Oracle Spatial є прикладами просторових баз даних.

- Інші спеціалізовані бази даних: Існують інші спеціалізовані бази даних, такі як бази даних з орієнтацією на події, бази даних з орієнтацією на голос, бази даних для зберігання зображень тощо. Вибір типу бази даних залежить від таких факторів, як структура даних, вимоги до масштабованості, продуктивності та конкретних випадків використання. Ймовірно, вибір спеціалізованої бази даних для конкретного випадку допоможе збільшити продуктивність та забезпечити більш гнучкий дизайн.

2.4.3 База даних що використовувалась для створення веб-застосунку для пошуку робочих місць з IT спеціальностей

PostgreSQL, яку зазвичай називають Postgres, є безкоштовною системою керування реляційною базою даних із відкритим кодом. PostgreSQL робить сильний акцент на розширюваності та дотриманні стандартів SQL. PostgreSQL отримав свою назву від свого попередника, проекту POSTGRES, який був розроблений як наступник бази даних Ingres UC Berkeley. У 1996 році назву проекту було змінено на PostgreSQL, щоб краще відобразити підтримку SQL.

PostgreSQL надає широкий спектр функцій, включаючи підтримку властивостей ACID (атомарність, послідовність, ізоляція, довговічність) у транзакціях, автоматичне оновлення подання, матеріалізовані подання, тригери, зовнішні ключі та збережені процедури. Він призначений для обробки різноманітних робочих навантажень, від однієї машини до великих сховищ даних або веб-сервісів із великою кількістю одночасних користувачів. Раніше вона слугувала базою даних за замовчуванням для macOS Server і наразі доступна для кількох операційних систем, у т.ч. Linux, FreeBSD, OpenBSD, та Windows.

2.4.4 Визначення поняття Java Database Connectivity

JDBC, що розшифровується як Java Database Connectivity, - це Java API (інтерфейс прикладного програмування), який надає набір класів і методів для взаємодії Java-додатків з реляційними базами даних. Він дозволяє

розробникам встановлювати з'єднання з базами даних, виконувати SQL-запити, отримувати та маніпулювати даними, а також керувати транзакціями баз даних.

JDBC діє як міст між Java-додатками та різними системами баз даних, забезпечуючи стандартизований спосіб доступу до даних та маніпулювання ними незалежно від конкретного постачальника баз даних, що використовується. Він дозволяє розробникам писати код, пов'язаний з базами даних, на Java, що робить його широко використовуваним і важливим компонентом в додатках на Java, які вимагають підключення до баз даних.

За допомогою JDBC розробники можуть виконувати різні операції, включаючи встановлення з'єднань з базами даних, створення та виконання операторів SQL, отримання результатів запитів, оновлення записів у базі даних та управління транзакціями для забезпечення узгодженості та цілісності даних.

2.4.5 Визначення поняття Java Persistence API

JPA, що розшифровується як Java Persistence API, - це специфікація Java, яка надає стандартизований набір API та інструкцій для управління реляційними даними в додатках на Java. Це фреймворк об'єктно-реляційного відображення (ORM), який дозволяє розробникам відображати об'єкти Java в таблиці реляційних баз даних і виконувати операції з базами даних, використовуючи концепції об'єктно-орієнтованого програмування.

JPA абстрагує низькорівневі деталі доступу до баз даних і забезпечує більш високорівневий, об'єктно-орієнтований підхід до взаємодії з базами даних. Він вводить такі поняття, як сутності, які є класами Java, що представляють постійні об'єкти, і анотації, які визначають відображення між сутностями і таблицями бази даних.

За допомогою JPA розробники можуть виконувати операції CRUD (створення, читання, оновлення, видалення) над базою даних за допомогою API, таких як EntityManager, який надає методи для управління сутностями та виконання запитів. JPA також включає мову запитів JPQL (Java Persistence

Query Language), яка дозволяє розробникам писати запити до бази даних, використовуючи атрибути сутностей та зв'язки.

2.5 Огляд аспектів розробки публічної частини веб-застосунку для пошуку робочих місць з IT спеціальностей.

2.5.1 JavaScript мова програмування для створення публічної частини веб-застосунку

JavaScript, часто скорочено JS, є мовою програмування, яка разом з HTML і CSS відіграє важливу роль у Всесвітній павутині. До 2022 року 98% усіх веб-застосунків використовують його для покращення поведінки веб-сторінок, часто інтегруючи бібліотеки сторонніх розробників. Усі основні веб-браузери оснащені спеціальними двигунами JavaScript, які виконують код безпосередньо на пристрої користувача.

JavaScript — це мова високого рівня, яка зазвичай компілюється точно вчасно та відповідає стандарту ECMAScript. Він використовує об’єктно-орієнтований підхід на основі прототипу та забезпечує першокласну функціональність завдяки динамічній типізації. Це багатопарадигмальна мова, яка підтримує подійно-орієнтований, функціональний та імперативний стилі програмування. JavaScript надає різні інтерфейси прикладного програмування (API) для таких завдань, як маніпулювання текстом, датами, регулярними виразами, стандартними структурами даних і маніпулювання обєктною моделлю документа (DOM).

Незважаючи на те, що стандарт ECMAScript не включає вбудовані функції введення/виведення (I/O), такі як мережеві, сховища або графіка, JavaScript API для введення/виведення надаються веб-браузерами або іншими системами виконання.

Механізми JavaScript, які спочатку використовувалися лише у веб-браузерах, тепер є невід’ємною частиною серверів і широкого спектру програм. Node.js став провідним середовищем виконання для таких цілей. Незважаючи на подібні назви та синтаксис, Java і JavaScript є різними

мовами, які мають значні відмінності в дизайні, незважаючи на відповідні стандартні бібліотеки.

2.5.2 Html мова для створення розмітки веб-сторінок

HTML, скорочення від HyperText Markup Language, є стандартною мовою, яка використовується для створення веб-документів, призначених для відображення у веб-браузері. Він працює разом з іншими технологіями, такими як каскадні таблиці стилів (CSS) і мовами сценаріїв, такими як JavaScript.

Коли веб-браузер отримує документ HTML із веб-сервера або локального сховища, він перетворює документ у мультимедійну веб-сторінку. HTML забезпечує семантичну структуру веб-сторінки та початково містить ознаки її візуального вигляду.

Елементи HTML служать основними будівельними блоками сторінок HTML. За допомогою конструкцій HTML зображення та інші об’єкти, включно з інтерактивними формами, можна вставляти у відтворену сторінку. HTML дозволяє створювати структуровані документи, визначаючи структурну семантику текстових елементів, таких як заголовки, абзаци, списки, посилання, цитати тощо. Ці елементи позначаються за допомогою тегів, які записуються в кутових дужках. Такі теги, як <img /> і <input />, безпосередньо вводять вміст на сторінку, тоді як інші, такі як і , містять і надають інформацію про текст документа, потенційно включаючи теги під елементів. Хоча теги HTML не відображаються в браузерах, вони використовуються для інтерпретації вмісту сторінки.

HTML також дозволяє включати програми, написані мовами сценаріїв, наприклад JavaScript, які можуть змінювати поведінку та вміст веб-сторінок. CSS використовується для визначення візуального стилю та компонування вмісту. Консорціум World Wide Web (W3C), який раніше відповідав за підтримку HTML і зараз підтримує стандарти CSS, з 1997 року заохочував використання CSS для цілей презентації замість того, щоб покладатися на явний презентаційний HTML. HTML5, версія HTML, спеціально

розроблений для відображення мультимедійного вмісту, такого як відео та аудіо, переважно з використанням елемента <canvas> у поєднанні з JavaScript.

2.5.3 Css каскадна таблиця стилів для створення стилів веб-сторінок

Каскадні таблиці стилів (CSS) - це мова, що використовується для визначення візуального представлення документа, написаного мовою розмітки HTML або XML (включаючи діалекти XML, такі як SVG, MathML або XHTML). Поряд з HTML і JavaScript, CSS є фундаментальною технологією Всесвітньої павутини.

Основна мета CSS - відокремити вміст від його представлення, охоплюючи такі аспекти, як макет, кольори та шрифти. Таке розділення дає різні переваги, зокрема покращує доступність контенту, підвищує гнучкість і контроль над характеристиками презентації, можливість спільного використання форматування на декількох веб-сторінках шляхом посилання на окремий файл .css, зменшує складність і повторюваність у структурному контенті. Крім того, кешування файлу .css допомагає підвищити швидкість завантаження сторінок, які мають спільний доступ до цього файлу та його форматування.

Відокремлення форматування від контенту також дозволяє представити одну й ту саму сторінку розмітки в різних стилях для різних методів відображення, таких як екранний перегляд, друк, перегляд на основі мовлення для користувачів голосових або екранних читачів, а також тактильних пристроїв на основі шрифту Брайля. CSS надає правила для альтернативного форматування при доступі до контенту на мобільних пристроях.

Термін "каскадний" у CSS означає визначену схему пріоритетів, яка визначає, яке правило стилю має пріоритет, коли певному елементу відповідає кілька правил. Ця каскадна схема пріоритетів дотримується передбачуваного порядку.

Специфікації CSS підтримуються Консорціумом Всесвітньої павутини (W3C). Тип MIME text/css зареєстрований для використання CSS відповідно до RFC 2318 (березень 1998 року). W3C також надає безкоштовну службу валідації CSS для перевірки документів CSS.

Окрім HTML, CSS підтримується іншими мовами розмітки, такими як XHTML, звичайний XML, SVG і XUL. Вона також використовується у інструментарії віджетів GTK.

2.5.4 React js - фреймворк для створення публічної частини застосунку

React, також відомий як React.js або ReactJS, - це популярна і широко використовувана бібліотека JavaScript для побудови користувацьких інтерфейсів за допомогою компонентів. Це безкоштовна бібліотека з відкритим вихідним кодом, яка підтримується компанією Meta, раніше відомою як Facebook, у співпраці з великою спільнотою розробників та організацій по всьому світу.

Однією з головних переваг використання React є його гнучкість, яка дозволяє розробникам створювати широкий спектр додатків, включаючи, але не обмежуючись, односторінкові додатки, мобільні додатки та додатки з серверним рендерингом. React також можна використовувати разом з іншими фреймворками, такими як Next.js, який надає додаткові можливості, такі як рендеринг на стороні сервера та автоматичне розділення коду.

Крім того, React фокусується в основному на користувацькому інтерфейсі та рендерингу компонентів до Document Object Model (DOM), що робить його високоефективною бібліотекою. Однак для більш складних завдань, таких як маршрутизація та клієнтські функції, можуть знадобитися додаткові бібліотеки, щоб повністю використати можливості React.

Таким чином, React - це потужна та універсальна бібліотека, яка може допомогти розробникам створювати високоякісні та візуально привабливі інтерфейси для різних типів додатків, з додатковою перевагою - з відкритим вихідним кодом та вільним доступом.

Отже, у ході аналізу програмно технологічних рішень з побудови веб застосунку, було розглянуто технологічний стек для розробки додатків для веб-застосунків з працевлаштування. Засвоєні важливі аспекти у створенні веб-застосунку для пошуку робочих місць з ІТ спеціальностей. Вивчені аспекти розробки бізнес-логіки веб-застосунку для пошуку робочих місць з ІТ спеціальностей, включаючи: Java мова програмування для створення бізнес-логіки веб-застосунку, Фреймворк Spring для створення бізнес-логіки веб-застосунку, Принцип інверсії контролю у Spring, Фреймворк що розширює Spring, Різниця Spring Boot та Spring Framework. Проаналізовані аспекти розробки система управління базами даних веб-застосунку для пошуку робочих місць з ІТ спеціальностей, що складаються з: Визначення баз даних та Типи баз даних, База даних що використовувалась для створення веб-застосунку для пошуку робочих місць з ІТ спеціальностей. Оглянуті аспекти розробки публічної частини веб-застосунку для пошуку робочих місць з ІТ спеціальностей. Переглянутий стек технологій для реалізації публічної частини веб-застосунку.

РОЗДІЛ 3
ПРОЕКТУВАННЯ, РОЗРОБЛЕННЯ, РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПОШУКУ РОБОЧИХ МІСЦЬ З ІТ СПЕЦІАЛЬНОСТЕЙ

3.1 Інженерія вимог до веб-застосунку для пошуку робочих місць з ІТ спеціальностей

3.1.1 Постановка задачі

Після проведення аналізу фундаментальних теоретичних принципів проектування веб-застосунків та виділення основних рекомендацій і пасток, можна сформулювати основну мету даної роботи: розробити прототип веб-застосунку для пошуку робочих місць в галузі ІТ-спеціальностей для представника малого бізнесу.

Застосунок буде побудований на основі клієнт-серверної архітектури та використовуватиме сучасні технології.

Для виконання поставленої задачі необхідно послідовно вирішити наступні завдання:

- Провести аналіз функціоналу, переваг та недоліків схожих ресурсів;
- розробити компонент доступу до даних з використанням реляційних баз даних;
- розробити зручний клієнт-орієнтований інтерфейс взаємодії з користувачами;
- забезпечити головні вимоги до побудови веб-застосунку та впровадити необхідний функціонал.

При побудові системи потрібно дотримуватися основних вимог та рекомендацій, наведених в попередніх розділах роботи, щоб забезпечити максимально ефективну та зручну систему здійснення пошуку роботи через Інтернет.

3.1.2 Аналіз функціоналу схожих систем пошуку роботи

Для створення актуального та релевантного ресурсу з пошуку робочих місць в сфері IT, необхідно спиратися не лише на теоретичні знання щодо розробки програмних систем для пошуку роботи. Окрім цього, важливим етапом проектування ефективної системи є аналіз конкурентних ресурсів з аналогічною тематикою.

На сучасному світовому та українському ринках, найбільш популярними та відомими ресурсами з пошуку роботи в IT-сфері є міжнародний веб-застосунок Jooble.com та українські ресурси Djinni.co і Dou.ua. Для кращого розуміння їх функціоналу та особливостей, було вирішено провести порівняльний аналіз, який наведений у Таблиці 3.1.

Таблиця 3.1 - Порівняння веб-застосунків для пошуку роботи із IT спеціальностей

Назва ресурсу	Jooble.com	Djinni.co	Dou.ua

Інтерфейс	Інтуїтивний	Складний	Інтуїтивний
Вибір мови	Ні	так	ні
Вхід через соціальні мережі	так	так	частково
Блог	ні	ні	так
Пошук вакансії	так	так	так
Пошук за категоріями	так	так	так
Пошук вакансій по компаніям	ні	частково	так
Фільтрація вакансій	частково	так	так
Сортування вакансій	частково	так	частково

Продовження таблиці 3.1

Можливість відправити відгук на вакансію	так	так	так
Можливість перегляду власного відгуку на вакансію	ні	ні	ні

Аналізуючи основний функціонал веб-застосунку для пошуку робочих місць з IT спеціальностей, що наведений у Таблиці 3.1, можна виділити ключові характеристики, які має бути передбачено та які я визначив для розробки в рамках цього проекту.

Також, під час порівняльного аналізу виявлено, що в функціоналі всіх трьох ресурсів відсутня можливість перегляду власного відгуку на обрану вакансію та завантаження власного резюме з відгуку. Крім того, розширений пошук вакансій присутній лише частково. На мою думку, ці можливості є важливими для пошуку робочих місць з IT спеціальностей, оскільки дозволяють легко порівняти вимоги різних вакансій, які зовні можуть бути схожими, і завантажити обраний відгук. Тому я вирішив включити ці функціональності до розробленого ресурсу.

3.1.3 Функціональні можливості проектованої системи

Планована програмна система включає реалізацію ключового функціоналу, який є популярним на платформах для пошуку роботи. Перелік функцій також враховує обрану тематику проекту, а саме ресурс з продажу виробів ручної роботи. Щоб виокремити функціональні можливості, які необхідно реалізувати на цьому ресурсі, було проведено аналіз кількох платформ зі схожою тематикою, таких як Jooble.com, Djinni.co і Dou.ua. На основі аналізу доступних можливостей цих платформ, а також виявлених недоліків, було сформульовано перелік функцій, які слід передбачити, і вони наведені у Таблиці 3.2.

Таблиця 3.2 – Функціонал програмної системи

Назва функціоналу	Опис функціоналу
Реєстрація	Створення нового облікового запису
Авторизація	Доступ до системи за допомогою

	існуючого облікового запису, створеного заздалегідь.
Каталог вакансій	Перегляд множини доступних вакансій на платформі
Перегляд вакансії	Перегляд картки окремої вибраної вакансії
Пошук вакансії	Можливість пошуку бажаної вакансії
Сортування та фільтрація пошукової системи з підбору вакансій	Реалізація розширеного пошуку товару
Збережені вакансії	Можливість користувача зберігати обрану вакансію
Перегляд збережених вакансій	Перегляд списку збережених вакансій
Відгук на вакансію	Можливість користувача надіслати свій відгук на обрану вакансію
Перегляд відгуків	Можливість користувача перегляду власного відгуку

У наведених таблицях перераховано основний функціонал веб-застосунку для пошуку робочих місць з IT спеціальностей, який є необхідним і вже реалізованим. Розроблені широкі можливості для зручного використання ресурсу. Під час проектування системи були враховані переваги характеристик аналогічних систем з пошуку робочих місць з IT спеціальностей з аналізу, а також виправлені їх недоліки.

3.2 Архітектурні рішення програмної системи пошуку роботи

3.2.1 Тип архітектури

Моя дипломна робота передбачала розробку системи з використанням клієнт-серверної архітектури. Проектована система є прикладом клієнт-серверної архітектури. Особливістю цього типу архітектури є наявність проміжного програмного забезпечення між клієнтською машиною та сервером даних. Всі дані та бізнес-логіка знаходяться в проміжному програмному забезпеченні. Використання такого проміжного програмного забезпечення підвищує гнучкість та продуктивність розробленої системи.

Архітектура клієнт-сервер є моделлю для проектування програмних систем, в якій клієнтські пристрої та серверні машини співпрацюють для виконання запитів користувачів. В цій архітектурі клієнти, такі як персональні комп'ютери або мобільні пристрої, ініціюють з'єднання, надсилаючи запити до централізованого сервера. Сервер обробляє ці запити, здійснює необхідні обчислення або пошук даних і повертає запитану інформацію клієнтам.

Веб-система з клієнт серверною архітектурою поділяється на декілька частин, включаючи рівень клієнтського представлення, рівень бізнес логіки, що включає в собі підключення до сховища даних.

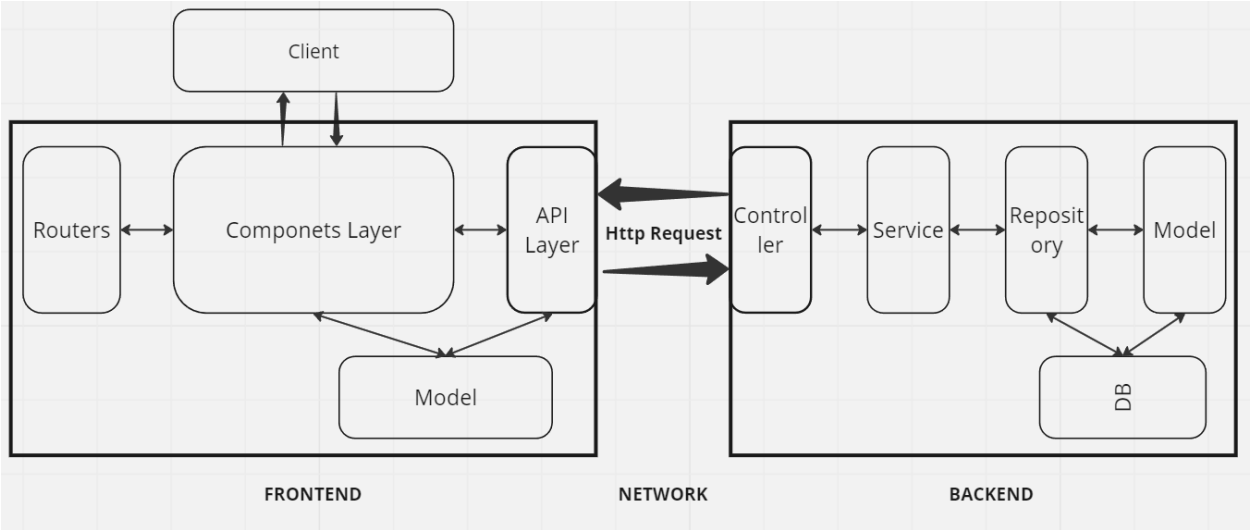

Рисунок 3.1 – Приклад клієнт-серверної архітектури

З рисунку 3.1 можна побачити що клієнтська частина архітектури відповідає за користувацький інтерфейс, збір даних, введених користувачем, і передачу запитів на сервер. Зазвичай вона складається з додатків або веб-браузерів, які взаємодіють з сервером для отримання бажаних даних або послуг.

Зі свого боку, серверна частина займається зберіганням, обробкою та керуванням даними і ресурсами. Вона приймає клієнтські запити, виконує необхідні операції, звертається до баз даних або зовнішніх сервісів і надсилає результати клієнтам.

Архітектура клієнт-сервер дозволяє створювати масштабовані та розподілені системи, оскільки декілька клієнтів можуть підключатися до одного сервера або кластера серверів. Вона також забезпечує централізований контроль та керування ресурсами, захист даних, а також полегшує обслуговування та оновлення.

3.3 Проектування, кодування, реалізація веб-застосунку для пошуку робочих місць з IT спеціальностей

3.3.1 Опис структури програми

Шаблон дизайну системи проекту базується на принципах моделі MVC (Model View Controller - Модель Представлення Контролер). Він розбиває додаток на модель даних, зовнішній інтерфейс та логіку управління. Цей шаблон дозволяє легко модифікувати та повторно використовувати модулі програми. Структура системи включає модель, представлення та контролер. Модель містить дані, представлення візуалізує ці дані, а контролер працює між ними, обробляючи події та забезпечуючи відповідну реакцію. Контролер обробляє вхідні запити та забезпечує взаємодію з моделлю.

Модель представляє дані та бізнес-логіку додатку, включаючи структуру даних, операції з базою даних та обчислення. Вона забезпечує цілісність та узгодженість даних.

Представлення представляє дані моделі в інтерфейсі користувача, визначаючи, як інформація відображається і взаємодіє з користувачем. Він може мати різні форми, наприклад, веб-сторінки або графічні інтерфейси.

Контролер діє як посередник між моделлю і представленням. Він керує введенням даних користувачем, визначає відповіді та оновлює модель на основі введених даних. Він також оновлює подання найновішими даними з моделі, забезпечуючи актуальність користувацького інтерфейсу.

Відповідно до принципів даної технології, ми отримали структуру системи, яка описана в Таблиці 3.3. Вона включає основні папки та файли, що містять важливу інформацію про додаток.

Таблиця 3.3 – Системні папки та файли програми

Common	Містить набір класів для реалізації часто зустрічаємо логіки застосунку
Config	Містить набір конфігураційних класів що реалізують підключення

Продовження таблиці 3.3

	до API та налаштування комунікації серверів
Controller	Містить набір класів контролерів що переплюють API запити
Dto	Містить набір класів що реалізують вигляд та зміст сутностей застосунку
Enums	Містить класи що перелічують статуси, категорії, ролі

Exception	Містить набір класів що обробляють зарезервовані помилки
Model	Містить набір класів що представляють сутності додатку
Repository	Містить набір класів що взаємодіють з базою даних
Service	Містить набір класів що реалізують бізнес логіку додатку
Application.properties	Файл що містить налаштування та залежності додатку
ItJobFinderApplicationTests.java	Файл з якого запускаються тести застосунку

3.3.2 Структура бази даних

Для реалізації бази даних стек технологій, що базується на використанні реляційної бази даних, яка була описана у розділі 2 даної роботи. Відповідно до реляційних принципів, у базі даних заздалегідь визначені зв'язки між її елементами. Елементи упорядковані у вигляді таблиць зі стовпцями та рядками, що представляють об'єкти або сутності, використовувані в системі. Стовпці таблиці мають визначений тип даних, а поле зберігає фактичне значення атрибуту. Рядки в таблиці відображають набір пов'язаних значень одного об'єкта або сутності. Кожен рядок таблиці має унікальний ідентифікатор (PK - primary key), що називається первинним ключем, а зв'язок між кількома таблицями встановлюється за допомогою зовнішніх ключів (FK - foreign key). Доступ до цих даних здійснюється за допомогою запитів на мові SQL.

Для зберігання даних була обрана база даних PostgreSql та програмний застосунок pgAdmin 4. PostgreSQL, яку зазвичай називають Postgres, є безкоштовною системою керування реляційною базою даних із відкритим

кодом. PostgreSQL робить сильний акцент на розширюваності та дотриманні стандартів SQL.

Робота з підключення серверу до бази даних та отримання, маніпулювання збереження даних реалізована через Java Persistence API, - це специфікація Java, яка надає стандартизований набір API та інструкцій для управління реляційними даними в додатках на Java. Це фреймворк об'єктно-реляційного відображення (ORM), який дозволяє розробникам відображати об'єкти Java в таблиці реляційних баз даних і виконувати операції з базами даних, використовуючи концепції об'єктно-орієнтованого програмування.

Діаграма бази даних відображає основні сутності системи у вигляді таблиць та зв’язки між ними за допомогою ключів на рисунку 3.2.

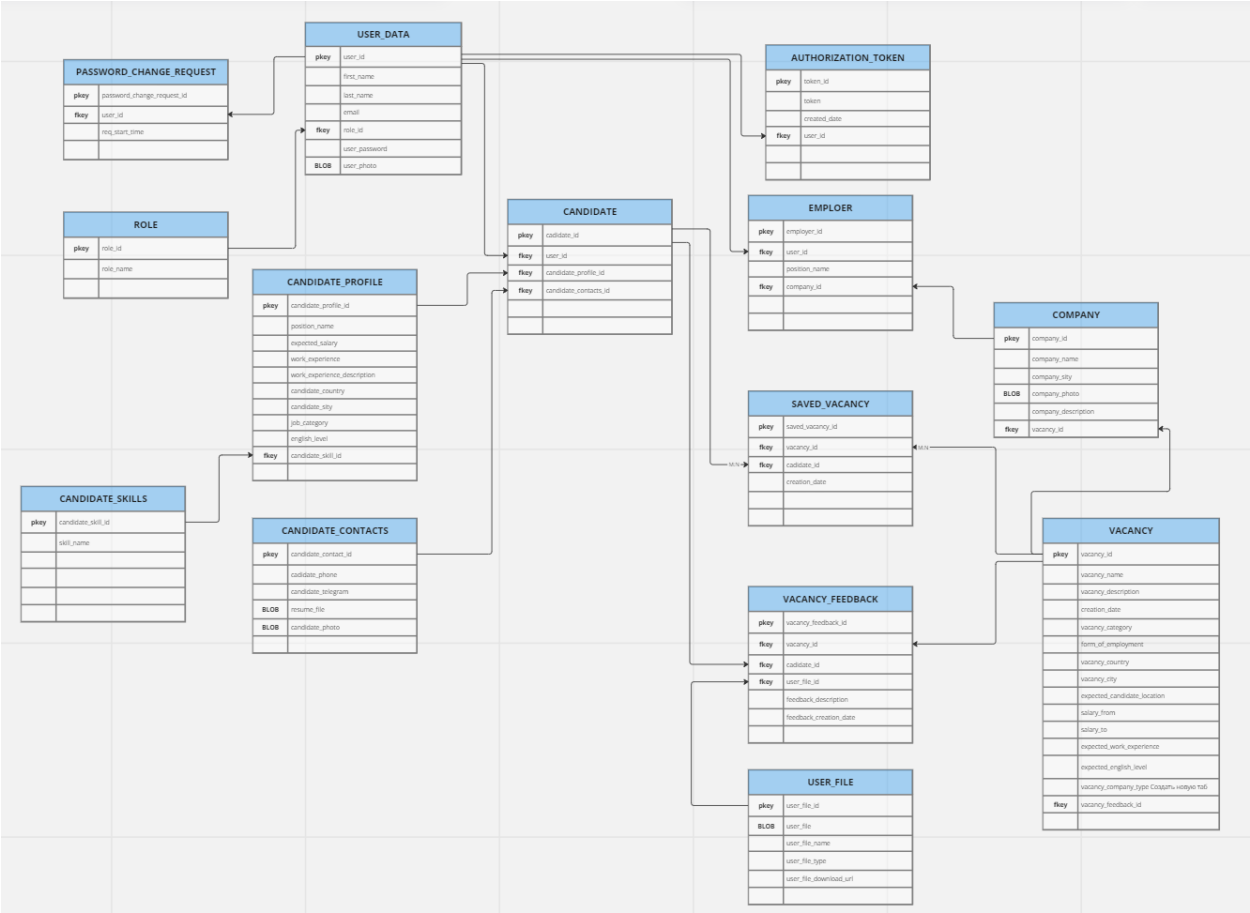

Рисунок 3.2 – Структура бази даних

Веб-застосунок за пошуку роботи з IT спеціальностей передбачає наявність таблиць, що надають можливість користувачу переглядати

вакансії, зберігати вакансії та відправляти резюме файл. База даних представлена такими таблицями: дані користувача, кандидат, вакансія, збережені вакансії, відгук на вакансію, файли користувача.

3.3.3 Структура публічної частини веб-застосунку

Для реалізації публічної частини веб-застосунку був обраний стек технологій, що базується на використанні фреймворку React.js, які були описані у розділі 2 даної роботи. Структура інтерфейсу веб-додатку на основі React.js зазвичай має компонентну архітектуру. React.js - це популярна бібліотека JavaScript для побудови користувацьких інтерфейсів, яка пропагує модульний та багаторазовий підхід до розробки інтерфейсу.

Застосунки на React.js будуються з використанням компонентів, які є багаторазовими та самодостатніми одиницями інтерфейсу користувача. Компоненти можна розділити на дві основні категорії:

- Функціональні компоненти. Це компоненти без стану, які визначені як функції і отримують дані через прокси. Вони в першу чергу відповідають за рендеринг елементів інтерфейсу і можуть бути простими і складними залежно від функціональності, яку вони надають.
- Компоненти класу. Це компоненти зі станом, які визначені як класи ES6 і розширюють клас React.Component. Вони мають власний внутрішній стан і можуть обробляти методи життєвого циклу. Класові компоненти використовуються, коли потрібне більш складне управління станами або інтерактивність.

Компоненти в React.js додатку структуровані в ієрархічний спосіб. Вони організовані на основі їх взаємозв'язку та рівнів вкладеності. Батьківський компонент може містити декілька дочірніх компонентів, а ці компоненти можуть мати власні дочірні компоненти. Ця ієрархія допомагає розділити інтерфейс на менші, керовані частини.

React.js має вбудовану систему управління станами. Стан представляє дані, які можуть змінюватися з часом, наприклад, дані, введені користувачем,

або відповіді API. Керувати станом можна в межах окремих компонентів за допомогою хука useState для функціональних компонентів або методу setState для компонентів класу. Крім того, для великих додатків можна використовувати складні бібліотеки управління станами, такі як Redux або MobX.

Застосунки на React.js часто потребують декількох сторінок або представлень. Для управління навігацією між цими сторінками зазвичай використовується бібліотека маршрутизації, така як React Router. React Router дозволяє визначати маршрути та рендерити різні компоненти на основі поточної URL-адреси.

React.js дозволяє використовувати різні підходи до стилізації. CSS можна застосовувати безпосередньо до компонентів, використовуючи вбудовані стилі або модулі CSS. Крім того, для стилізації компонентів можна використовувати популярні бібліотеки CSS-in-JS, такі як styled-components або CSS-препроцесори, такі як SASS або LESS.

Застосунки на React.js часто потребують отримання даних з API. Це можна зробити за допомогою вбудованого API вибірки JavaScript або за допомогою зовнішніх бібліотек, таких як Axios або хук useEffect в React, щоб робити асинхронні запити і обробляти процес вибірки даних.

Застосунки на React.js зазвичай збираються та оптимізуються для виробництва за допомогою інструментів збірки, таких як webpack або create-react-app. Ці інструменти об'єднують JavaScript, CSS та інші ресурси в оптимізовані файли, які можна розгорнути на веб-сервері або в мережі доставки контенту (CDN).

3.3.4 Структура бізнес-логіки веб-застосунку

Для реалізації публічної частини бізнес-логіки був обраний стек технологій, що базується на використанні фреймворку Spring Boot, які були описані у розділі 2 даної роботи. Внутрішня структура веб-додатку на основі Spring Boot та JPA (Java Persistence API) має багаторівневу архітектуру, широко відому як паттерн Model-View-Controller (MVC).

Першим чином створюються класи сутностей майбутнього застосунку Класи представляють бізнес-сутності або об'єкти домену у додатку. Вони відображаються на таблиці бази даних і визначають структуру даних та взаємозв'язки. Також стовоюються інтерфейси, що визначають операції CRUD (створення, читання, оновлення, видалення) та інші користувацькі запити для взаємодії з базою даних. Вони розширюють інтерфейс JpaRepository, що надається Spring Data JPA.

Сервісний рівень містять класи, що обробляють бізнес-логіку застосунку. Вони взаємодіють зі сховищами для виконання операцій з даними та інкапсулюють складні операції або робочі процеси. Для передачі даних між фронтендом і бекендом використовуються об'єкти передачі даних DTO, які містять лише необхідні дані для конкретного випадку використання, що допомагає мінімізувати мережевий трафік і підвищити продуктивність.

Рівень контролерів складається класів, які отримують і обробляють HTTP-запити від фронтенду. Вони містять методи, анотовані відповідними відображеннями запитів (наприклад, @GetMapping, @PostMapping), і взаємодіють з сервісним рівнем для обробки запитів і повернення відповідних відповідей. Створюються DTO, які визначають структуру даних, отриманих у запитах, і структуру даних, надісланих у відповідях. Вони допомагають перевіряти вхідні дані і забезпечують узгоджений інтерфейс API.

Важливим аспектом є налагодження конфігурацій проекту. Spring Boot надає централізований механізм конфігурації, який зазвичай виконується за допомогою анотацій, таких як @Configuration, @Bean та application.properties файли. Файли конфігурації визначають такі властивості, як деталі підключення до бази даних, налаштування безпеки, ведення журналів тощо. Конфігурація JPA необхідна для визначення з'єднання з базою даних, відображення сутностей та інших параметрів, пов'язаних з персистентністю.

Це можна зробити за допомогою анотацій, конфігурації XML або комбінації обох способів.

Створювалися валідатори винятків використовуються для перехоплення та обробки винятків, що виникають під час обробки запитів. Вони забезпечують централізований механізм обробки помилок і повернення клієнту відповідних відповідей на помилки.

3.3.5 Тестування програмної системи

Додатковим програмним забезпеченням для тестування запитів між серверами використовувався Swagger, оскільки він дозволяє визначати різні аспекти свого API, такі як кінцеві точки, корисне навантаження запиту/відповіді, методи автентифікації, обробку помилок тощо, використовуючи формат, зрозумілий машинам. Swagger, як стандартизована специфікація слугує одночасно і договором, і документацією для API, полегшуючи безперешкодно взаємодію і розуміння як людьми, так і машинами.

Щоб полегшити роботу зі специфікацією OpenAPI, Swagger пропонує набір інструментів. Серед цих інструментів важливу роль відіграє Swagger UI, який автоматично генерує інтерактивний інтерфейс документації на основі специфікації API. Цей інтуїтивно зрозумілий інтерфейс дозволяє користувачам без особливих зусиль досліджувати і тестувати кінцеві точки API, вивчати зразки запитів і відповідей, а також отримувати повне уявлення про доступні операції і параметри.

Були розроблені основні API для виконання бізнес-логіки веб-застосунку для пошуку роботи з IT на рисунку 3.3.

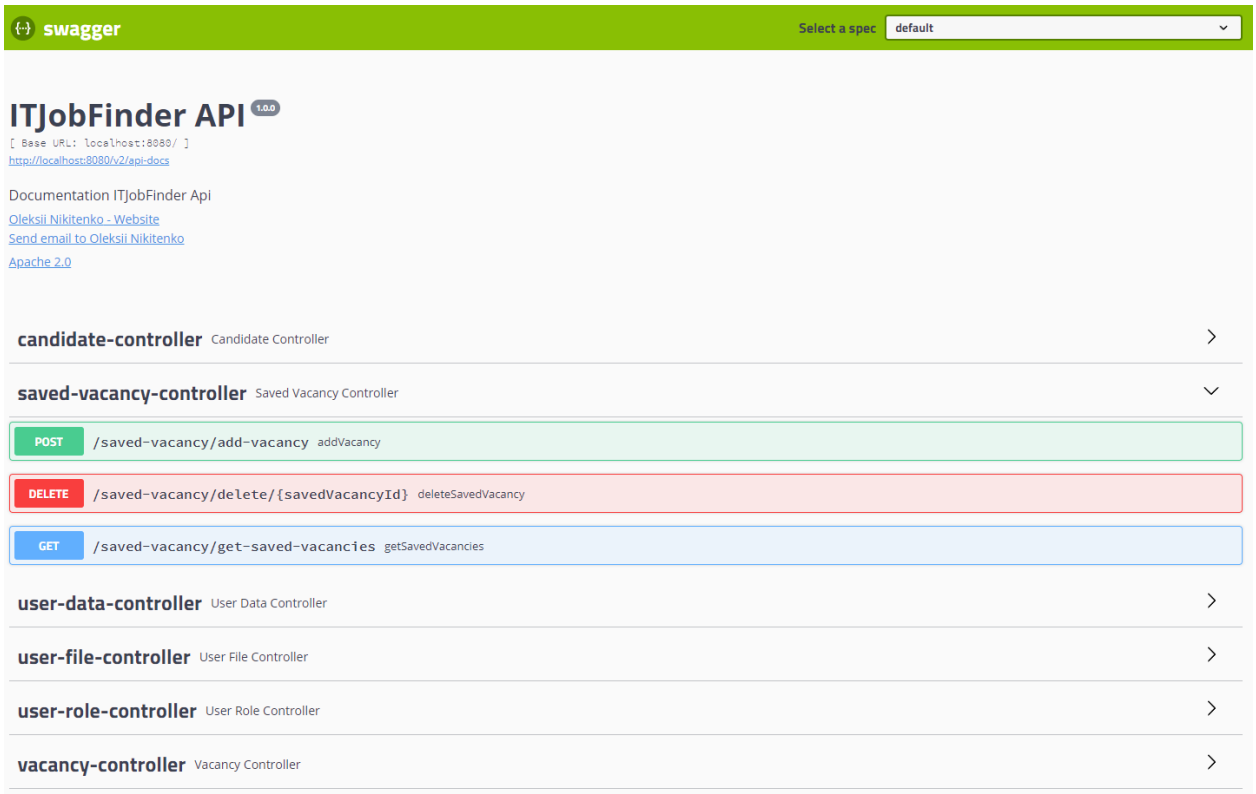

Рисунок 3.3 – Головна сторінка Swagger

Веб-застосунок з пошуку роботи з IT спеціальностей передбачає наявність API запитів, що надають можливість користувачу переглядати вакансії, зберігати вакансії та відправляти резюме файл. Основні контролери представлені таким набором API колів: реєстрація користувача, авторизація створення кандидата, створення вакансія, створення збереженої вакансії, створення відгуку на вакансію, завантаження файлів користувача.

Процес тестування API запиту проводится з формування запиту до серверної частини що виконує публічна частина застосунку що потребує дані та статус даних від серверу. У ролі публічної частина застосунку що потребує даних у ході тестування виступає розробник. Розробник використовує API запити, включаючи необхідні параметри для успішного виконання API запитів.

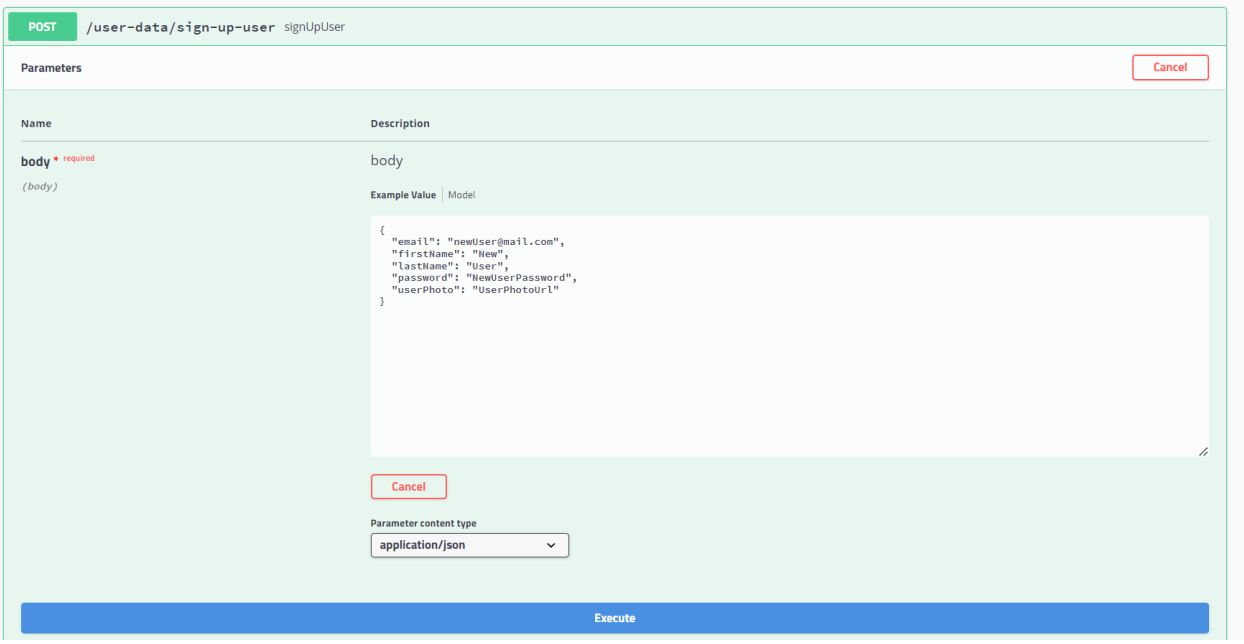

Рисунок 3.4 – Приклад внесення параметрів для створення реєстрації юзера

Після надсилання запиту до сервера сервер надсилає відповідь яку публічна частина аналізує та видає результат клієнту. Залежності від змісту відповіді сервера аналізується статус та коректна відповідь від сервера.

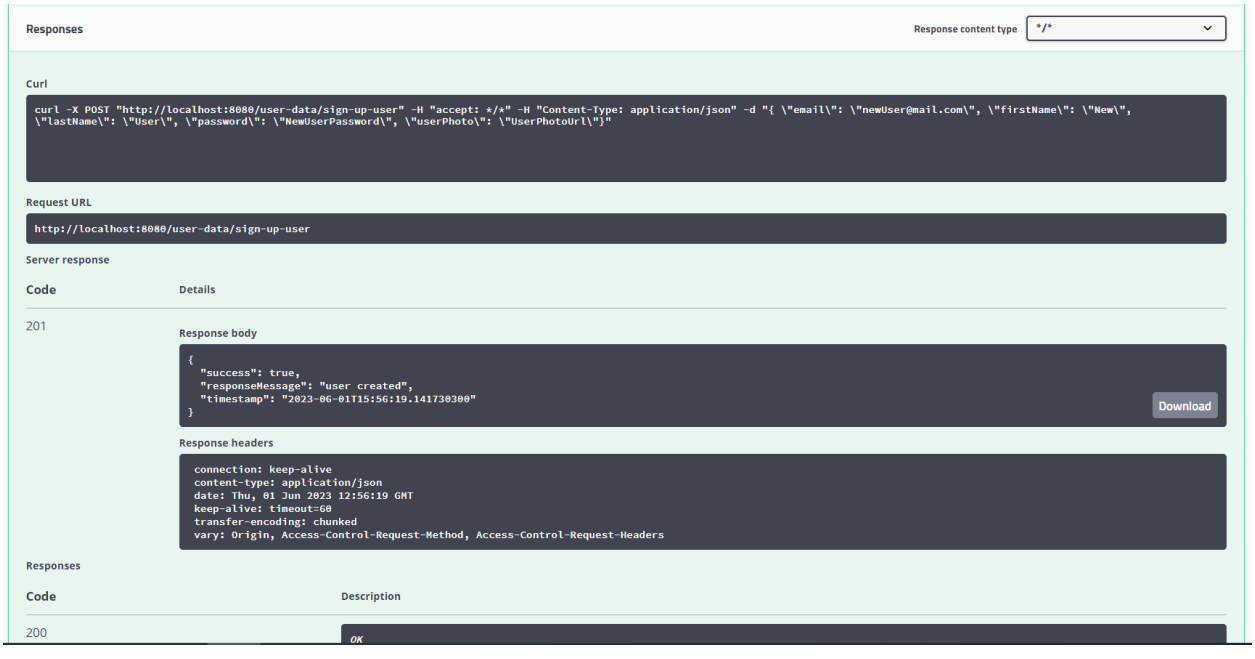

Рисунок 3.6 – Приклад отримання успішної відповіді від сервера

Після отримання помилки або зарезервованої помилки від сервера публічна частина застосунку обробляє зарезервовану помилку та сповіщає про це юзера. При отриманні не очікуваної помилки розробник виконує процес відлагодження помилки та зміни програмного коду.

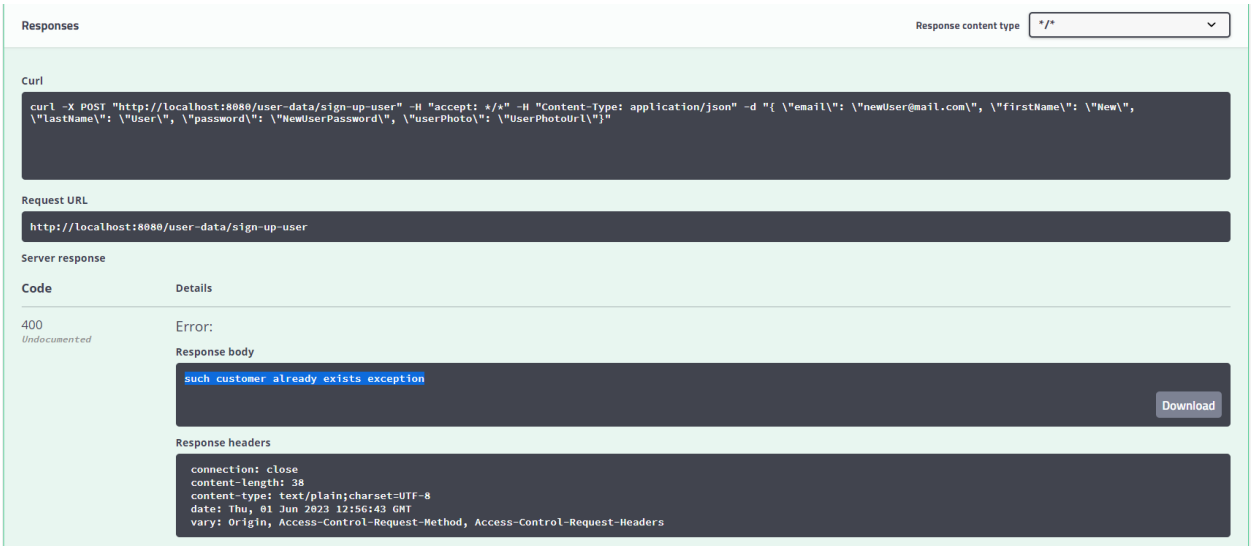

Рисунок 3.7– Приклад отримання помилки від сервера

Отже, у ході опису реалізації веб-застосунку з пошуку роботи з IT спеціальностей, було розроблено структуру веб застосунку, включаючи базу даних та модулі програми, візуальне представлення програми та бізнес-логіка веб-застосунку, описано принципи побудови сучасних веб застосунків та описати структуру програмного застосунку, висвітлений та доповнений опис детальними рисунками діяльності системи, рисунку побудови системи з використанням та схема бази даних.

ВИСНОВОК

Таким чином, у результаті виконання кваліфікаційної роботи бакалавра були проведені дослідження, на основі яких розроблено веб-застосунок з пошуку роботи з ІТ спеціальностей, зокрема:

- досліджено функціональні можливості та особливості побудови веб-застосунків для пошуку робочих місць з ІТ спеціальностей;
- здійснено аналіз програмно-технологічних рішень побудови веб-застосунків з пошуку роботи з ІТ спеціальностей;
- спроектовано, реалізовано, впроваджено веб-застосунок з пошуку роботи з ІТ спеціальностей з урахуванням технічних вимог;

А саме, були досліджені основні поняття з пошуку роботи, процеси пошуку роботи, основні документи для подачі заявки на працевлаштування та процес співбесіди на роботу. З’ясовані поняття веб-застосунку для пошуку роботи. Вивчені критерії унікальності веб-застосунків пошуку роботи з ІТ

спеціальностей. Переглянуті та засвоєні основні виклики при створенні веб-застосунку для пошуку роботи IT спеціальностей.

Були розглянуті основні кроки створення веб-застосунку для пошуку роботи. Проведено маркетингового дослідження ринку та вибір бізнес-моделі, включаючи: вибір обов'язкових функцій для веб-застосунку з пошуку роботи, вибір особливих функцій для веб-застосунку з пошуку роботи, вибір підходу до розробки. Озвучені основні етапи розробки з веб-застосунку дошки оголошень про вакансії. Розглянуто технологічний стек для розробки додатків для веб-застосунків з працевлаштування. Засвоєні важливі аспекти у створенні веб-застосунку для пошуку робочих місць з IT спеціальностей. Вивчені аспекти розробки бізнес-логіки веб-застосунку для пошуку робочих місць з IT спеціальностей, включаючи: Java мова програмування для створення бізнес-логіки веб-застосунку, Фреймворк Spring для створення бізнес-логіки веб-застосунку, Принцип інверсії контролю у Spring, Фреймворк що розширює Spring, Різниця Spring Boot та Spring Framework. Проаналізовані аспекти розробки система управління базами даних веб-застосунку для пошуку робочих місць з IT спеціальностей, що складаються з: Визначення баз даних та Типи баз даних, База даних що використовувалась для створення веб-застосунку для пошуку робочих місць з IT спеціальностей. Оглянуті аспекти розробки публічної частини веб-застосунку для пошуку робочих місць з IT спеціальностей. Переглянутий стек технологій для реалізації публічної частини веб-застосунку.

Розроблена структура веб застосунку, включаючи базу даних та модулі програми, візуальне представлення програми та бізнес-логіка веб-застосунку. Робота доповнена детальними рисунками діяльності системи, рисунку побудови системи з використанням та схема бази даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мінаєва В.І., Нікітенко О.О., Ващіліна О.В. Використання технології API для веб-застосунків. Прикладні інформаційні системи та технології в інформаційному суспільстві. Збірник тез VI Міжнародної науково-практичної конференції 30 вересня 2022 року, 2022, С. 167-172
2. How to Build a Job Search Website [Електронний ресурс] // syndicode - URL: https://syndicode.com/blog/how-to-build-a-job-search-website/ (дата звернення: 16.02.2023)
3. How Much Does It Cost to Build a Website Like Indeed [Електронний ресурс] // codica - URL: https://www.codica.com/blog/how-to-create-a-job-search-website-like-indeed-3-fast-solutions/ (дата звернення: 16.02.2023)
4. How to Build a Business App like Indeed Job Search? [Електронний ресурс] // ideausher - URL: https://ideausher.com/blog/how-to-build-a-business-app-like-indeed-job-search/ (дата звернення: 17.02.2023)
5. Guide on How to Create a Job Board: Pro Tips, Core Features, and Cost [Електронний ресурс] // sloboda-studio - URL: https://sloboda-

studio.com/blog/how-to-build-a-job-search-website/ (дата звернення: 17.02.2023)
6. How to Create a Job Search Website? [Електронний ресурс] // devteam - URL: https://www.devteam.space/blog/how-to-create-a-job-board-website/ (дата звернення: 17.02.2023)
7. How to Create a Job Board on Your Website [Електронний ресурс] // indeed - URL: https://www.indeed.com/hire/c/info/how-to-create-a-job-board-on-your-website (дата звернення: 18.02.2023)
8. How to Create a Virtual Job Board And Monetize It – Step-by-Step [Електронний ресурс] // hostinger - URL: https://www.hostinger.com/tutorials/how-to-create-an-online-job-board (дата звернення: 18.02.2023)
9. How to Build a Job Search Website? [Електронний ресурс] // appmaster - URL: https://appmaster.io/blog/how-to-build-a-job-search-website (дата звернення: 18.02.2023)
10. What Technologies do You Use For Job Portal App Development In 2023? [Електронний ресурс] // developers - URL: https://www.developers.dev/tech-talk/technology/job-portal-app-development-technologies.html (дата звернення: 19.02.2023)
11. 3 Unique Tech Tools to Help Your Job Hunt [Електронний ресурс] // topresume - URL: https://www.topresume.com/career-advice/tech-tools-help-job-search (дата звернення: 19.02.2023)
12. What Technologies You Use for Job Portal App Development 2023? [Електронний ресурс] // cisin - URL: https://www.cisin.com/coffee-break/technology/job-portal-app-development-technologies.html (дата звернення: 19.02.2023)
13. 10 Best IT and Technology Job Search Websites [Електронний ресурс] // decideconsulting - URL: https://decideconsulting.com/10-best-it-and-technology-job-search-websites/ (дата звернення: 20.02.2023)

14. How To Build A Website That Helps Job Hunters [Електронний ресурс] // ukrecruiter - URL: https://ukrecruiter.co.uk/2019/09/09/how-to-build-a-website-that-helps-job-hunters/ (дата звернення: 20.02.2023)
15. How to Create a Job Board Website Like Glassdoor [Електронний ресурс] // themindstudios - URL: https://themindstudios.com/blog/create-a-job-board-website/ (дата звернення: 20.02.2023)
16. How to Protect Your Website from Hackers? [Електронний ресурс] // hacken - URL: https://hacken.io/discover/how-to-protect-website-from-hackers-prevent-website-from-hacking/ (дата звернення: 21.02.2023)
17. How Search Engine Algorithms Work: Everything You Need to Know [Електронний ресурс] // searchenginejournal -URL: https://www.searchenginejournal.com/search-engines/algorithms/ (дата звернення: 20.02.2023)
18. Search Engine Algorithms: What You Need to Know [Електронний ресурс] // hawksem - URL: https://hawksem.com/blog/search-engine-algorithms-what-to-know/ (дата звернення: 20.02.2023)
19. What Are Core Web Vitals? [Електронний ресурс] // hawksem - URL: https://hawksem.com/blog/what-are-core-web-vitals/ (дата звернення: 22.02.2023)
20. Java (programming language) [Електронний ресурс] // wikipedia - URL: https://en.wikipedia.org/wiki/Java_(programming_language) (дата звернення: 22.02.2023)
21. Spring Framework [Електронний ресурс] // wikipedia - URL: https://en.wikipedia.org/wiki/Spring_Framework (дата звернення: 23.02.2023)
22. Spring Framework Overview [Електронний ресурс] // spring - URL: https://docs.spring.io/spring-framework/reference/overview.html (дата звернення: 25.02.2023)
23. Spring Framework release v6.0.8 [Електронний ресурс] // github - URL: https://github.com/spring-projects/spring-framework/releases/tag/v6.0.8 (дата звернення: 26.02.2023)

24. Spring Framework 1.0 Final Released [Електронний ресурс] // spring - URL: https://spring.io/blog/2004/03/24/spring-framework-1-0-final-released (дата звернення: 26.02.2023)
25. The tradition continues with the 16th Annual Jolt Awards, with new entries and new categories [Електронний ресурс] // drdobbs - URL: https://www.drdobbs.com/joltawards/2006-jolt-awards/187900423?pgno=10 (дата звернення: 26.02.2023)
26. Die Gewinner des JAX Innovation Award [Електронний ресурс] // archive - URL:https://web.archive.org/web/20090817202514/http://jax-award.de/jax_award06/gewinner_de.php (дата звернення: 26.02.2023)
27. Spring Framework 3.2.5 release [Електронний ресурс] // spring - URL:https://spring.io/blog/2013/11/07/spring-framework-3-2-5-released (дата звернення: 27.02.2023)
28. Announcing Spring Framework 4.0 GA Release [Електронний ресурс] // spring - URL: https://spring.io/blog/2013/12/12/announcing-spring-framework-4-0-ga-release (дата звернення: 27.02.2023)
29. Spring Framework v1.0.0.RELEASE [Електронний ресурс] // github - URL: https://github.com/spring-projects/spring-boot/releases/tag/v1.0.0.RELEASE (дата звернення: 28.02.2023)
30. Spring Framework 4.2 goes GA [Електронний ресурс] // spring - URL: https://spring.io/blog/2015/07/31/spring-framework-4-2-goes-ga (дата звернення: 28.02.2023)
31. Upgrading to Spring Framework 5.x [Електронний ресурс] // github - URL:https://github.com/spring-projects/spring-framework/wiki/Upgrading-to-Spring-Framework-5.x (дата звернення: 28.02.2023)
32. Reactive Spring [Електронний ресурс] // spring - URL:https://spring.io/blog/2016/02/09/reactive-spring (дата звернення: 28.02.2023)

33. Spring Framework 6.0 goes GA [Електронний ресурс] // spring - URL: https://spring.io/blog/2022/11/16/spring-framework-6-0-goes-ga (дата звернення: 28.02.2023)
34. Intro to Inversion of Control and Dependency Injection with Spring [Електронний ресурс] // baeldung - URL: https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring (дата звернення: 10.03.2023)
35. Legacy forums will be shutdown February 28 [Електронний ресурс] // spring - URL: https://spring.io/blog/2019/02/06/legacy-forums-will-be-shutdown-february-28 (дата звернення: 11.03.2023)
36. Is Spring between the devil and the EJB? [Електронний ресурс] // andygibson - URL: http://www.andygibson.net/blog/article/is-spring-between-the-devil-and-the-ejb/ (дата звернення: 12.03.2023)
37. Ресурс Spring.io [Електронний ресурс] // spring - URL: https://spring.io/ (дата звернення: 12.03.2023)
38. Spring Boot 3.1.0 [Електронний ресурс] // spring - URL: https://spring.io/projects/spring-boot (дата звернення: 12.03.2023)
39. What is Java Spring Boot? [Електронний ресурс] // ibm - URL: https://www.ibm.com/topics/java-spring-boot#:~:text=Java%20Spring%20Boot%20(Spring%20Boot,Autoconfiguration (дата звернення: 12.03.2023)
40. What is Spring Boot? [Електронний ресурс] // educative - URL: https://www.educative.io/answers/what-is-spring-boot (дата звернення: 14.03.2023)
41. Encyclopedia of Database Systems [Електронний ресурс] // springer - URL: https://link.springer.com/referencework/10.1007/978-0-387-39940-9 (дата звернення: 15.03.2023)
42. Database Management Systems by Raghu Ramakrishnan and Johannes Gehrke [Електронний посібник] // wisc - URL: https://pages.cs.wisc.edu/~dbbook/ (дата звернення: 16.03.2023)

43. Database System Concepts Seventh Edition by Avi Silberschatz, Henry F. Korth, S. Sudarshan [Електронний посібник] // db-book - URL: https://www.db-book.com/ (дата звернення: 16.03.2023)
44. IBM Information Management System (IMS) 13 Transaction and Database Servers delivers high performance and low total cost of ownership [Електронний ресурс] // ibm - URL: https://www.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&appname=iSource&supplier=897&letternum=ENUS213-381 (дата звернення: 16.03.2023)
45. PostgreSQL: The World's Most Advanced Open Source Relational Database [Електронний ресурс] // postgresql - URL: https://www.postgresql.org/ (дата звернення: 16.03.2023)
46. PostgreSQL 15.3, 14.8, 13.11, 12.15, and 11.20 Released! [Електронний ресурс] // postgresql - Режим доступу до ресурсу https://www.postgresql.org/about/news/postgresql-153-148-1311-1215-and-1120-released-2637/ (дата звернення: 16.03.2023)
47. PostgreSQL is a powerful database system with a strong reputation for reliability, data integrity, and correctness. [Електронний ресурс] // directory - URL: https://directory.fsf.org/wiki/PostgreSQL (дата звернення: 16.03.2023)
48. What is PostgreSQL? How is it pronounced? What is Postgres? [Електронний ресурс] // postgresql - URL: https://wiki.postgresql.org/wiki/FAQ#What_is_PostgreSQL.3F_How_is_it_pronounced.3F_What_is_Postgres.3F (дата звернення: 16.03.2023)
49. Postgresql History [Електронний ресурс] // archive URL: https://web.archive.org/web/20170326020245/https://www.postgresql.org/about/history/# (дата звернення: 16.03.2023)
50. Java Database Connectivity [Електронний ресурс] // wikipedia - URL: https://en.wikipedia.org/wiki/Java_Database_Connectivity (дата звернення: 19.03.2023)

51. JDK 1.1 Offers Substantial AWT Improvements, Internationalization and Enterprise Features [Електронний ресурс] // archive - URL: https://web.archive.org/web/20080210044125/http://www.sun.com/smi/Press/sunflash/1997-02/sunflash.970219.0001.xml (дата звернення: 20.03.2023)
52. Java SE Technologies - Database [Електронний ресурс] // oracle - URL: https://www.oracle.com/java/technologies/javase/javase-tech-database.html#corespec40 (дата звернення: 01.04.2023)
53. Java Community Process [Електронний ресурс] // jcp - Режим доступу до ресурсу https://jcp.org/aboutJava/communityprocess/mrel/jsr221/index.html (дата звернення: 13.04.2023)
54. Jakarta Persistence [Електронний ресурс] // wikipedia - URL: https://en.wikipedia.org/wiki/Jakarta_Persistenc (дата звернення: 15.04.2023)
55. Jakarta Persistence 3.1 [Електронний ресурс] // jakarta - URL: https://jakarta.ee/specifications/persistence/3.1/ (дата звернення: 15.04.2023)
56. Bnadb [Електронний ресурс] // oracle - URL: https://docs.oracle.com/javaee/6/tutorial/doc/bnacj.html#bnadb (дата звернення: 16.04.2023)
57. Hibernate [Електронний ресурс] // jboss - URL: https://docs.jboss.org/hibernate/entitymanager/3.5/reference/en/html_single/#architecture-javase (дата звернення: 16.04.2023)
58. JavaScript [Електронний ресурс] // wikibooks - URL: https://en.wikibooks.org/wiki/JavaScript (дата звернення: 5.05.2023)
59. HTML [Електронний ресурс] // wikipedia URL: https://en.wikipedia.org/wiki/HTML (дата звернення: 16.05.2023)
60. HTML specifications. [Електронний ресурс] // w3 URL: https://www.w3.org/html/ (дата звернення: 16.05.2023)
61. 4.2 SGML [Електронний ресурс] // w3 - URL: https://www.w3.org/TR/REC-html40-971218/conform.html#deprecated (дата звернення: 16.05.2023)
62. CSS [Електронний ресурс] // wikipedia - URL: https://en.wikipedia.org/wiki/CSS (дата звернення: 16.05.2023)

63. Learn CSS [Електронний ресурс] // mozilla - URL: https://developer.mozilla.org/en-US/docs/Learn/CSS (дата звернення: 16.05.2023)
64. What is css [Електронний ресурс] // w3 URL: https://www.w3.org/standards/webdesign/htmlcss#whatcss (дата звернення: 16.05.2023)
65. Html5 css js in mobile apps [Електронний ресурс] // htmlgoodies - URL: https://www.htmlgoodies.com/webmaster/html5-css-js-mobile-apps/ (дата звернення: 16.05.2023)
66. React (software) [Електронний ресурс] // wikipedia URL: https://en.wikipedia.org/wiki/React_(software) (дата звернення: 16.05.2023)

ДОДАТКИ

ДОДАТОК А

```
@RestController
@RequestMapping("/vacancy")
public class VacancyController {

    @Autowired
    VacancyService vacancyService;

    @GetMapping("/all")
    public ResponseEntity<List<Vacancy>> getAllVacancies() {
        List<Vacancy> vacancyList = vacancyService.getAllVacancies();
        return new ResponseEntity<>(vacancyList, HttpStatus.OK);
    }

    @PostMapping("/create")
    public ResponseEntity<ApiResponse> createVacancy(@RequestBody VacancyDTO body) {
        vacancyService.createVacancy(body);
        return new ResponseEntity<>(new ApiResponse(true, "vacancy created"),
HttpStatus.CREATED);
    }

    @GetMapping("/get/{vacancyId}")
    public ResponseEntity<Vacancy> getVacancyById(@PathVariable("vacancyId") Long
vacancyId) {
        Vacancy vacancy = vacancyService.getVacancyById(vacancyId);
        return new ResponseEntity<>(vacancy, HttpStatus.OK);
    }

    @PutMapping("/update/{vacancyId}")
    public ResponseEntity<ApiResponse> updateVacancyById(@PathVariable("vacancyId") Long
vacancyId,
                                                         @RequestBody VacancyDTO body) {
        if (!vacancyService.findVacancyById(vacancyId)) {
            return new ResponseEntity<>(new ApiResponse(true, "vacancy not found"),
HttpStatus.BAD_REQUEST);
        }
        vacancyService.updateVacancy(vacancyId, body);
        return new ResponseEntity<>(new ApiResponse(true, "vacancy updated"),
HttpStatus.OK);
    }

    @DeleteMapping("/delete/{vacancyId}")
    public ResponseEntity<ApiResponse> deleteVacancyById(@PathVariable("vacancyId") Long
vacancyId) {
        vacancyService.deleteVacancyId(vacancyId);
        return new ResponseEntity<>(new ApiResponse(true, "vacancy deleted"),
HttpStatus.OK);
    }
}
```

Рисунок А1 – Контролер з реалізованим CRUD функціоналом для збереження обраної вакансії

```
spring.datasource.url=jdbc:postgresql://localhost:5433/itjobfinder
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.datasource.hikari.auto-commit=false

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update

spring.mvc.pathmatch.matching-strategy = ANT_PATH_MATCHER

spring.servlet.multipart.enabled=true
spring.servlet.multipart.file-size-threshold=1B
spring.servlet.multipart.max-file-size=50MB
spring.servlet.multipart.max-request-size=60MB
```

Рисунок 1Б – Параметри проекту для підключення бази даних

```
2 usages  Oleksii Nikitenko
@Repository
public interface SavedVacancyRepository extends JpaRepository<SavedVacancy, Long> {

    1 usage  Oleksii Nikitenko
    List<SavedVacancy> findAllByCandidateOrderByCreationDateDesc(Candidate candidate);

    Oleksii Nikitenko
    Optional<SavedVacancy> findSavedVacancyByVacancy(Vacancy Vacancy);
    1 usage  Oleksii Nikitenko
    Optional<SavedVacancy> findSavedVacancyByVacancyAndCandidate(Vacancy Vacancy, Candidate candidate);
}
```

Рисунок 2Б – Інтерфейс збережених вакансій для взаємодії з базою даних

ДОДАТОК В

```
useEffect(() => {
  fetch("http://localhost:8080/vacancy/all")
    .then((response) => response.json())
    .then((data) => setVacancies(data))
    .catch((error) => console.error(error));
}, []);
```

Рисунок 1В – Рядок надсилання запиту на бекенд для отримання списку всіх вакансій

```
<div className="vacancies-container">
  {paginatedVacancies.length > 0 ? (
    paginatedVacancies.map((vacancy) => (
      <div key={vacancy.vacancyId} className="vacancy-card">
        <div className="main-vac-info">
          <Link to={`/vacancies/${vacancy.vacancyId}`}>
            {vacancy.vacancyName}
          </Link>
          <h2>
            {vacancy.salaryFrom}$ - {vacancy.salaryTo}$
          </h2>
          <p> {formatDate(vacancy.creationDate)}</p>
        </div>
        <p>
          {showMoreId === vacancy.vacancyId
            ? vacancy.vacancyDescription
            : vacancy.vacancyDescription.slice(0, 254) + "..."}
        </p>
        {vacancy.vacancyDescription.length > 254 && (
          <a
            href="#"
            onClick={(e) => {
              e.preventDefault();
              toggleShowMore(vacancy.vacancyId);
            }}
          >
            {showMoreId === vacancy.vacancyId
              ? "Show less"
              : "Show more"}
          </a>
```

Рисунок 2В – Контейнер відображення вакансій отриманих від бекенду