ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «СИСТЕМА ПІДТРИМКИ РІШЕНЬ У СФЕРІ ЗАЙНЯТОСТІ НАСЕЛЕННЯ»

на здобуття освітнього ступеня бакалавра
зі спеціальності *124 Системний аналіз*
(код, найменування спеціальності)
освітньо-професійної програми *Системний аналіз*
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Тимофій ФЕДОРЧУК
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. САД-41
Тимофій ФЕДОРЧУК
Ім'я, ПРІЗВИЩЕ
Керівник: Ровіл НАФЄЄВ
к.ф.-м.н., доцент.. Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Бакалавр

Спеціальність 124 Системний аналіз

Освітньо-професійна програма Системний аналіз

ЗАТВЕРДЖУЮ
Завідувач кафедрою
інформаційних систем та технологій
_____Каміла СТОРЧАК
«____» _____2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Федорчук Тимофій Русланович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Система підтримки рішень у сфері зайнятості населення»

керівник кваліфікаційної роботи Нафєєв Ровіл Касимович к.ф.-м.н., доцент..
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026р. № 51

2. Строк подання кваліфікаційної роботи «01» червня 2026р.
3. Вихідні дані до кваліфікаційної роботи: інформаційно-аналітична система підтримки прийняття рішень у сфері зайнятості населення;
наукова та технічна література з інформаційних систем, веб-технологій, аналізу даних та систем підтримки прийняття рішень..
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. визначення потреб у створенні системи підтримки рішень у сфері зайнятості населення; формування функціональних вимог до інформаційно-аналітичної системи;
2. вибір та обґрунтування архітектури інформаційної системи; дослідження принципів збору, зберігання та обробки даних ринку праці;
3. розробка механізмів збору, обробки та аналізу інформації щодо вакансій, резюме та заявок; інтеграція інструментів аналітики та підтримки прийняття рішень.
5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «20» лютого 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Визначення потреб та вимог до системи підтримки рішень у сфері зайнятості населення. Формування цілей розробки та обґрунтування вибору технологічних рішень	24.02.2026 – 28.02.2026.	
2	Аналіз наукової та технічної літератури. Дослідження сучасних підходів до створення інформаційних систем зайнятості та систем підтримки прийняття рішень	01.03.2026 – 14.03.2026	
3	Огляд, порівняння та аналіз технологій і платформ для обробки даних і аналітики (веб-системи, інструменти візуалізації, засоби збору даних та аналітики ринку праці)	15.03.2026 – 28.03.2026	
4	Розробка рекомендацій щодо побудови ефективної інформаційно-аналітичної системи підтримки рішень у сфері зайнятості населення, забезпечення надійності, масштабованості та ефективності обробки даних	29.03.2026 – 11.04.2026	
5	Реалізація програмної частини: розробка та інтеграція компонентів системи (модулі користувачів, вакансій, заявок, аналітики); налаштування взаємодії між підсистемами	12.04.2026 – 25.04.2026	
6	Оформлення результатів дослідження: написання та структуризація кваліфікаційної роботи, підготовка графічних матеріалів	26.04.2026 – 12.05.2026	
7	Підготовка доповіді та презентаційних матеріалів до захисту	13.05.2026 – 21.05.2026	

Здобувач вищої освіти _____ Тимофій ФЕДОРЧУК

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Ровіл НАФЄЄВ

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина бакалаврської роботи: 57 сторінок, 27 рисунків, 10 таблиць, 30 джерел.

Об'єкт дослідження – інформаційні системи підтримки прийняття рішень у сфері зайнятості населення.

Предмет дослідження – методи та засоби проєктування і реалізації веб-системи для автоматизації процесів працевлаштування, аналізу даних ринку праці та підтримки прийняття управлінських рішень.

Мета роботи – розробка системи підтримки рішень у сфері зайнятості населення, яка забезпечує ефективну взаємодію між шукачами роботи, роботодавцями та адміністраторами, а також підвищує якість підбору кандидатів і вакансій.

Короткий зміст роботи. У кваліфікаційній роботі досліджено теоретичні та практичні аспекти розробки інформаційної системи підтримки прийняття рішень у сфері зайнятості населення. Проведено аналіз сучасних підходів до організації інформаційних процесів у системах працевлаштування, зокрема обробки даних щодо вакансій, резюме та заявок, а також використання аналітичних інструментів для підтримки управлінських рішень. Розглянуто можливості сучасних веб-технологій та інструментів аналізу даних, що застосовуються для побудови інформаційно-аналітичних систем.

У роботі виконано проєктування архітектури системи, визначено її основні компоненти, функціональні модулі та принципи взаємодії між користувачами різних ролей (працівники, роботодавці, адміністратори).

КЛЮЧОВІ СЛОВА: СИСТЕМА ПІДТРИМКИ РІШЕНЬ, ЗАЙНЯТІСТЬ НАСЕЛЕННЯ, ВЕБ-ДОДАТОК, NODE.JS, EXPRESS, SQLITE, JWT, ВАКАНСІЇ, РЕЗЮМЕ, ПРАЦЕВЛАШТУВАННЯ, АНАЛІТИКА ДАНИХ

ЗМІСТ

ВСТУП

Актуальність теми. Актуальність дослідження зумовлена зростанням ролі інформаційних технологій у сфері зайнятості населення та необхідністю підвищення ефективності прийняття управлінських рішень на основі даних. В умовах динамічних змін ринку праці, цифровізації економіки та збільшення обсягів інформації виникає потреба у створенні інтелектуальних систем, здатних забезпечити оперативний аналіз даних щодо вакансій, резюме та попиту на робочу силу.

Традиційні підходи до організації процесів працевлаштування не забезпечують достатнього рівня автоматизації, швидкості обробки інформації та аналітичної підтримки, що ускладнює ефективне управління ресурсами ринку праці. У зв’язку з цим розробка системи підтримки прийняття рішень у сфері зайнятості населення є актуальним завданням, оскільки дозволяє підвищити якість підбору кандидатів, оптимізувати взаємодію між учасниками процесу та забезпечити обґрунтованість управлінських рішень на основі актуальних даних.

Мета роботи – розробка системи підтримки рішень у сфері зайнятості населення, яка забезпечує ефективну взаємодію між шукачами роботи, роботодавцями та адміністраторами, а також підвищує якість підбору кандидатів і вакансій.

Для досягнення мети, у бакалаврській роботі успішно виконано наступні завдання:

– дослідження сучасних тенденцій розвитку та використання інформаційних систем у сфері зайнятості населення;

– огляд методів обробки та аналізу даних ринку праці, зокрема інформації про вакансії, резюме та заяви на працевлаштування;

– аналіз результатів впровадження інформаційно-аналітичних систем підтримки прийняття рішень, оцінка ефективності їх використання для підвищення якості підбору персоналу та оптимізації процесів працевлаштування.

Об'єкт дослідження – інформаційні системи підтримки прийняття рішень у сфері зайнятості населення.

Предмет дослідження – методи та засоби проєктування і реалізації веб-системи для автоматизації процесів працевлаштування, аналізу даних ринку праці та підтримки прийняття управлінських рішень.

Методи дослідження:

1. Аналіз наукових та технічних джерел – дослідження сучасних інформаційних систем у сфері зайнятості та підходів до підтримки прийняття рішень.

2. Методи системного аналізу та моделювання – визначення структури системи, ролей користувачів і бізнес-процесів.

3. Методи проєктування інформаційних систем – розробка архітектури клієнтської та серверної частин, бази даних і API.

4. Методи програмної реалізації – створення серверної частини на Node.js з використанням Express, розробка клієнтського інтерфейсу із застосуванням HTML, CSS, JavaScript.

Наукова новизна одержаних результатів. У ході дослідження запропоновано підхід до побудови інформаційної системи підтримки прийняття рішень у сфері зайнятості населення, який ґрунтується на комплексному використанні методів збору, обробки та аналітичного опрацювання даних ринку праці. Запропонована модель передбачає інтеграцію функціональних модулів роботи з вакансіями, резюме та заявками з аналітичними механізмами обробки інформації, що дозволяє підвищити обґрунтованість і оперативність прийняття управлінських рішень. Особливістю підходу є поєднання операційного обліку даних із їх подальшою аналітичною інтерпретацією у вигляді показників, статистики та візуалізацій.

Практична значущість одержаних результатів. Розроблена інформаційна система забезпечує автоматизацію основних процесів у сфері працевлаштування, зокрема обробку заявок, управління вакансіями та взаємодію між користувачами різних ролей. Система дозволяє здійснювати аналіз даних ринку праці, формувати

ключові показники ефективності та надавати користувачам зручні інструменти візуалізації інформації. Отримані результати можуть бути використані для підвищення ефективності управління процесами зайнятості населення, оптимізації підбору кандидатів і покращення якості прийняття рішень у реальних умовах функціонування інформаційних систем.

Апробація результатів та публікації. Основні положення і результати бакалаврської роботи доповідались на науково практичних конференціях, що проходили на базі Державного університету інформаційно-комунікаційних технологій. VII Міжнародну науково-технічну конференцію «Сучасний стан та перспективи розвитку IoT» на тему «Система підтримки рішень у сфері зайнятості населення».

1 ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМ ПІДТРИМКИ РІШЕНЬ У СФЕРІ ЗАЙНЯТОСТІ НАСЕЛЕННЯ

1.1 Поняття та роль систем підтримки рішень у сучасних інформаційних системах

Система підтримки прийняття рішень є різновидом інформаційної системи, призначеної для збору, обробки, аналізу та подання даних у такій формі, яка допомагає користувачу обрати найбільш доцільний варіант дій у конкретній ситуації. На відміну від звичайних інформаційних систем, що переважно забезпечують зберігання та відображення інформації, СППР орієнтована саме на інтелектуальну підтримку процесу прийняття рішень. Її основне призначення полягає у тому, щоб надати користувачеві не просто набір відомостей, а аналітично опрацьовану інформацію, яка може бути використана для оцінки поточної ситуації, порівняння альтернатив і вибору оптимального рішення [1].

Система підтримки прийняття рішень поєднує в собі базу даних, аналітичні методи, моделі обробки інформації та інтерфейс взаємодії з користувачем. Завдяки цьому вона дозволяє працювати з великими обсягами даних, виявляти закономірності, формувати прогнози та подавати результати у зручному для сприйняття вигляді. Важливою особливістю СППР є те, що вона не замінює людину у процесі ухвалення рішення, а лише підвищує його обґрунтованість, точність і оперативність за рахунок використання сучасних інформаційних технологій.

Системи підтримки прийняття рішень набули широкого застосування у різних галузях діяльності, де ефективність управління значною мірою залежить від якості обробки інформації та здатності оперативно реагувати на зміни зовнішнього середовища. Системи активно використовуються для фінансового аналізу, прогнозування попиту на продукцію чи послуги, оптимізації витрат, управління запасами та логістичними процесами. Завдяки застосуванню СППР

підприємства отримують можливість оцінювати різні сценарії розвитку, мінімізувати ризики та підвищувати конкурентоспроможність на ринку (рис. 1.1).

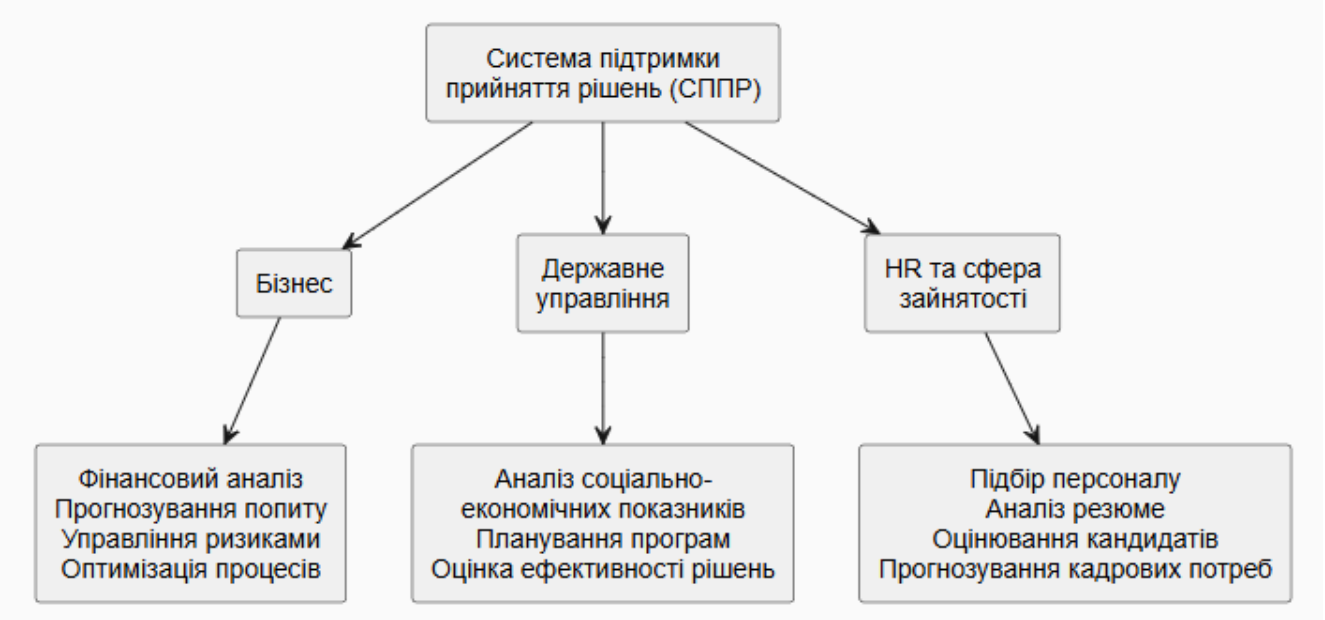

Рисунок 1.1 — Сфери застосування СППР

У сфері державного управління системи підтримки прийняття рішень відіграють не менш важливу роль, оскільки дозволяють обробляти значні обсяги статистичної та аналітичної інформації, що необхідна для формування державної політики та прийняття стратегічних рішень. СППР застосовуються для аналізу соціально-економічних показників, планування бюджету, прогнозування розвитку окремих галузей економіки, а також для оцінки ефективності реалізації державних програм. Особливе значення такі системи мають у сфері зайнятості населення, де необхідно враховувати динаміку ринку праці, рівень безробіття, попит на різні професії та регіональні особливості працевлаштування [2]. Використання СППР у цій сфері дозволяє органам державної влади приймати більш точні та обґрунтовані рішення щодо регулювання ринку праці, розробки програм підтримки населення та підвищення ефективності діяльності центрів зайнятості.

Окремого значення системи підтримки прийняття рішень набувають у сфері управління персоналом (HR), де процеси прийняття рішень пов’язані з великим обсягом інформації про кандидатів, вакансії та вимоги роботодавців. У цьому контексті СППР використовуються для автоматизації підбору персоналу, аналізу

резюме, оцінювання відповідності кандидатів встановленим критеріям, а також для прогнозування кадрових потреб організації. Завдяки використанню таких систем значно зменшується навантаження на рекрутерів, підвищується швидкість обробки інформації та покращується якість прийнятих рішень щодо найму. Крім того, СППР дозволяють враховувати різноманітні параметри, такі як професійні навички, досвід роботи, рівень освіти, що сприяє більш точному підбору персоналу та зменшенню ризику помилкових рішень. Для відображення ролі системи підтримки прийняття рішень у процесі обробки інформації та формування управлінських рішень доцільно представити узагальнену схему її функціонування (рис. 1.2).

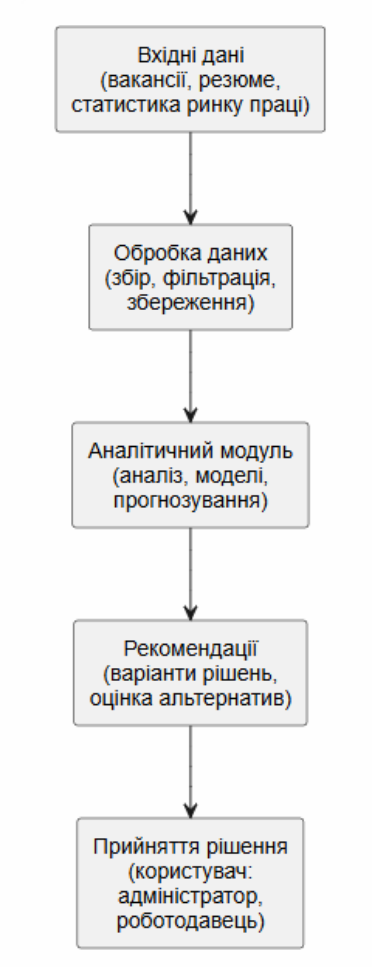

Рисунок 1.2 – Роль системи підтримки прийняття рішень у процесі формування управлінських рішень

Система підтримки прийняття рішень виконує послідовне перетворення первинних даних у рекомендації, що використовуються для прийняття управлінських рішень. На кожному етапі здійснюється обробка та аналітичне опрацювання інформації, що дозволяє зменшити невизначеність і підвищити обґрунтованість вибору [3]. Таким чином, СППР виступає ключовим інструментом, який забезпечує зв’язок між даними та кінцевим рішенням користувача.

Інформаційних системах системи підтримки прийняття рішень виконують ряд ключових функцій, що забезпечують ефективне використання даних та підвищення обґрунтованості управлінських рішень (рис. 1.3).

− аналіз даних, який передбачає збір, обробку та інтерпретацію інформації з різних джерел. У сфері зайнятості це може включати обробку даних про вакансії, резюме, рівень безробіття, попит на професії та інші показники ринку праці. Завдяки цьому користувач отримує структуровану та зрозумілу інформацію, що дозволяє оцінити поточну ситуацію.

− безпосередня підтримка процесу прийняття рішень. СППР формує рекомендації на основі проведеного аналізу, дозволяє порівнювати альтернативні варіанти та обирати найбільш доцільний. У контексті системи зайнятості це може проявлятися у підборі найбільш відповідних кандидатів до вакансій або визначенні оптимальних напрямів працевлаштування.

− прогнозування, забезпечує можливість передбачення майбутніх змін на основі наявних даних. Застосування методів прогнозування дозволяє визначати тенденції розвитку ринку праці, оцінювати попит на певні професії, а також планувати кадрові потреби. Це особливо важливо в умовах динамічних змін економіки та швидкого розвитку технологій.

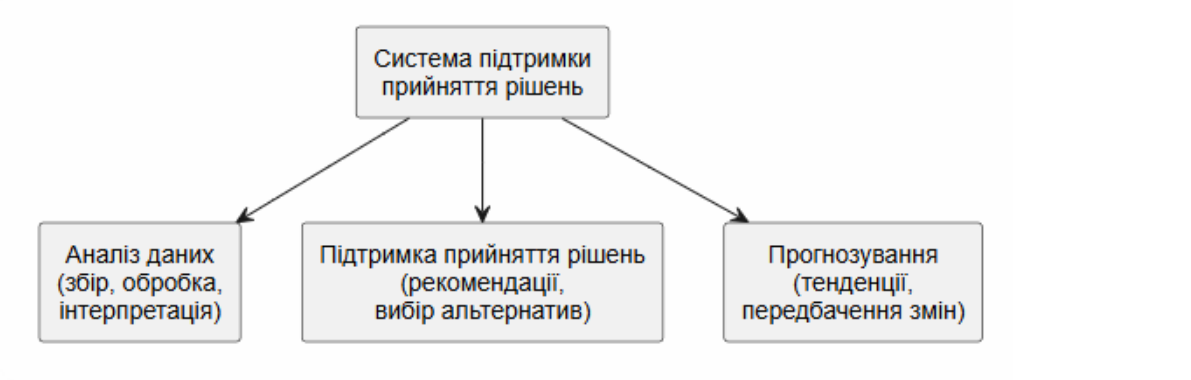

Рисунок 1.3 – Основні функції системи підтримки прийняття рішень

Система підтримки прийняття рішень реалізує три ключові функції: аналіз даних, формування рекомендацій та прогнозування. Їх поєднання забезпечує комплексний підхід до обробки інформації та дозволяє підвищити ефективність прийняття управлінських рішень.

Системи підтримки прийняття рішень забезпечує ефективне використання даних для обґрунтування управлінських рішень. Вони поєднують у собі функції аналізу інформації, формування рекомендацій та прогнозування, що дозволяє підвищити точність, швидкість і якість прийняття рішень у різних сферах діяльності. Використання таких систем сприяє оптимізації процесів працевлаштування, підвищенню ефективності взаємодії між учасниками ринку та формуванню більш обґрунтованих управлінських рішень, що відповідають сучасним вимогам цифрового суспільства.

1.2 Особливості функціонування ринку праці та інформаційних систем зайнятості населення

Ринок праці – це складова економічної системи держави, оскільки саме в його межах відбувається взаємодія між роботодавцями, які формують попит на робочу силу, та населенням, яке пропонує свої знання, навички, професійний досвід і трудовий потенціал. У загальному розумінні ринок праці являє собою систему соціально-економічних відносин, пов’язаних із формуванням, розподілом, перерозподілом і використанням трудових ресурсів. Його функціонування безпосередньо впливає на рівень зайнятості населення, динаміку безробіття,

розвиток окремих галузей економіки та загальний соціально-економічний стан суспільства [4]. На відміну від інших ринків, де об’єктом обміну виступають товари чи послуги, на ринку праці основним об’єктом є здатність людини до праці, що надає йому особливого соціального значення та робить процеси регулювання значно складнішими.

Особливість ринку праці полягає в тому, що він функціонує під впливом великої кількості взаємопов’язаних факторів, серед яких важливе місце займають економічна ситуація в державі, рівень розвитку технологій, демографічна структура населення, освітній рівень громадян, професійна мобільність, міграційні процеси та зміни у структурі попиту на професії. Усе це призводить до того, що ринок праці є динамічним середовищем, у якому параметри попиту і пропозиції постійно змінюються. Роботодавці висувають нові вимоги до кандидатів, з’являються нові професії, змінюється набір актуальних компетентностей, а частина спеціальностей, навпаки, поступово втрачає свою затребуваність. За таких умов ефективне функціонування ринку праці вимагає оперативного збору, аналізу та інтерпретації великої кількості інформації.

Проблеми ринку праці – це дисбаланс між попитом і пропозицією робочої сили. На практиці це означає, що кількість осіб, які шукають роботу, та кількість наявних вакансій не завжди відповідають одна одній ні за кількістю, ні за якісними характеристиками. Часто виникає ситуація, коли на ринку є значна кількість претендентів, однак їхня кваліфікація, рівень досвіду або професійні навички не відповідають вимогам роботодавців. Водночас роботодавці можуть відчувати дефіцит спеціалістів у певних галузях навіть за загального високого рівня безробіття [5]. Такий дисбаланс може бути зумовлений різними причинами: невідповідністю освітніх програм актуальним потребам економіки, недостатнім рівнем професійної підготовки, регіональною нерівномірністю розподілу робочих місць, низькою мобільністю працівників або швидкими змінами в структурі економіки.

Проблема дисбалансу попиту і пропозиції має не лише економічний, а й соціальний характер. Для шукачів роботи вона означає збільшення тривалості

пошуку вакансії, складнощі з працевлаштуванням за фахом, ризик втрати професійної кваліфікації та зниження рівня доходів. Для роботодавців така ситуація призводить до труднощів у пошуку відповідних кадрів, збільшення витрат на рекрутинг, зниження продуктивності через нестачу персоналу та ризику затримки виконання виробничих або управлінських процесів. Саме тому одним із важливих завдань сучасних інформаційних систем у сфері зайнятості є зменшення цього дисбалансу шляхом більш точного зіставлення характеристик вакансій і профілів кандидатів.

Ринок праці генерує значну кількість інформації: відомості про вакансії, професійні вимоги, рівень заробітної плати, географічне розташування робочих місць, освіту та досвід кандидатів, результати співбесід, динаміку ринку, попит на окремі спеціальності, аналітику зайнятості за регіонами та галузями. Ці дані можуть надходити з різних джерел, мати різний формат, структуру та ступінь актуальності. У результаті виникає потреба не лише у збереженні такої інформації, а й у її систематизації, фільтрації, обробці та подальшому використанні для прийняття рішень.

Без використання сучасних інформаційних технологій ефективно працювати з таким масивом відомостей стає практично неможливо. Ручний аналіз великої кількості резюме, вакансій і заявок потребує значних часових витрат, підвищує ймовірність помилок та не дає змоги швидко реагувати на зміни ринку. Крім того, значна кількість даних сама по собі не гарантує якісного результату, якщо відсутні інструменти їх аналітичного опрацювання. Саме тому в сучасних умовах інформаційні системи зайнятості повинні виконувати не лише роль електронних сховищ або каталогів вакансій, а й функцію інтелектуальної підтримки аналізу, що дозволяє перетворювати первинні дані на корисну для управління інформацію [6].

Окрему проблему становить повільний процес підбору персоналу та працевлаштування, який часто є наслідком неефективної організації роботи з даними та недостатнього рівня автоматизації. У традиційному підході пошук кандидатів, аналіз резюме, порівняння їх із вимогами вакансій, комунікація між сторонами та прийняття кінцевого рішення потребують значної участі людини на

кожному етапі. Такий процес може бути тривалим, особливо коли йдеться про велику кількість претендентів або складні вакансії з багатьма критеріями відбору. У результаті роботодавці витрачають більше часу на закриття вакансій, а шукачі роботи довше залишаються без працевлаштування.

Повільний підбір негативно впливає на всіх учасників ринку праці. Для роботодавця це означає втрату часу, зниження ефективності роботи організації, перевантаження наявного персоналу та потенційні фінансові збитки. Для кандидата це створює невизначеність, погіршує його можливості швидко знайти роботу та ускладнює професійну реалізацію. Для державних і муніципальних структур, що відповідають за регулювання зайнятості населення, повільність цих процесів зменшує ефективність програм працевлаштування та ускладнює оцінку реального стану ринку праці. Саме тому важливим напрямом розвитку є впровадження таких систем, які здатні автоматизувати первинний відбір, швидко зіставляти параметри вакансій і кандидатів, а також забезпечувати аналітичну підтримку прийняття рішень на кожному етапі.

Значення набувають інформаційні системи зайнятості населення, які покликані забезпечити цифрову підтримку процесів, пов’язаних із пошуком роботи, розміщенням вакансій, обліком заявок, аналізом попиту та пропозиції, а також моніторингом загальних тенденцій ринку праці. Такі системи можуть використовуватися як у діяльності державних центрів зайнятості, так і на комерційних платформах працевлаштування. Їх основне завдання полягає в тому, щоб забезпечити оперативне збирання та оновлення інформації, спростити доступ користувачів до потрібних відомостей і підвищити ефективність взаємодії між роботодавцями та шукачами роботи.

Інформаційні системи зайнятості поступово трансформуються з простих платформ розміщення вакансій у більш складні програмні рішення, які здатні виконувати аналітичні та рекомендаційні функції. Якщо раніше основний акцент робився на зберіганні резюме та вакансій, то нині зростає потреба в системах, які можуть автоматично групувати інформацію, виконувати фільтрацію, визначати релевантність кандидатів, формувати рейтинги, відстежувати динаміку заявок та

надавати узагальнені статистичні показники. Саме в цьому проявляється перехід від звичайної інформаційної системи до системи підтримки прийняття рішень у сфері зайнятості. Вони мають забезпечувати не лише технічну обробку інформації, а й створювати умови для більш обґрунтованого прийняття рішень. Для цього вони повинні включати засоби аналізу даних, формування рекомендацій, оцінювання відповідності кандидатів вакансіям, візуалізації статистики та контролю за проходженням заявок. Наявність таких функцій дозволяє одночасно вирішувати кілька важливих завдань: скорочувати тривалість пошуку роботи, підвищувати точність підбору персоналу, зменшувати навантаження на адміністраторів системи та покращувати загальну керованість процесами працевлаштування.

Функціонування ринку праці особливого значення набуває використання інформаційних технологій, які виступають ключовим інструментом підвищення ефективності процесів працевлаштування. Роль ІТ у цій сфері полягає насамперед у автоматизації основних операцій, пов’язаних із обробкою інформації, взаємодією між учасниками ринку та прийняттям управлінських рішень. Завдяки автоматизації значно скорочується час виконання рутинних задач, таких як реєстрація користувачів, обробка резюме, пошук вакансій, подача заявок і їх облік. Це дозволяє мінімізувати вплив людського фактору, зменшити кількість помилок та забезпечити більш стабільну роботу системи (рис. 1.4).

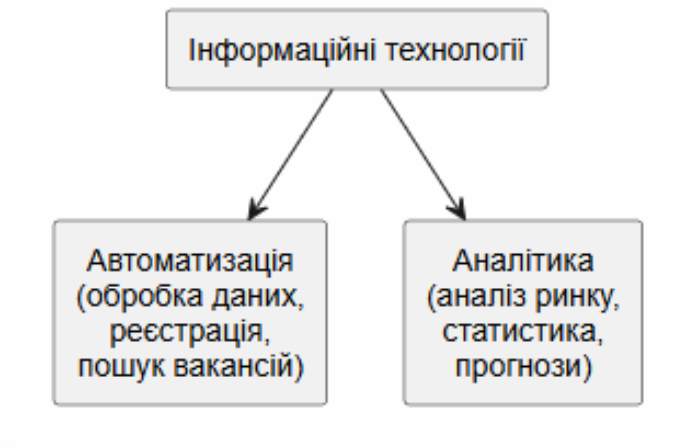

Рисунок 1.4 – Роль інформаційних технологій у системах зайнятості населення

Крім автоматизації, функцією інформаційних технологій є забезпечення аналітичної обробки даних. Сучасні інформаційні системи зайнятості здатні не лише накопичувати великі обсяги інформації, а й виконувати її глибокий аналіз. Це включає виявлення тенденцій ринку праці, аналіз попиту на професії, визначення найбільш затребуваних навичок, оцінку ефективності працевлаштування та формування статистичних показників. Використання аналітичних інструментів дозволяє перетворити розрізнені дані у структуровану інформацію, що може бути використана для прийняття обґрунтованих рішень як на рівні окремих користувачів, так і на рівні організацій чи державних установ [7].

Застосування інформаційних технологій також сприяє підвищенню швидкості обміну даними між учасниками ринку праці, забезпечує доступ до актуальної інформації в режимі реального часу та дозволяє реалізувати централізовані механізми управління процесами зайнятості. У результаті формується єдине інформаційне середовище, у якому всі дії — від пошуку вакансії до прийняття рішення про працевлаштування — виконуються більш ефективно та прозоро.

До основних категорій користувачів належать працівники (шукачі роботи), роботодавці та адміністратори системи, кожен із яких виконує власну роль у загальному процесі працевлаштування (рис. 1.5).

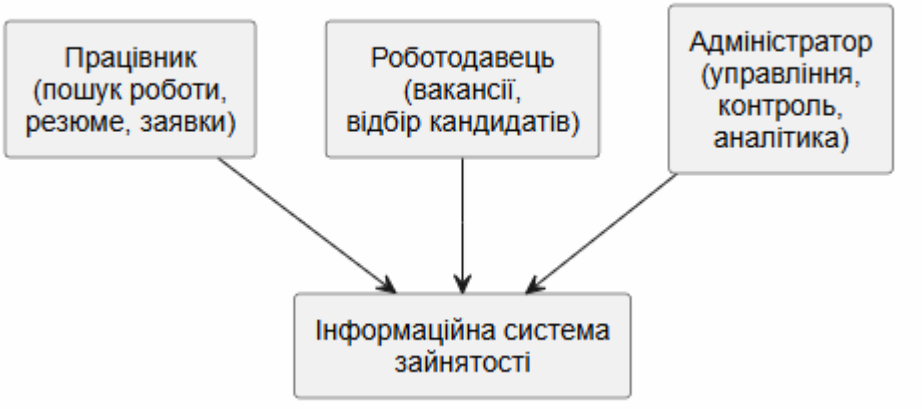

Рисунок 1.5 – Основні користувачі інформаційної системи зайнятості населення

Працівник виступає як користувач, зацікавлений у пошуку роботи, створенні та оновленні власного профілю, формуванні резюме, поданні заявок на вакансії та

відстеженні їхнього статусу. Для нього система є інструментом доступу до актуальних пропозицій на ринку праці та засобом представлення власних професійних характеристик.

Роботодавець, у свою чергу, використовує систему для розміщення вакансій, формування вимог до кандидатів, перегляду отриманих заявок, відбору претендентів і прийняття рішень щодо працевлаштування. Для нього інформаційна система виступає інструментом пошуку та оцінювання персоналу, що дозволяє значно скоротити час підбору та підвищити його якість.

Адміністратор забезпечує функціонування системи в цілому, здійснює контроль за її роботою, управління користувачами, моніторинг активності, обробку заявок і підтримку процесів взаємодії між іншими учасниками [8]. Крім того, адміністратор може виконувати аналітичні функції, пов’язані з оцінкою ефективності роботи системи та формуванням управлінських рішень на основі зібраних даних.

Інформаційні технології забезпечують не лише технічну основу функціонування систем зайнятості, а й створюють умови для їх трансформації у повноцінні системи підтримки прийняття рішень, що враховують потреби різних категорій користувачів і дозволяють підвищити ефективність взаємодії на ринку праці.

1.3 Аналіз існуючих інформаційних систем та платформ працевлаштування

У умовах розвитку ринку праці важливу роль відіграють спеціалізовані інформаційні системи та онлайн-платформи, які забезпечують взаємодію між роботодавцями та шукачами роботи. Найбільш поширеними серед них є такі платформи, як Work.ua, Rabota.ua, LinkedIn та Djinni. Кожна з цих систем має свої особливості функціонування, переваги та обмеження, що визначає їх ефективність у різних умовах використання.

Платформа Work.ua є однією з найбільших в Україні систем пошуку роботи, яка надає широкий вибір вакансій, зручний інтерфейс та інструменти для

створення резюме. Вона дозволяє користувачам здійснювати пошук за різними параметрами, отримувати рекомендації та підписуватися на нові вакансії . Основною перевагою платформи є її масовість і простота використання, однак вона має обмежені можливості аналітики та персоналізованих рекомендацій.

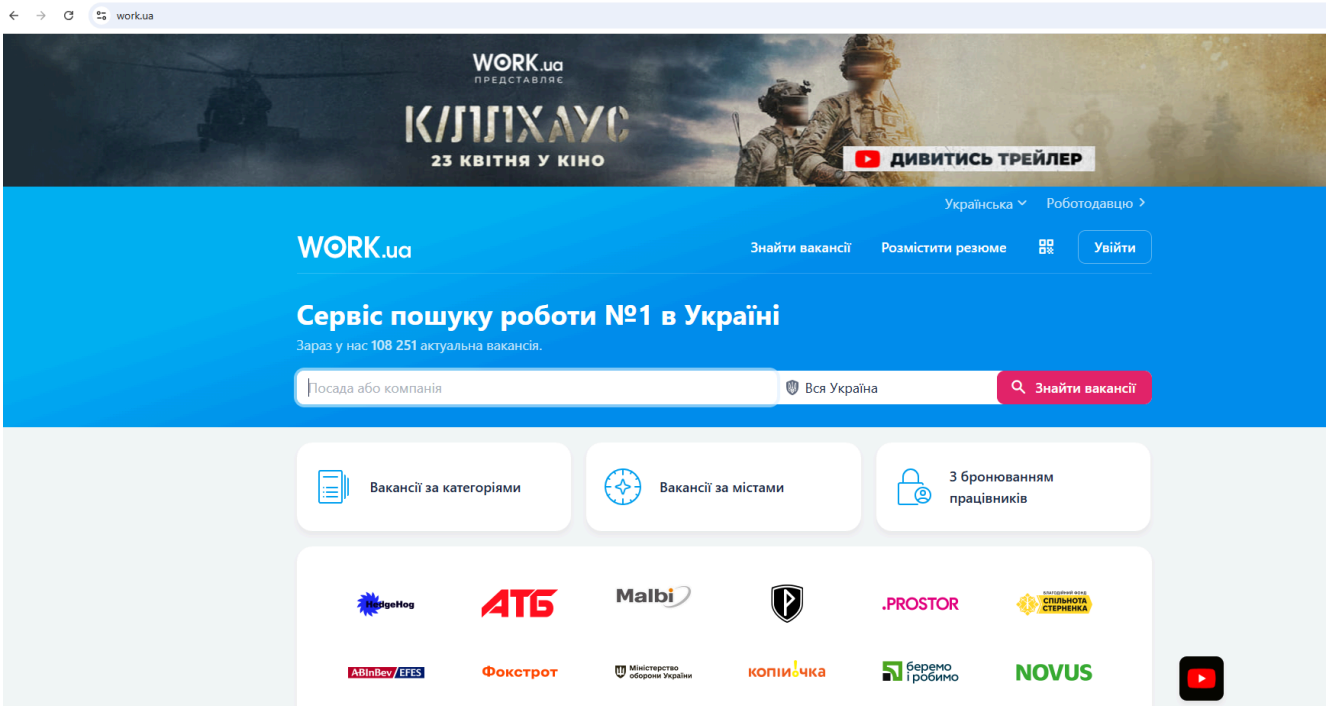

Рисунок 1.6 – Головна сторінка платформи Work.ua

Платформа Rabota.ua також є популярним інструментом працевлаштування, який забезпечує доступ до великої бази вакансій і резюме, а також пропонує зручні фільтри пошуку та додаткові сервіси для роботодавців . Вона орієнтована на широкий спектр професій і галузей, однак, як і більшість подібних платформ, не забезпечує глибокої аналітичної підтримки прийняття рішень (рис. 1.7).

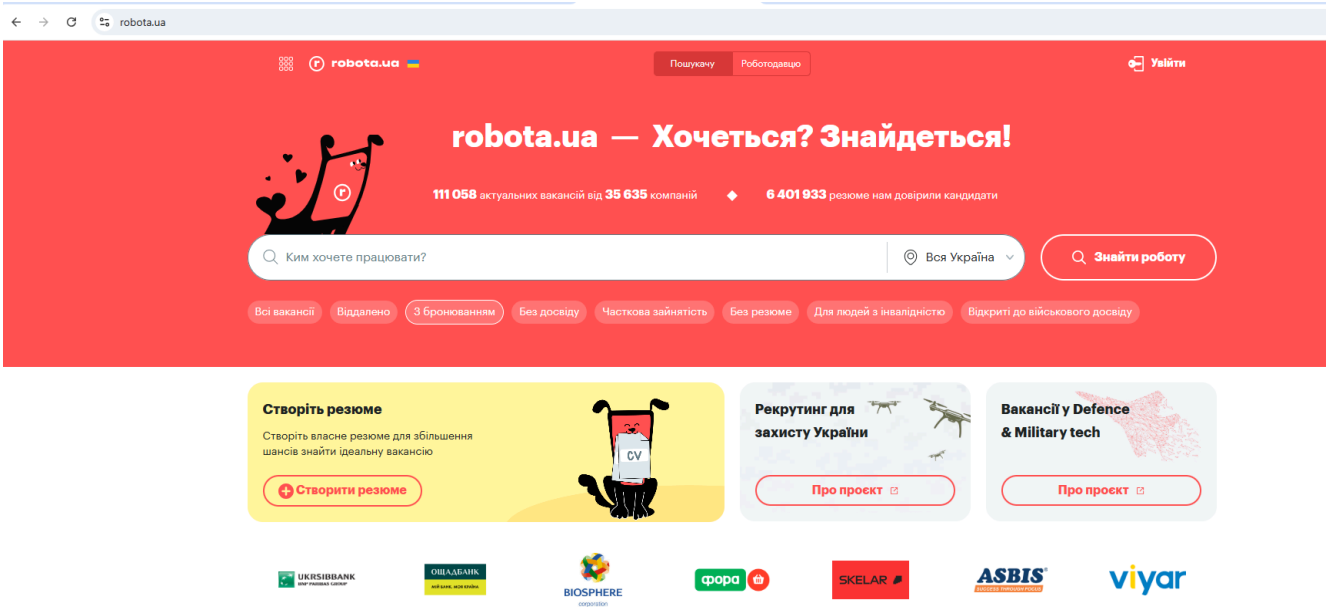

Рисунок 1.7 – Головна сторінка платформи Rabota.ua

Платформа LinkedIn відрізняється тим, що поєднує функції соціальної мережі та системи пошуку роботи. Вона дозволяє користувачам не лише знаходити вакансії, а й будувати професійні зв’язки, отримувати рекомендації та взаємодіяти з роботодавцями. Основною перевагою LinkedIn є можливість нетворкінгу та доступ до міжнародного ринку праці, однак складність інтерфейсу та залежність від соціальних зв’язків можуть ускладнювати процес пошуку роботи (рис. 1.8).

Рисунок 1.8 – Головна сторінка платформи LinkedIn

Платформа Djinni спеціалізується переважно на IT-сфері та пропонує інноваційний підхід до підбору персоналу. Однією з її особливостей є можливість анонімного пошуку роботи та прямої взаємодії між кандидатами і роботодавцями без посередників [9]. Крім того, система дозволяє бачити очікувану заробітну плату кандидатів, що спрощує процес відбору. Разом із тим, обмеженням платформи є її вузька спеціалізація та орієнтація переважно на IT-ринок (рис. 1.9).

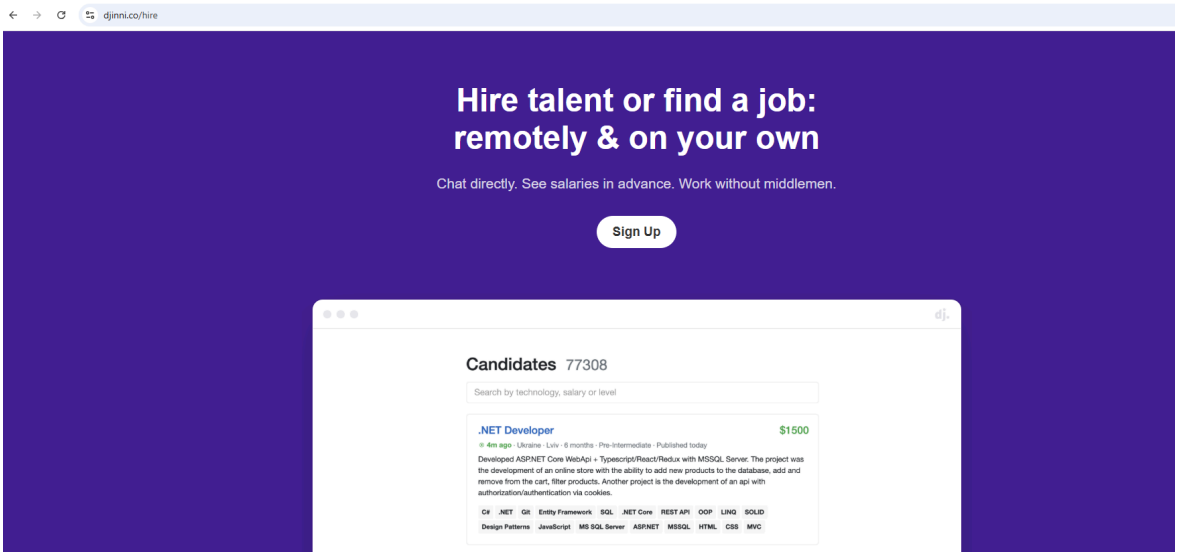

Рисунок 1.8 – Головна сторінка платформи Djinni

Таблиця 1.1 – Порівняння платформ працевлаштування

Критерій	Work.ua	Rabota.ua	**LinkedIn**	**Djinni**
Тип платформи	Класична job-платформа	Job-платформа	Соціальна мережа + jobs	IT-рекрутинг
Масштаб	Великий (Україна)	Великий (Україна)	Глобальний	Середній (IT)
Аналітика	Обмежена	Обмежена	Часткова	Часткова
Персоналізація	Середня	Середня	Висока	Висока
Спеціалізація	Універсальна	Універсальна	Універсальна	IT

Взаємодія	Через платформу	Через платформу	Через мережу контактів	Пряма
Основна перевага	Простота та кількість вакансій	Зручні фільтри	Нетворкінг	Прямий контакт
Основний недолік	Мало аналітики	Мало аналітики	Складність	Вузька сфера

Незважаючи на широку популярність і функціональність розглянутих платформ, таких як Work.ua, Rabota.ua, LinkedIn та Djinni, їх основним обмеженням є відсутність повноцінних механізмів підтримки прийняття рішень. Більшість із них орієнтовані на відображення інформації та забезпечення базової взаємодії між користувачами, але не надають глибокого аналітичного опрацювання даних, прогнозування або формування обґрунтованих рекомендацій. Зокрема, такі системи не забезпечують комплексного аналізу відповідності кандидатів вакансіям, не враховують динаміку ринку праці та не використовують інтелектуальні алгоритми для оптимізації процесу підбору персоналу.

Існуючі платформи недостатньо інтегрують функції аналітики з процесами прийняття рішень, що призводить до збереження значної ролі людського фактору та суб’єктивності у виборі кандидатів. Відсутність централізованої системи оцінювання ефективності працевлаштування, обмежені можливості прогнозування та слабка підтримка управлінських рішень на рівні організацій або державних структур також знижують їх загальну ефективність.

У зв’язку з цим виникає потреба у створенні інформаційної системи нового рівня, яка поєднує функції збору та зберігання даних із розвиненими аналітичними можливостями, механізмами формування рекомендацій та підтримки прийняття рішень. Саме така система дозволить підвищити якість підбору персоналу, оптимізувати процеси працевлаштування та забезпечити більш ефективне управління ринком праці, що і визначає актуальність розробки системи підтримки рішень у сфері зайнятості населення.

Висновок до розділу 1

Розглянуто теоретичні основи систем підтримки прийняття рішень та їх застосування у сфері зайнятості населення. Зокрема, визначено сутність систем підтримки прийняття рішень, їх основні характеристики та функціональне призначення в сучасних інформаційних системах. Показано, що СППР є ефективним інструментом обробки та аналізу даних, який дозволяє підвищити обґрунтованість, швидкість і якість прийняття управлінських рішень.

Проаналізовано особливості функціонування ринку праці, який характеризується динамічністю, складністю та залежністю від великої кількості факторів. Визначено основні проблеми сучасного ринку праці, серед яких дисбаланс між попитом і пропозицією робочої сили, наявність значних обсягів різнорідних даних та повільність процесів підбору персоналу. Обґрунтовано, що ефективне вирішення цих проблем потребує використання сучасних інформаційних технологій, зокрема систем підтримки прийняття рішень, які забезпечують автоматизацію процесів та аналітичну обробку інформації.

Розглянуто роль інформаційних систем зайнятості населення та визначено основні категорії користувачів, до яких належать працівники, роботодавці та адміністратори. Показано, що кожна з цих груп має власні функціональні потреби, що повинні враховуватися при розробці інформаційних систем.

2 ПРОЄКТУВАННЯ СИСТЕМИ ПІДТРИМКИ РІШЕНЬ У СФЕРІ ЗАЙНЯТОСТІ НАСЕЛЕННЯ

2.1 Аналіз функціональних та нефункціональних вимог до системи

Розробка інформаційної системи підтримки прийняття рішень у сфері зайнятості населення потребує чіткого визначення вимог до її функціонування, які забезпечують досягнення поставленої мети та ефективну взаємодію між користувачами. Формування вимог є одним із ключових етапів проєктування системи, оскільки саме на цьому етапі визначаються її основні можливості, обмеження та характеристики як програмного продукту. У межах даного підрозділу здійснюється аналіз функціональних та нефункціональних вимог до системи, що дозволяє сформувати цілісне уявлення про її структуру, поведінку та критерії якості [10]. Такий підхід забезпечує узгодженість між очікуваннями користувачів і технічною реалізацією системи, а також створює основу для подальшого проєктування архітектури та реалізації програмного рішення.

Функціональні вимоги визначають основні можливості системи та описують дії, які вона повинна виконувати для забезпечення потреб користувачів. У контексті розробки системи підтримки прийняття рішень у сфері зайнятості населення функціональні вимоги орієнтовані на забезпечення ефективної взаємодії між працівниками, роботодавцями та адміністраторами, а також на автоматизацію процесів пошуку роботи, підбору персоналу та аналізу даних ринку праці. Визначення таких вимог дозволяє чітко окреслити межі функціонування системи та сформувати основу для її подальшої реалізації.

Таблиця 2.1 – Функціональні вимоги до системи

№	Функція	Опис
1	Реєстрація та авторизація користувачів	Забезпечення можливості створення облікового запису, входу в систему та ідентифікації користувачів відповідно до їх ролей

Продовження таблиці 2.1

2	Створення та редагування резюме	Надання можливості працівникам створювати, переглядати та оновлювати власні резюме з інформацією про навички, досвід та освіту
3	Створення та управління вакансіями	Забезпечення роботодавцям можливості створювати вакансії, редагувати їх та керувати списком відкритих позицій
4	Подача та обробка заявок	Реалізація механізму подачі заявок на вакансії, перегляду їх статусу та взаємодії між кандидатами і роботодавцями
5	Аналітика та підтримка прийняття рішень	Формування статистичних даних, аналіз активності користувачів, вакансій і заявок, а також надання рекомендацій для прийняття управлінських рішень

Функціональні вимоги охоплюють основні процеси, пов’язані з роботою системи зайнятості, включаючи взаємодію користувачів, обробку інформації та аналітичну підтримку прийняття рішень. Реалізація зазначених функцій дозволяє забезпечити комплексний підхід до організації процесів працевлаштування та підвищити ефективність використання інформаційних ресурсів системи [11].

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на ефективність її функціонування, зручність використання та надійність роботи. На відміну від функціональних вимог, які описують конкретні дії системи, нефункціональні вимоги встановлюють критерії якості, яким має відповідати програмне рішення. У контексті системи підтримки прийняття рішень у сфері зайнятості населення особливу увагу приділено таким аспектам, як безпека, швидкодія, масштабованість та зручність використання, оскільки саме вони забезпечують стабільну роботу системи та позитивний досвід користувачів (табл. 2.2).

Таблиця 2.2 – Нефункціональні вимоги до системи

№	Вимога	Опис
1	Безпека	Забезпечення захисту даних користувачів шляхом використання механізмів автентифікації та авторизації, зокрема JWT-токенів, шифрування паролів та контролю доступу
2	Швидкодія	Забезпечення швидкої обробки запитів користувачів, мінімального часу відгуку системи та ефективної роботи з великими обсягами даних
3	Масштабованість	Можливість розширення функціоналу системи та обробки зростаючої кількості користувачів і даних без втрати продуктивності
4	Зручність використання	Наявність інтуїтивно зрозумілого інтерфейсу, простоти навігації та доступності функцій для різних категорій користувачів

Нефункціональні вимоги забезпечують якісні характеристики системи та створюють умови для її стабільного функціонування. Дотримання зазначених вимог дозволяє підвищити рівень захищеності даних, забезпечити швидку обробку інформації, адаптувати систему до зростання навантаження та покращити взаємодію користувачів із програмним продуктом [12].

Визначення функціональних та нефункціональних вимог дозволило сформувати цілісне уявлення про можливості, поведінку та якісні характеристики розроблюваної системи підтримки прийняття рішень у сфері зайнятості населення. Виділені вимоги відображають ключові потреби користувачів, а також забезпечують основу для реалізації ефективної, безпечної та зручної інформаційної системи. Зокрема, функціональні вимоги визначають основні сценарії взаємодії між учасниками системи, тоді як нефункціональні — встановлюють критерії її надійності, продуктивності та зручності використання.

Сформований набір вимог є базою для подальшого етапу проєктування, у межах якого визначається архітектура системи та принципи взаємодії її основних компонентів. З огляду на це, у наступному підрозділі доцільно розглянути структуру системи підтримки прийняття рішень, особливості її клієнт-серверної організації та взаємодію між окремими модулями.

2.2 Архітектура системи та взаємодія основних модулів

Інформаційна система «Employment Center» реалізована за клієнт-серверною архітектурою. Клієнтська частина (frontend) відповідає за інтерфейс користувача та формування HTTP-запитів до API, тоді як серверна частина (backend) забезпечує бізнес-логіку, автентифікацію, розмежування доступу за ролями та роботу з базою даних SQLite. Такий підхід забезпечує модульність, спрощує супровід і дозволяє масштабувати функціональність окремих підсистем незалежно.

Ключовими серверними модулями є маршрути автентифікації, модулі для ролей worker, employer, admin, а також модуль статистики. Централізована точка входу сервера виконує підключення middleware, публікацію статичних ресурсів та маршрутизацію API-запитів. З боку клієнта взаємодія з сервером уніфікована через функцію api(...), яка автоматично додає JWT-токен до заголовка Authorization [16, 17].

Для опису композиції серверної частини доцільно використати фрагмент ініціалізації Express-застосунку.

На рисунку 2.1 представлено фрагмент коду централізованої маршрутизації серверної частини, що відображає поділ API на функціональні модулі.

```
server.js U
employment-center > backend > server.js > ...
1   require('dotenv').config();
2
3   const express = require('express');
4   const cors = require('cors');
5   const path = require('path');
6
7   const authRoutes = require('./routes/auth');
8   const workerRoutes = require('./routes/workers');
9   const employerRoutes = require('./routes/employers');
10  const adminRoutes = require('./routes/admin');
11  const statsRoutes = require('./routes/stats');
12
13  const app = express();
14  const PORT = process.env.PORT || 9999;
15
16  // Middleware
17  app.use(cors());
18  app.use(express.json());
19
20  // Статичні файли фронтенду
21  app.use(express.static(path.join(__dirname, '../frontend')));
22
23  // API роутери
24  app.use('/api/auth', authRoutes);
25  app.use('/api/workers', workerRoutes);
26  app.use('/api/employers', employerRoutes);
27  app.use('/api/admin', adminRoutes);
28  app.use('/api/stats', statsRoutes);
29
30  // Головна сторінка
31  app.get('/', (req, res) => {
32    res.sendFile(path.join(__dirname, '../frontend/index.html'));
33  });
34
35  // Обробка помилок
36  app.use((err, req, res, next) => {
37    console.error(err.stack);
38    res.status(500).json({ error: 'Something went wrong!' });
39  });
40
41  // Запуск сервера
42  app.listen(PORT, () => {
43    console.log(`🚀 Server running on http://localhost:${PORT}`);
44    console.log(`📁 Frontend: http://localhost:${PORT}`);
45    console.log(`🔧 API: http://localhost:${PORT}/api`);
46  });
47
```

Рисунок 2.1 – Централізована маршрутизація модулів API у Node.js/Express

Як показано на рисунку 2.1, кожен функціональний блок винесено в окремий роутер, що забезпечує слабке зв’язування модулів і спрощує розширення системи.

Для демонстрації взаємодії frontend із backend варто навести уніфіковану клієнтську функцію виклику API.

На рисунку 2.2 наведено фрагмент клієнтського модуля доступу до API, який реалізує єдиний механізм обміну даними між інтерфейсом та сервером.

Рисунок 2.2 – Функція уніфікованого API-виклику у клієнтській частині

З рисунка 2.2 видно, що запити стандартизовано на рівні заголовків, серіалізації тіла та обробки помилок, що підвищує надійність клієнт-серверної взаємодії.

Для відображення логіки контролю доступу доцільно використати middleware автентифікації.

На рисунку 2.3 представлено реалізацію перевірки JWT-токена для захищених маршрутів серверної частини [17].

auth.js U

employment-center > backend > middleware > auth.js > ...

```
1   const jwt = require('jsonwebtoken');
2
3   const JWT_SECRET = process.env.JWT_SECRET || 'employment-center-secre
4
5   function authenticateToken(req, res, next) {
6     const authHeader = req.headers['authorization'];
7     const token = authHeader && authHeader.split(' ')[1];
8
9     if (!token) {
10      return res.status(401).json({ error: 'Access token required' });
11    }
12
13    jwt.verify(token, JWT_SECRET, (err, user) => {
14      if (err) {
15        return res.status(403).json({ error: 'Invalid or expired token'
16      }
17      req.user = user;
18      next();
19    });
20  }
21
22  function requireRole(...roles) {
23    return (req, res, next) => {
24      if (!req.user || !roles.includes(req.user.role)) {
25        return res.status(403).json({ error: 'Access denied: insufficie
26      }
27      next();
28    };
29  }
30
31  module.exports = { authenticateToken, requireRole, JWT_SECRET };
```

Рисунок 2.3 – Middleware автентифікації та перевірки JWT

Як показано на рисунку 2.3, доступ до захищених ресурсів надається лише після валідної верифікації токена, що формує базовий рівень безпеки системи.

Для формального подання архітектури пропонується компонентна UML-діаграма.

Для узагальнення структури програмного рішення побудовано компонентну UML-діаграму. На рисунку 2.4 представлено основні підсистеми та напрямки їх взаємодії [18].

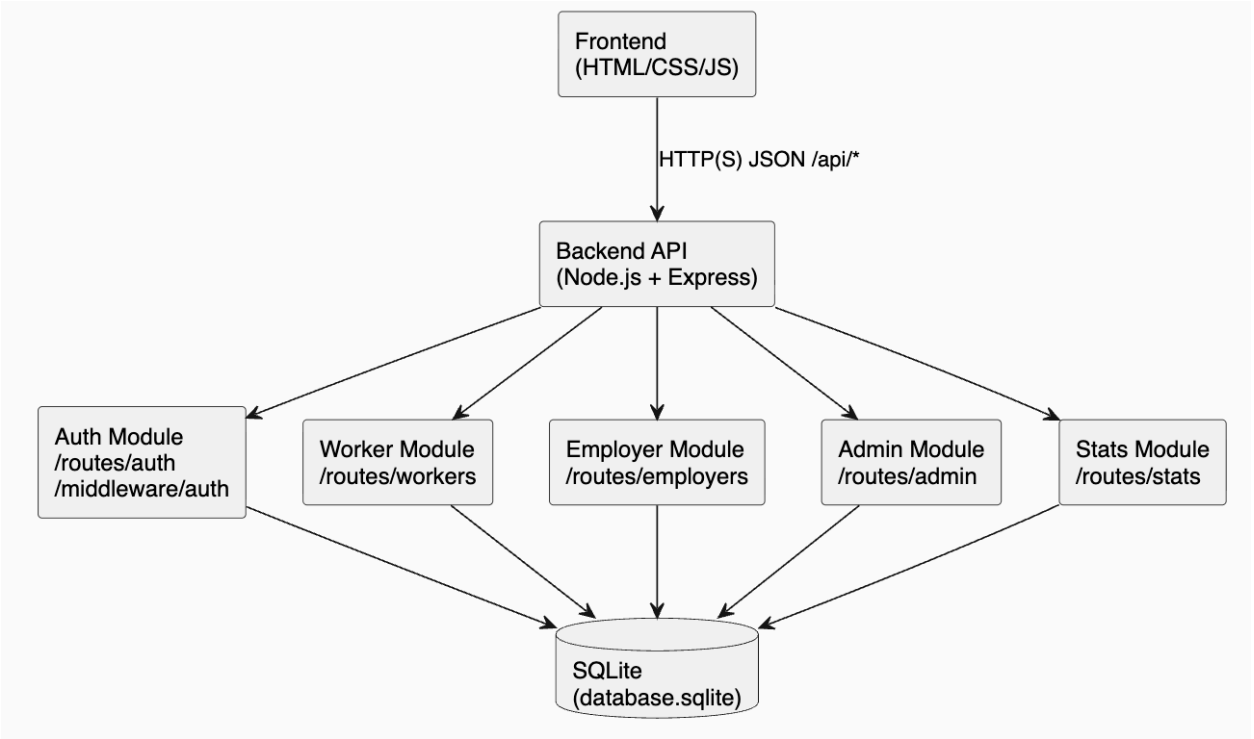

Рисунок 2.4 – UML компонентна діаграма архітектури системи

З рисунка 2.4 видно, що система має чітко виражену модульну структуру: клієнтський рівень взаємодіє лише з API, а вся доменна логіка зосереджена на сервері з доступом до єдиного сховища даних.

2.3 Проєктування бази даних та API взаємодії

У підсистемі збереження даних використано реляційну БД SQLite, у якій структура предметної області розділена на сутності користувачів, профілів, вакансій, заявок, призначень та журналу подій. Такий підхід дозволяє підтримувати цілісність даних за рахунок зовнішніх ключів і забезпечує прозору реалізацію бізнес-процесу обробки заявок через REST API.

Схема БД реалізована в модулі ініціалізації: db.js (line 45). Базові API-маршрути реалізовано у модулях: workers.js (line 7), admin.js (line 7), auth.js (line 9).

На рисунку 2.5 представлено фрагмент коду проєктування базових сутностей користувачів та рольових профілів у реляційній моделі даних.

Для таблиць ядра системи доцільно взяти скріншот такого фрагмента [19]:

```
CREATE TABLE IF NOT EXISTS users (... role TEXT NOT NULL CHECK(role IN ('worker', '

CREATE TABLE IF NOT EXISTS workers (
  ...
  user_id INTEGER UNIQUE NOT NULL,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
)

CREATE TABLE IF NOT EXISTS employers (
  ...
  user_id INTEGER UNIQUE NOT NULL,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
)
```

Рисунок 2.5 – Реалізація таблиць users, workers, employers у SQLite

Як показано на рисунку 2.5, модель даних забезпечує рольову диференціацію користувачів і підтримує цілісність зв’язків через зовнішні ключі.

На рисунку 2.6 наведено фрагмент схеми, що описує життєвий цикл заявки та механізм її призначення роботодавцю.

Для логіки обробки заявок варто показати зв’язок worker_applications і admin_assignments [20, 21]:

```
CREATE TABLE IF NOT EXISTS worker_applications (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  worker_id INTEGER NOT NULL,
  ...
  status TEXT DEFAULT 'pending' CHECK(status IN (
    'pending','admin_assigned','employer_viewed','employer_accepted',
    'employer_rejected','completed','cancelled'
  )),
  FOREIGN KEY (worker_id) REFERENCES workers(id) ON DELETE CASCADE
)

CREATE TABLE IF NOT EXISTS admin_assignments (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  application_id INTEGER NOT NULL,
  employer_id INTEGER NOT NULL,
  admin_id INTEGER NOT NULL,
  ...
  UNIQUE(application_id, employer_id)
)
```

Рисунок 2.6 – Структура таблиць worker_applications та admin_assignments

З рисунка 2.6 видно, що статусна модель заявок і таблиця призначень формують основу маршрутизації кандидата між учасниками процесу.

На рисунку 2.7 представлено реалізацію REST-ендпоінта створення заявки працівником із валідацією ролі доступу.

```
JS workers.js U ×
employment-center > backend > routes > JS workers.js > ...
1   const express = require('express');
2   const db = require('../config/db');
3   const { authenticateToken, requireRole } = require('../middleware/auth'
4
5   const router = express.Router();
6
7   // Створити нову заявку (замість подання на конкретну вакансію)
8   router.post('/applications', authenticateToken, requireRole('worker'),
9     const { title, skills, experience, education, desired_salary, preferr
10
11    db.get('SELECT id FROM workers WHERE user_id = ?', [req.user.id], (er
12      if (err || !worker) {
13        return res.status(404).json({ error: 'Worker not found' });
14      }
15
16      const preferredVacanciesJson = preferred_vacancies ? JSON.stringify
17
18      db.run(
19        `INSERT INTO worker_applications
20         (worker_id, industry_id, title, skills, experience, education, d
21         VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, 'pending')`,
22        [worker.id, industry_id || null, title || '', skills || '', exper
23         desired_salary || 0, preferredVacanciesJson, additional_info ||
24        function(err) {
25          if (err) {
26            return res.status(500).json({ error: err.message });
27          }
28
29          res.status(201).json({
30            message: 'Application created successfully',
31            application_id: this.lastID
32          });
33        }
34      );
35    });
36  });
37
38  // Отримати всі заявки працівника з історією
39  router.get('/applications', authenticateToken, requireRole('worker'), (
40    db.get('SELECT id FROM workers WHERE user_id = ?', [req.user.id], (er
41      if (err || !worker) {
42        return res.status(404).json({ error: 'Worker not found' });
43      }
44
45      db.all(
46        `SELECT
47          wa.*,
48          i.name as industry_name,
49          aa.id as assignment_id,
50          aa.employer_id,
51          aa.assigned_at,
52          aa.notes as admin_notes,
53          e.company_name,
54          u.full_name as admin_name
```

Рисунок 2.7 – API-метод POST /api/workers/applications

Як показано на рисунку 2.7, ендпоінт виконує авторизоване створення запису в таблиці заявок зі стартовим статусом pending.

На рисунку 2.8 наведено фрагмент серверної логіки, у якій операція призначення заявки синхронізується зі зміною її статусу.

```
JS admin.js U ×
employment-center > backend > routes > JS admin.js > ...
1   const express = require('express');
2   const db = require('../config/db');
3   const { authenticateToken, requireRole } = require('../middleware/auth'
4
5   const router = express.Router();
6
7   // Отримати всі заявки працівників (з розширеними фільтрами)
8   router.get('/applications', authenticateToken, requireRole('admin'), (r
9     const { status, worker_id, skills, salary_min, salary_max, date_from,
10
11    // Підзапит для отримання всіх призначень для кожної заявки
12    let query = `
13      SELECT
14        wa.*,
15        w.user_id as worker_user_id,
16        u.full_name as worker_name,
17        u.email as worker_email,
18        u.phone as worker_phone,
19        i.name as industry_name,
20        -- Отримуємо всі призначені компанії через GROUP_CONCAT
21        GROUP_CONCAT(DISTINCT e.company_name) as assigned_companies,
22        GROUP_CONCAT(DISTINCT e.id) as assigned_employer_ids,
23        COUNT(DISTINCT aa.id) as assignments_count
24      FROM worker_applications wa
25      JOIN workers w ON wa.worker_id = w.id
26      JOIN users u ON w.user_id = u.id
27      LEFT JOIN industries i ON wa.industry_id = i.id
28      LEFT JOIN admin_assignments aa ON wa.id = aa.application_id AND aa.
29      LEFT JOIN employers e ON aa.employer_id = e.id
30      WHERE 1=1
31    `;
32    const params = [];
33
34    if (status) {
35      query += ' AND wa.status = ?';
36      params.push(status);
37    }
38
39    if (worker_id) {
40      query += ' AND wa.worker_id = ?';
41      params.push(worker_id);
42    }
43
44    if (skills) {
45      query += ' AND (wa.skills LIKE ? OR wa.title LIKE ?)';
46      const searchTerm = `%${skills}%`;
47      params.push(searchTerm, searchTerm);
48    }
49
50    if (salary_min) {
51      query += ' AND wa.desired_salary >= ?';
52      params.push(parseInt(salary_min));
53    }
54
```

Рисунок 2.8 – Обробка призначення заявки в модулі адміністратора

З рисунка 2.8 видно, що API реалізує керований перехід заявки до стану admin_assigned після формування запису про призначення.

Для формалізації структури даних побудовано UML-діаграму класів (ER-орієнтоване подання).

На рисунку 2.9 представлено ключові сутності та їх зв’язки [22].

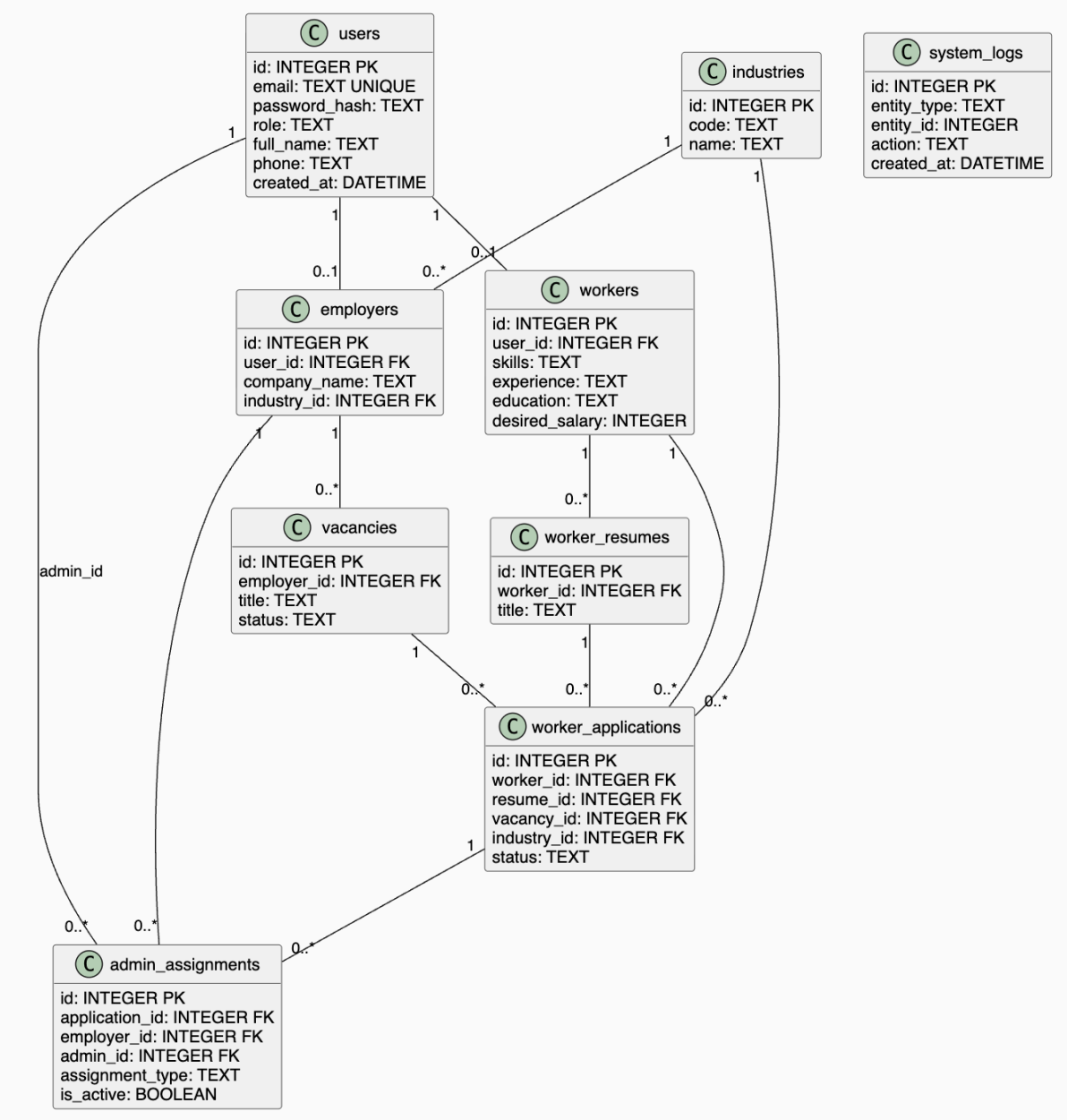

Рисунок 2.9 – UML-діаграма структури бази даних

Як видно з рисунка 2.9, проєктування БД орієнтоване на нормалізовану модель із чітким поділом довідникових, транзакційних і сервісних сутностей.

Для опису прикладного сценарію обміну даними побудовано UML-діаграму послідовності. На рисунку 2.10 відображено процес створення та призначення заявки.

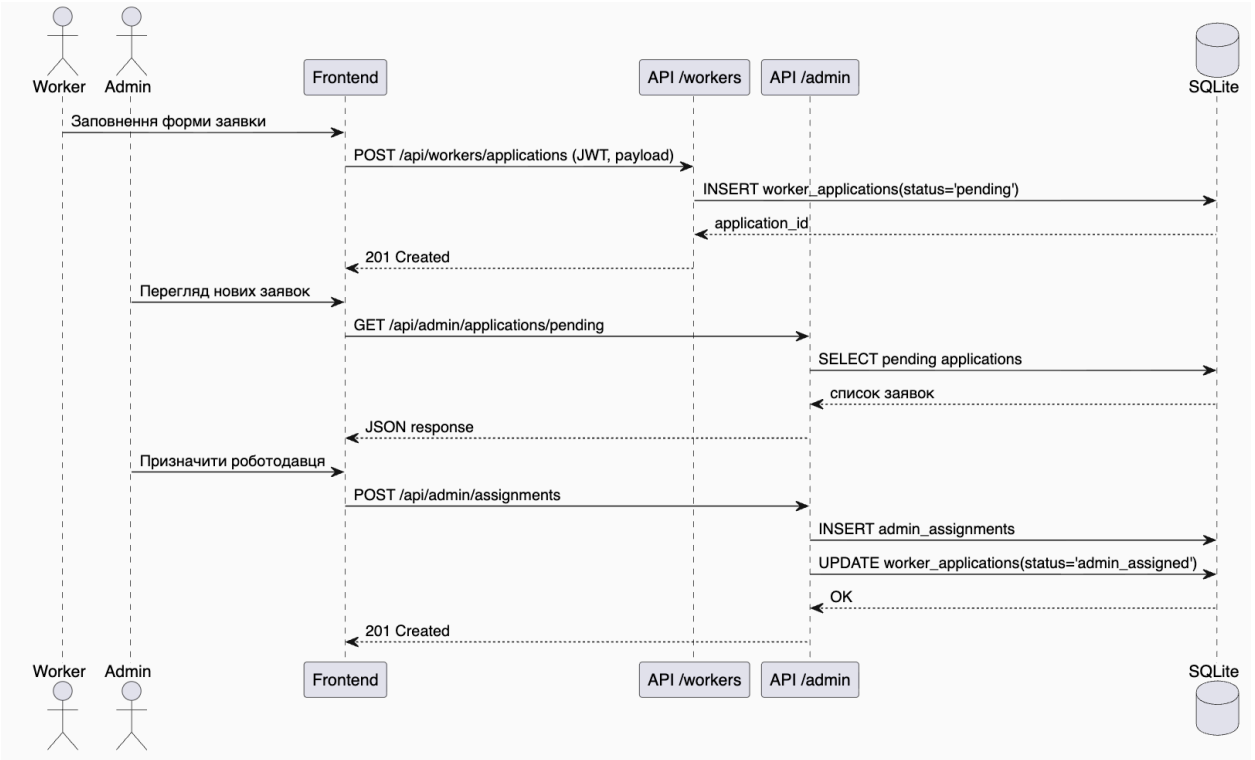

Рисунок 2.10 – UML-діаграма послідовності API-взаємодії

З рисунка 2.10 видно, що API забезпечує послідовний перехід заявки від етапу створення до етапу адміністративного призначення з фіксацією стану в БД.

Висновок до розділу 2

У другому розділі виконано системне проєктування програмного рішення «Employment Center» на рівні архітектури, модульної структури та моделі взаємодії компонентів. Було обґрунтовано вибір клієнт-серверного підходу, у межах якого frontend забезпечує інтерфейс і первинну взаємодію з користувачем, а backend реалізує бізнес-логіку, авторизацію, обробку ролей і централізовану маршрутизацію запитів. Така організація дозволяє розділити відповідальність між підсистемами, підвищити керованість коду та створити основу для подальшого функціонального розширення.

У підрозділах, присвячених архітектурі системи та взаємодії основних модулів, визначено структуру API і механізми доступу до функцій залежно від ролі користувача (працівник, роботодавець, адміністратор). Окремо розглянуто принципи автентифікації та контролю прав доступу на основі JWT, що забезпечує безпечну роботу із захищеними маршрутами. Побудовані UML-діаграми відобразили як статичну композицію компонентів, так і логіку їх взаємодії в межах типових сценаріїв роботи системи.

У частині проєктування бази даних та API взаємодії сформовано реляційну модель, що охоплює ключові сутності предметної області: користувачів, профілі, резюме, вакансії, заявки, призначення та журнал подій. Визначені зв’язки між таблицями, статусна модель обробки заявок і REST-ендпоінти забезпечують цілісність даних та узгодженість бізнес-процесів. Отримані результати розділу 2 формують завершену проєктну основу для подальшої реалізації, тестування та оцінювання ефективності системи в наступних розділах дипломної роботи.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ СИСТЕМИ ПІДТРИМКИ РІШЕНЬ

3.1 Реалізація серверної частини системи та API (Node.js, Express, SQLite)

Серверну частину системи реалізовано на платформі Node.js із використанням фреймворку Express. Такий вибір забезпечує швидку побудову REST API, зручну маршрутизацію та просту інтеграцію middleware для автентифікації, обробки помилок і серіалізації JSON. Як сховище даних використано SQLite, що є достатнім для локального розгортання, тестування і навчального проєкту [23, 24].

Таблиця 3.1 – Технологічний стек серверної частини системи та призначення компонентів

Компонент	Технологія	Призначення
Runtime	Node.js	Виконання JavaScript на сервері
Framework	Express.js	Веб-фреймворк для API
База даних	SQLite3	Локальна реляційна БД
Автентифікація	JWT (jsonwebtoken)	Токен-базована авторизація
Хешування	bcryptjs	Шифрування паролів
CORS	cors	Налаштування крос-доменних запитів
HTML	Vanilla HTML5	Розмітка сторінок
CSS	CSS3 + Custom Properties	Темна тема, адаптивний дизайн
JavaScript	Vanilla ES6+	Логіка клієнта
Графіки	Chart.js	Візуалізація статистики
Іконки	Font Awesome	Векторні іконки

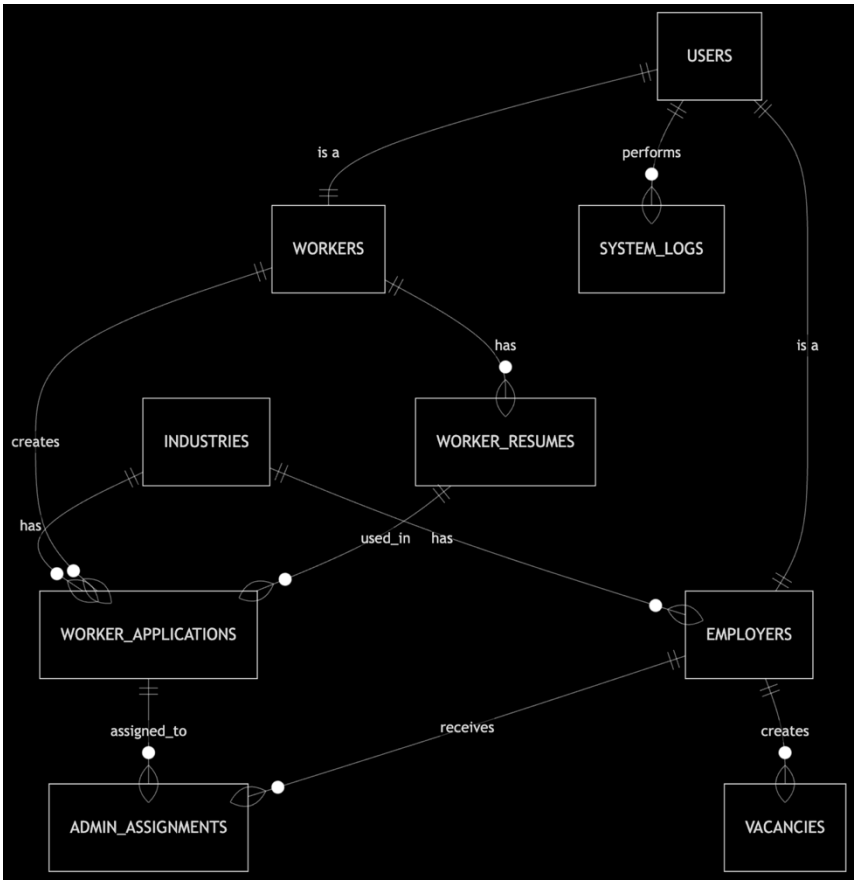

Рисунок 3.1 – Схема даних

Таблиці

`users` - Користувачі системи

```
- id (PK, AUTOINCREMENT)
- email (UNIQUE, NOT NULL)
- password_hash (NOT NULL)
- role (admin/worker/employer)
- full_name (NOT NULL)
- phone
- created_at (DEFAULT CURRENT_TIMESTAMP)
```

`workers` - Профілі працівників

```
- id (PK)
- user_id (FK → users.id, UNIQUE)
- skills (TEXT)
- experience (TEXT)
- education (TEXT)
- desired_salary (INTEGER)
- is_visible_to_employers (BOOLEAN)
```

`employers` - Профілі роботодавців

```
- id (PK)
- user_id (FK → users.id, UNIQUE)
- company_name (NOT NULL)
- description (TEXT)
- address (TEXT)
- industry_id (FK → industries.id)
- website (TEXT)
- company_size (TEXT)
- founded_year (INTEGER)
```

`worker_resumes` - Резюме працівників

```
- id (PK)
- worker_id (FK → workers.id)
- title (NOT NULL)
- skills, experience, education
- desired_salary, preferred_location
- employment_type (full_time/part_time/remote)
- is_active (BOOLEAN)
```

`worker_applications` - Заявки на працевлаштування

```
- id (PK)
- worker_id (FK → workers.id)
- resume_id (FK → worker_resumes.id)
- industry_id (FK → industries.id)
- title, skills, experience, education
- desired_salary, preferred_location
- status (pending/admin_assigned/employer_accepted/employer_rejected/completed/cancelled)
```

`vacancies` - Вакансії роботодавців

```
- id (PK)
- employer_id (FK → employers.id)
- title (NOT NULL)
- description, requirements, skills, location
- salary (INTEGER)
- employment_type
- status (active/closed)
```

`admin_assignments` - Призначення заявок

```
- id (PK)
- application_id (FK → worker_applications.id)
- employer_id (FK → employers.id)
- admin_id (FK → users.id)
- assigned_at (DEFAULT CURRENT_TIMESTAMP)
- notes (TEXT)
- is_active (BOOLEAN)
- UNIQUE(application_id, employer_id)
```

`system_logs` - Логи системи

```
- id (PK)
- entity_type, entity_id, action
- performed_by, performed_by_role
- old_value, new_value
- details, created_at
```

API Endpoints

Таблиця 3.3 – Авторизація (`/api/auth`)

Метод	**Endpoint**	**Опис**
POST	/register	Реєстрація нового користувача
POST	/login	Вхід в систему (повертає JWT)
GET	/profile	Отримання профілю поточного користувача

Таблиця 3.4 – Працівник (`/api/workers`)

Метод	**Endpoint**	**Опис**
GET	/profile	Профіль працівника
PUT	/profile	Оновлення профілю
GET	/resumes	Список резюме
POST	/resumes	Створення резюме
PUT	/resumes/:id	Редагування резюме
DELETE	/resumes/:id	Видалення резюме
GET	/applications	Заявки працівника
POST	/applications	Створення заявки
DELETE	/applications/:id	Видалення заявки

Таблиця 3.5 – Роботодавець (`/api/employers`)

Метод	**Endpoint**	**Опис**
GET	/profile	Профіль компанії
PUT	/profile	Оновлення профілю

GET	/industries	Список галузей

Продовження таблиці 3.5

GET	/vacancies	Вакансії компанії
POST	/vacancies	Створення вакансії
PUT	/vacancies/:id	Редагування вакансії
DELETE	/vacancies/:id	Видалення вакансії

Таблиця 3.6 – Адміністратор (`/api/admin`)

Метод	Endpoint	Опис
GET	/applications	Всі заявки (з фільтрами)
GET	/applications/:id	Деталі заявки
GET	/applications/:id/assignments	Призначення заявки
POST	/assignments	Призначення заявки роботодавцю
PUT	/assignments/:id/cancel	Скасування призначення
GET	/workers	Список працівників
GET	/workers/:id	Деталі працівника
GET	/employers	Список роботодавців
GET	/employers/:id/details	Деталі компанії
GET	/users	Всі користувачі
DELETE	/users/:id	Видалення користувача

Таблиця 3.7 – Статистика (`/api/stats`)

Метод	Endpoint	Опис
GET	/applications	Всі заявки (з фільтрами)
GET	/applications/:id	Деталі заявки
GET	/applications/:id/assignments	Призначення заявки
POST	/assignments	Призначення заявки роботодавцю
PUT	/assignments/:id/cancel	Скасування призначення

GET	/workers	Список працівників
GET	/workers/:id	Деталі працівника

Продовження таблиці 3.7

GET	/employers	Список роботодавців
GET	/employers/:id/details	Деталі компанії
GET	/users	Всі користувачі
DELETE	/users/:id	Видалення користувача

3.2 Реалізація підсистеми підтримки прийняття рішень та аналітики

Підсистема підтримки прийняття рішень реалізована у вигляді окремого серверного модуля routes/stats.js, який формує агреговані показники, часові зрізи, метрики ефективності та журнал подій для адміністратора. Вона працює поверх транзакційних даних системи (заявки, призначення, користувачі, вакансії) і повертає результати у форматі JSON для подальшої візуалізації на дашборді. Такий підхід дозволяє перейти від операційного обліку до аналітичної інтерпретації процесів працевлаштування.

Для базової аналітики використовується endpoint оглядової статистики: stats.js (line 7) [25, 26].

На рисунку 3.2 представлено фрагмент реалізації endpoint загальної статистики, що формує ключові KPI системи.

```
JS stats.js U ×
employment-center > backend > routes > JS stats.js > ...
1   const express = require('express');
2   const db = require('../config/db');
3   const { authenticateToken, requireRole } = require('../middleware/auth
4
5   const router = express.Router();
6
7   // Загальна статистика
8   router.get('/overview', authenticateToken, (req, res) => {
9     const stats = {};
10
11    db.get('SELECT COUNT(*) as count FROM users WHERE role = ?', ['worke
12      stats.totalWorkers = row ? row.count : 0;
13
14      db.get('SELECT COUNT(*) as count FROM users WHERE role = ?', ['emp
15        stats.totalEmployers = row ? row.count : 0;
16
17        db.get('SELECT COUNT(*) as count FROM vacancies WHERE status = ?
18          stats.activeVacancies = row ? row.count : 0;
19
20          db.get('SELECT COUNT(*) as count FROM worker_applications', (e
21            stats.totalApplications = row ? row.count : 0;
22
23            db.get('SELECT COUNT(*) as count FROM worker_applications WH
24              stats.completedPlacements = row ? row.count : 0;
25
26              db.get('SELECT COUNT(*) as count FROM worker_applications
27                stats.pendingApplications = row ? row.count : 0;
28
29                db.get('SELECT COUNT(*) as count FROM worker_application
30                  stats.assignedApplications = row ? row.count : 0;
31
32                  db.get('SELECT COUNT(*) as count FROM worker_applicati
33                    stats.acceptedApplications = row ? row.count : 0;
34
35                    res.json(stats);
36                  });
37                });
38              });
39            });
40          });
41        });
42      });
43    });
44  });
45
46  // Дані для графіків
47  router.get('/charts', authenticateToken, (req, res) => {
48    const data = {};
49
50    // Заявки за статусами
51    db.all(
52      `SELECT status, COUNT(*) as count FROM worker_applications GROUP B
53      [],
54      (err, applicationsByStatus) => {
```

Рисунок 3.2 – Реалізація API-методу GET /api/stats/overview

Як показано на рисунку 3.2, сервер повертає зведені метрики, які використовуються для оперативного моніторингу стану платформи.

Поглиблена аналітика реалізована в endpoint GET /api/stats/charts, де формується набір даних для графіків: статуси заявок, динаміка за днями, топ навичок, галузеві зрізи, зарплатні діапазони, помісячна активність. Джерело: stats.js (line 46).

На рисунку 3.3 наведено фрагмент модуля формування аналітичних вибірок для візуалізації [26].

```
72   T role, COUNT(*) as count FROM users GROUP BY role`,
73
74   usersByRole) => {
75   .usersByRole = usersByRole || [];
76
77   акансії за статусом
78   ll(
79   ELECT status, COUNT(*) as count FROM vacancies GROUP BY status`,
80   ,
81   rr, vacanciesByStatus) => {
82   data.vacanciesByStatus = vacanciesByStatus || [];
83
84   // Топ навичок
85   db.all(
86     `SELECT skills FROM worker_applications WHERE skills IS NOT NULL AND
87     [],
88     (err, rows) => {
89       const skillCounts = {};
90       rows.forEach(row => {
91         const skills = row.skills.split(/[,;]/).map(s => s.trim().toLowe
92         skills.forEach(skill => {
93           if (skill) {
94             skillCounts[skill] = (skillCounts[skill] || 0) + 1;
95           }
96         });
97       });
98
99       const topSkills = Object.entries(skillCounts)
100        .sort((a, b) => b[1] - a[1])
101        .slice(0, 10)
102        .map(([name, count]) => ({ name, count }));
103
104      data.topSkills = topSkills;
105
106      // Заявки по галузях
107      db.all(
108        `SELECT
109         i.name as industry_name,
110         COUNT(*) as count
111        FROM worker_applications wa
112        LEFT JOIN industries i ON wa.industry_id = i.id
113        WHERE wa.industry_id IS NOT NULL
114        GROUP BY wa.industry_id
115        ORDER BY count DESC`,
116        [],
117        (err, applicationsByIndustry) => {
118          data.applicationsByIndustry = applicationsByIndustry || [];
119
120          // Роботодавці по галузях
```

Рисунок 3.3 – Формування наборів даних для графіків у модулі статистики

З рисунка 3.3 видно, що підсистема агрегує дані з різних сутностей і повертає їх у єдиному структурованому форматі.

У клієнтській частині результати аналітики відображаються на адміністративному дашборді через виклики API та побудову графіків (Chart.js). Джерело: app.js (line 1490).

На рисунку 3.4 представлено клієнтську логіку завантаження показників та побудови графічних елементів дашборду [27].

```
const stats = await api('/stats/overview');
const chartData = await api('/stats/charts');

new Chart($('#status-chart'), { type: 'bar', data: ... });
new Chart($('#users-chart'), { type: 'bar', data: ... });
```

Рисунок 3.4 – Візуалізація аналітики на frontend

Як показано на рисунку 3.4, аналітичні дані інтегруються в інтерфейс адміністратора у вигляді карток KPI та діаграм.

Окремим елементом підтримки рішень є журнал системних подій (GET /api/stats/logs), який забезпечує аудит дій і пояснюваність змін. Джерело: stats.js (line 341), app.js (line 2209).

Для формалізації роботи підсистеми підтримки прийняття рішень побудовано UML-діаграму послідовності. На рисунку 3.5 показано цикл отримання та візуалізації аналітичних даних.

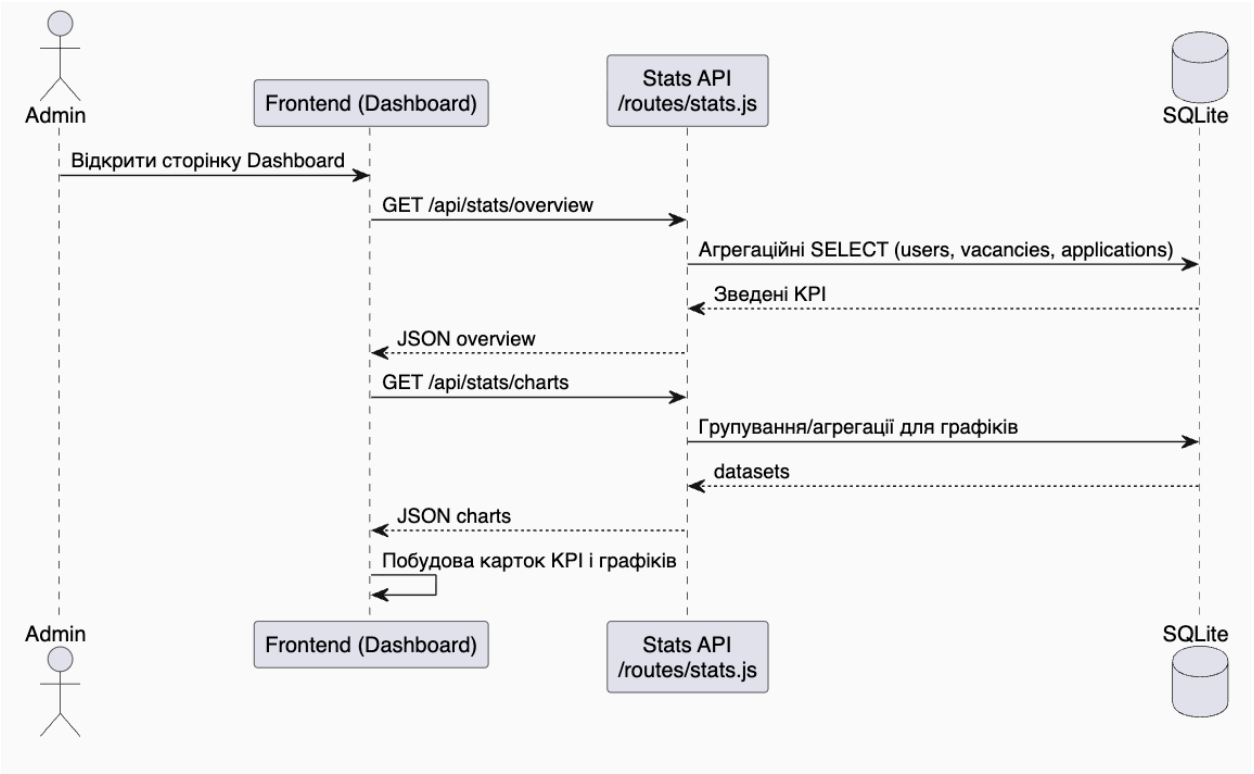

Рисунок 3.5 – UML-діаграма взаємодії модуля аналітики

Як видно з рисунка 3.5, підсистема аналітики реалізує повний контур підтримки рішень: від агрегації даних у БД до візуального представлення управлінських індикаторів [28].

3.3 Реалізація клієнтського інтерфейсу та взаємодії користувача з системою

У підрозділі 3.3 представлено практичну реалізацію клієнтського інтерфейсу системи та демонстрацію взаємодії користувача з основними функціональними модулями. Основну увагу зосереджено на побудові зрозумілого веб-інтерфейсу, що забезпечує зручний доступ до сценаріїв автентифікації, роботи з заявками, перегляду аналітики та адміністрування. Реалізація виконана засобами HTML, CSS та JavaScript із підключенням до REST API серверної частини.

Інтерфейс спроєктовано за принципами рольової взаємодії: після входу користувач отримує доступ лише до тих розділів, що відповідають його типу (працівник, роботодавець або адміністратор). Це дозволяє поєднати єдину

фронтенд-платформу з диференційованими бізнес-процесами, зменшити складність навігації та підвищити ергономіку використання системи. Обмін даними з сервером реалізовано через API-виклики з токен-автентифікацією, що забезпечує цілісність сеансу та захищений доступ до функцій.

У цьому підрозділі далі наведено послідовну демонстрацію інтерфейсних екранів і прикладів користувацьких сценаріїв у форматі рисунків із коротким поясненням їх призначення в загальній логіці системи [29].

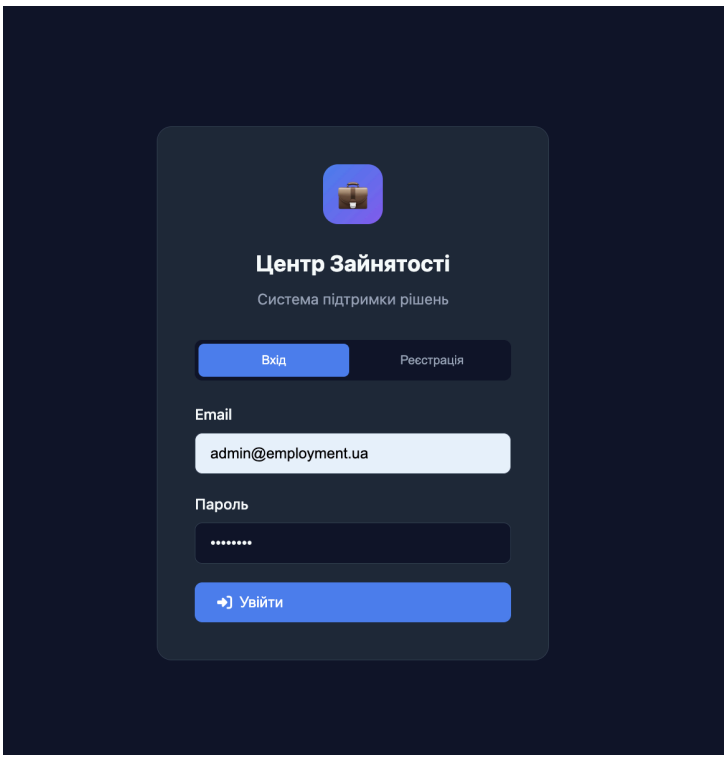

Рисунок 3.6 – Форма автентифікації користувача в системі «Центр Зайнятості»

На рисунку 3.6 представлено стартовий екран входу, який забезпечує доступ до системи за обліковими даними та перемикання між режимами входу і реєстрації.

На рисунку 3.7 представлено головний екран підсистеми аналітики, що використовується адміністратором для оперативного моніторингу стану платформи.

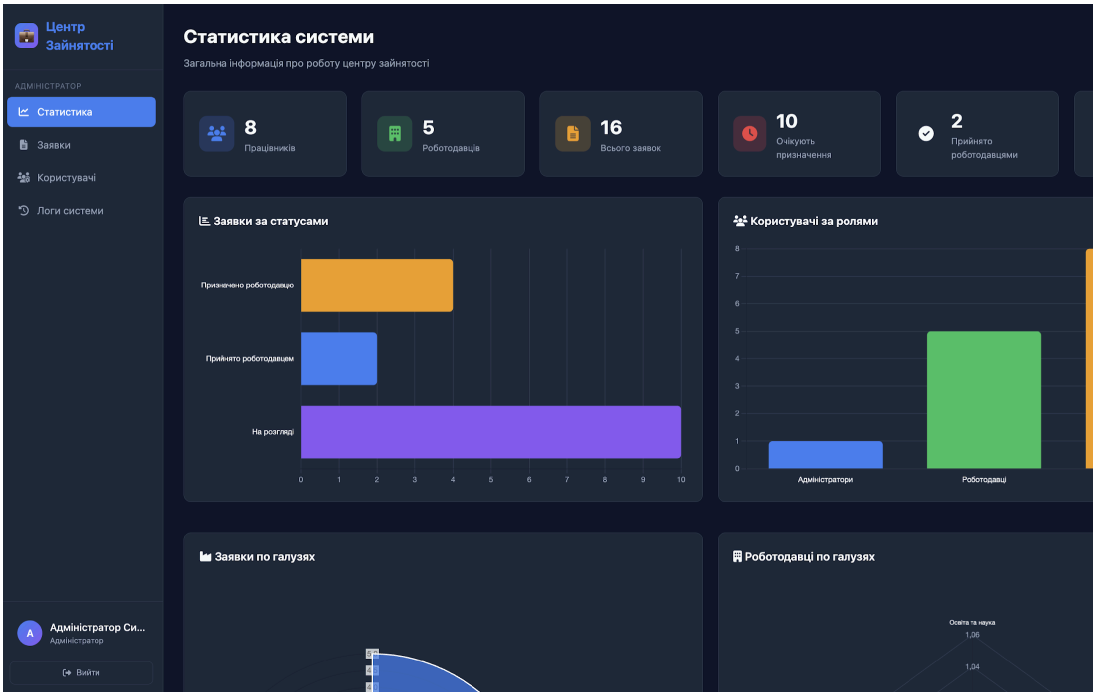

Рисунок 3.7 – Інформаційно-аналітична панель адміністратора системи

Як показано на рисунку 3.7, інтерфейс об’єднує KPI-картки та графічні візуалізації (статуси заявок, розподіл ролей, галузеві зрізи), що підтримує прийняття управлінських рішень на основі актуальних даних.

На рисунку 3.7 представлено сторінку перегляду всіх заявок із засобами пошуку та фільтрації за навичками, діапазоном заробітної плати, датою, галуззю та статусом.

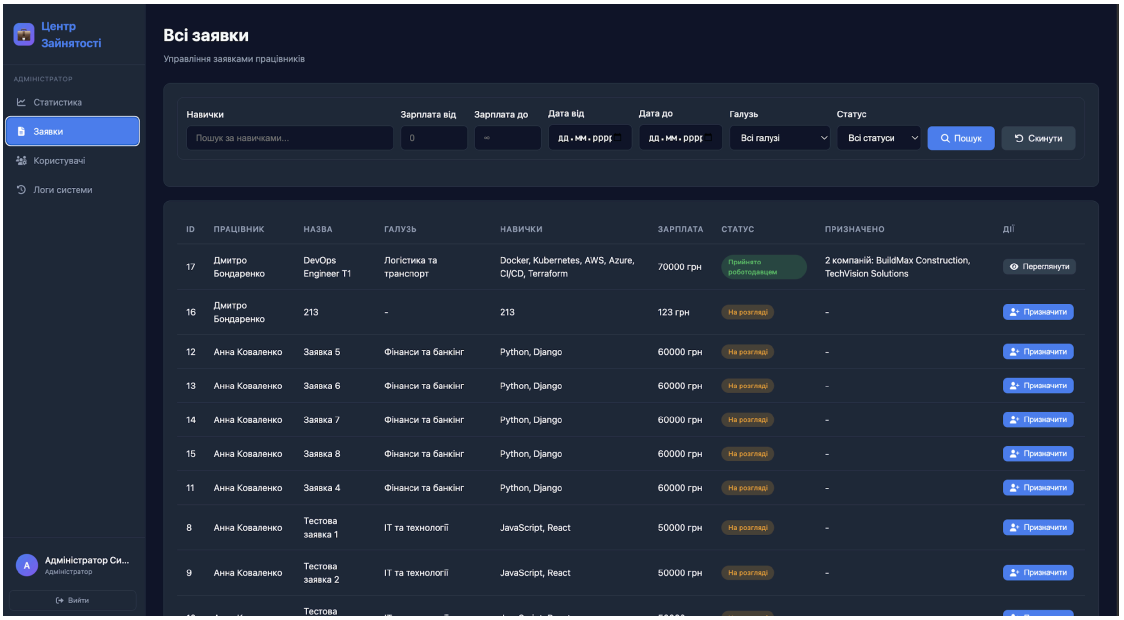

Рисунок 3.7 – Інтерфейс управління заявками працівників для адміністратора

Як показано на рисунку 3.7, табличне подання заявок підтримує операції адміністрування (перегляд і призначення роботодавцю), що забезпечує контроль етапів обробки кандидатів у межах єдиного робочого інтерфейсу.

На рисунку 3.8 представлено сторінку керування користувачами, що містить зведений перелік облікових записів із розподілом за ролями та вкладками категорій.

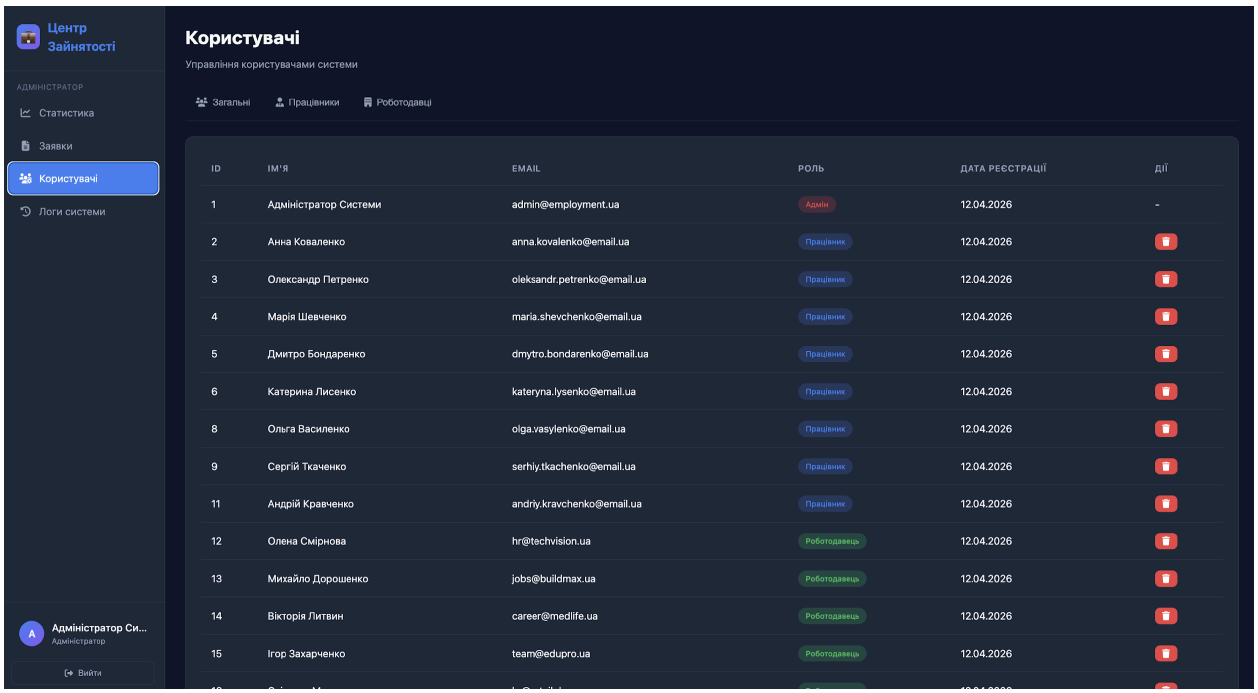

Рисунок 3.8 – Інтерфейс адміністрування користувачів системи

Як показано на рисунку 3.8, адміністратор має змогу контролювати склад користувачів, переглядати реєстраційні дані та виконувати адміністративні дії з обліковими записами в межах єдиного інтерфейсу.

На рисунку 3.9 представлено сторінку перегляду системних логів із фільтрацією за типом сутності та видом дії для аналізу змін у процесі обробки заявок.

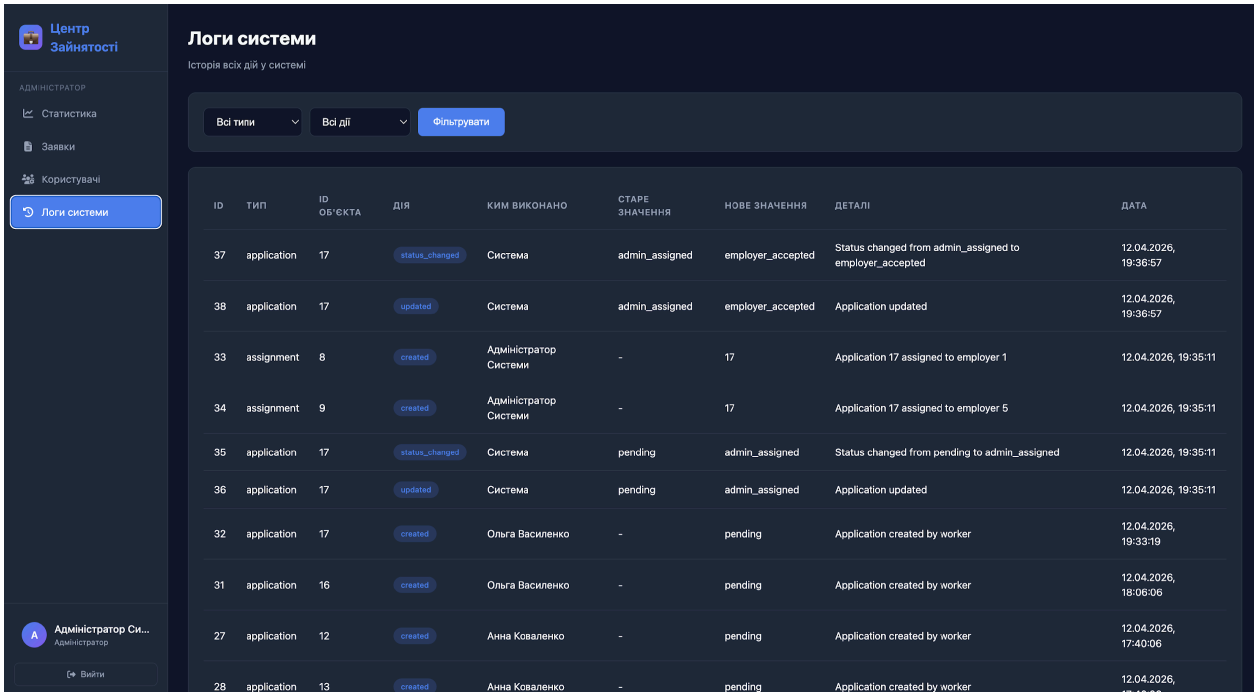

Рисунок 3.9 – Інтерфейс журналу подій (логів) системи

Як показано на рисунку 3.9, журнал подій забезпечує трасованість операцій у системі, відображаючи виконавця, попередні та нові значення, деталі операції й часові мітки.

Висновок до розділу 3

У третьому розділі було реалізовано практичну частину інформаційної системи «Центр зайнятості», що охоплює серверний рівень, підсистему аналітики та клієнтський інтерфейс. На рівні backend побудовано REST API на базі Node.js та Express із підключенням SQLite, реалізовано модулі автентифікації, рольового доступу та обробки ключових бізнес-сценаріїв (робота із заявками, призначеннями, користувачами). Така реалізація підтвердила працездатність обраної архітектури та можливість централізованого керування процесами працевлаштування.

У межах підрозділу, присвяченого підтримці прийняття рішень, впроваджено аналітичні API-методи для формування зведених показників, динамічних зрізів і службового журналу подій. Отримані дані використовуються для побудови дашбордів, що відображають стан заявок, ролі користувачів, галузеву структуру та інші управлінсько значущі метрики. Це забезпечує не лише оперативний моніторинг, а й основу для прийняття обґрунтованих адміністративних рішень на основі актуальної інформації.

Реалізація клієнтського інтерфейсу продемонструвала завершений цикл взаємодії користувачів із системою: від автентифікації до виконання прикладних операцій та контролю результатів. Інтерфейсні екрани для адміністратора підтверджують зручність доступу до функцій статистики, керування заявками, користувачами та логами, а також узгодженість frontend із серверними сервісами. Отже, у розділі 3 досягнуто мети практичної реалізації програмного продукту та підготовлено повноцінну основу для подальшого тестування й оцінювання ефективності системи.

ВИСНОВКИ

У дипломній роботі розглянуто актуальну задачу цифровізації процесів у сфері працевлаштування та обґрунтовано доцільність створення інформаційної системи підтримки прийняття рішень для центру зайнятості. Визначено практичну цінність автоматизації обліку заявок, взаємодії між працівниками, роботодавцями й адміністраторами, а також потребу в аналітичному супроводі управлінських рішень. Сформульовано мету, завдання та загальну логіку дослідження, що забезпечило послідовний перехід від теоретичного аналізу до реалізації програмного продукту.

У процесі виконання роботи проаналізовано предметну область, основні бізнес-процеси та рольові сценарії користувачів системи. На цій основі виокремлено ключові функціональні вимоги до програмного рішення: автентифікація, керування заявками, адміністрування користувачів, призначення роботодавців та моніторинг змін. Окрему увагу приділено формуванню нефункціональних вимог, зокрема модульності, надійності, зручності використання та підтримці подальшого розширення системи.

У межах проєктування досліджено архітектурні підходи та розроблено клієнт-серверну структуру застосунку з розподілом відповідальності між frontend і backend. Побудовано логіку взаємодії модулів, визначено маршрути API та принципи рольового доступу. Використання Node.js, Express і SQLite забезпечило практичну реалізацію системи з чіткою організацією серверних компонентів і централізованою обробкою даних.

Для підтримки цілісності даних розроблено реляційну модель бази даних, що охоплює сутності користувачів, профілів, вакансій, резюме, заявок, призначень і журналу подій. Проаналізовано зв’язки між таблицями, статусну модель обробки заявок та механізми фіксації змін у системі. Реалізований REST API забезпечує стабільний обмін даними між клієнтською та серверною частинами, а також коректне виконання основних операцій у межах предметної області.

Важливим результатом є те, що розроблено підсистему аналітики та підтримки прийняття рішень, яка формує зведені показники, статистичні зрізи та історію дій користувачів. Це дозволило перейти від простого накопичення даних до їх практичного використання в управлінні процесами працевлаштування. У клієнтському інтерфейсі реалізовано наочне відображення KPI, графіків і таблиць, що підвищує швидкість орієнтації адміністратора в поточному стані системи.

Отже, поставлену мету дипломної роботи досягнуто: розглянуто, проаналізовано, досліджено та розроблено повнофункціональне програмне рішення для підтримки діяльності центру зайнятості. Отримані результати мають прикладне значення та можуть бути використані як основа для подальшого розвитку системи, зокрема розширення аналітики, підвищення рівня безпеки, інтеграції з зовнішніми сервісами та масштабування на більші обсяги користувачів і даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Doctoral dissertation. – Irvine : University of California, 2000. – 162 p.
2. Richardson L., Amundsen M. RESTful Web APIs. – Sebastopol : O'Reilly Media, 2013. – 408 p.
3. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs // IEEE Internet Computing. – 2010. – Vol. 14, No. 6. – P. 80–83.
4. Node.js Documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/docs/latest/api/ (дата звернення: 21.04.2026).
5. Express.js Documentation [Електронний ресурс]. – Режим доступу: https://expressjs.com/ (дата звернення: 21.04.2026).
6. SQLite Documentation [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html (дата звернення: 21.04.2026).
7. Hipp R. D. SQLite. Lightweight SQL Database Engine [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/about.html (дата звернення: 21.04.2026).
8. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT): RFC 7519 [Електронний ресурс]. – IETF, 2015. – Режим доступу: https://www.rfc-editor.org/rfc/rfc7519 (дата звернення: 21.04.2026).
9. OWASP Foundation. OWASP Top 10:2021 – The Ten Most Critical Web Application Security Risks [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-top-ten/ (дата звернення: 21.04.2026).
10. Mozilla Developer Network. Fetch API [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/docs/Web/API/Fetch_API (дата звернення: 21.04.2026).
11. Mozilla Developer Network. Web Storage API [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/docs/Web/API/Web_Storage_API (дата звернення: 21.04.2026).

12. Chart.js Documentation [Електронний ресурс]. – Режим доступу: https://www.chartjs.org/docs/latest/ (дата звернення: 21.04.2026).

13. ISO/IEC 25010:2011. Systems and software engineering – System and software quality models. – Geneva : ISO, 2015.

14. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. – 9th ed. – New York : McGraw-Hill, 2019. – 992 p.

15. Turban E., Sharda R., Delen D. Decision Support and Business Intelligence Systems. – 10th ed. – Pearson, 2015. – 720 p.

16. Power D. J. Decision Support Systems: Concepts and Resources for Managers. – 2nd ed. – Praeger, 2015. – 272 p.

17. Laudon K. C., Laudon J. P. Management Information Systems: Managing the Digital Firm. – 15th ed. – Pearson, 2018. – 688 p.

18. Marakas G. M. Decision Support Systems in the 21st Century. – Prentice Hall, 2017. – 560 p.

19. Sommerville I. Software Engineering. – 10th ed. – Pearson, 2016. – 816 p.

20. ISO/IEC 27001:2013 Information Security Management Systems. – Geneva : ISO, 2015.

21. NIST. Big Data Interoperability Framework [Електронний ресурс]. – Режим доступу: https://www.nist.gov/programs-projects/nist-big-data-program (дата звернення: 22.04.2026).

22. World Bank. World Development Report 2016: Digital Dividends [Електронний ресурс]. – Режим доступу: https://www.worldbank.org/en/publication/wdr2016 (дата звернення: 22.04.2026).

23. European Commission. Labour Market Analysis [Електронний ресурс]. – Режим доступу: https://ec.europa.eu/social (дата звернення: 22.04.2026).

24. LinkedIn. Global Talent Trends Report [Електронний ресурс]. – Режим доступу: https://business.linkedin.com (дата звернення: 22.04.2026).

25. Work.ua. Офіційний сайт платформи працевлаштування [Електронний ресурс]. – Режим доступу: https://www.work.ua (дата звернення: 22.04.2026).

26. Robota.ua. Офіційний сайт платформи працевлаштування [Електронний ресурс]. – Режим доступу: https://robota.ua (дата звернення: 22.04.2026).

27. Djinni. IT Recruitment Platform [Електронний ресурс]. – Режим доступу: https://djinni.co (дата звернення: 22.04.2026).

28. MDN Web Docs. JavaScript Guide [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 22.04.2026).

29. React Documentation [Електронний ресурс]. – Режим доступу: https://react.dev (дата звернення: 22.04.2026).

30. Google Developers. Web Fundamentals [Електронний ресурс]. – Режим доступу: https://developers.google.com/web (дата звернення: 22.04.2026).

ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

«Система підтримки рішень у сфері зайнятості населення»

Виконав: здобувач вищої освіти гр. САД-41 Тимофій ФЕДОРЧУК

Керівник: к. ф.-м. н., старший викладач Ровіл НАФЄЄВ

Київ – 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи

Розробка системи підтримки рішень у сфері зайнятості населення, яка забезпечує ефективну взаємодію між шукачами роботи, роботодавцями та адміністраторами, а також підвищує якість підбору кандидатів і вакансій.

Об'єкт дослідження

Інформаційні системи підтримки прийняття рішень у сфері зайнятості населення.

Предмет дослідження

Методи та засоби проєктування і реалізації веб-системи для автоматизації процесів працевлаштування, аналізу даних ринку праці та підтримки прийняття управлінських рішень.

НАУКОВІ ЗАВДАННЯ ТА МЕТОДИ ДОСЛІДЖЕННЯ

Наукові завдання:

1. Дослідження сучасних тенденцій розвитку інформаційних систем у сфері зайнятості

2. Огляд методів обробки та аналізу даних ринку праці (вакансії, резюме, заявки)

3. Аналіз існуючих платформ: Work.ua, Rabota.ua, LinkedIn, Djinni

4. Формування функціональних та нефункціональних вимог до системи

5. Проєктування клієнт-серверної архітектури, API та структури бази даних

6. Реалізація серверної частини (Node.js, Express, SQLite, JWT)

7. Розробка підсистеми аналітики та клієнтського інтерфейсу

Методи дослідження:

- Аналіз наукових та технічних джерел
- Системний аналіз та моделювання бізнес-процесів
- Методи проєктування ІС (архітектура, БД, API)
- Розробка та реалізація програмного продукту
- Оцінювання ефективності системи

СИСТЕМИ ПІДТРИМКИ РІШЕНЬ У СФЕРІ ЗАЙНЯТОСТІ

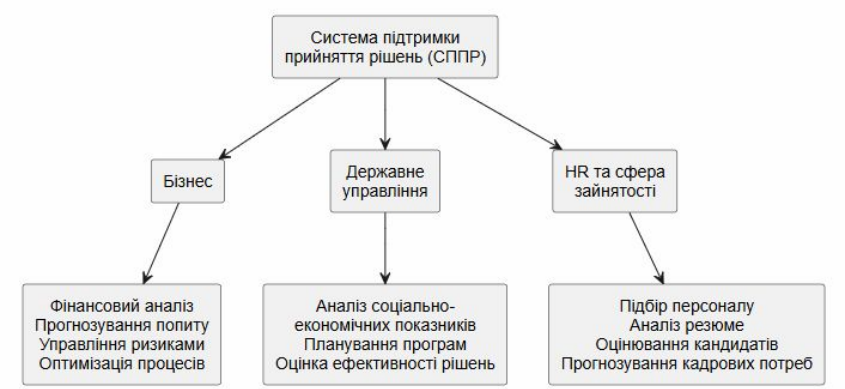

Рисунок 1 – Сфери застосування систем підтримки прийняття рішень

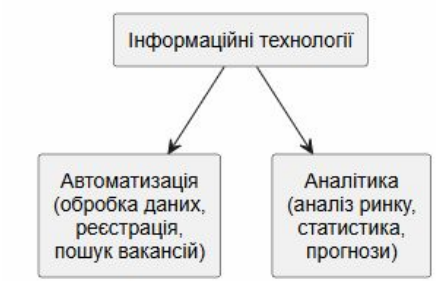

Рисунок 2 – Роль інформаційних технологій у системах зайнятості

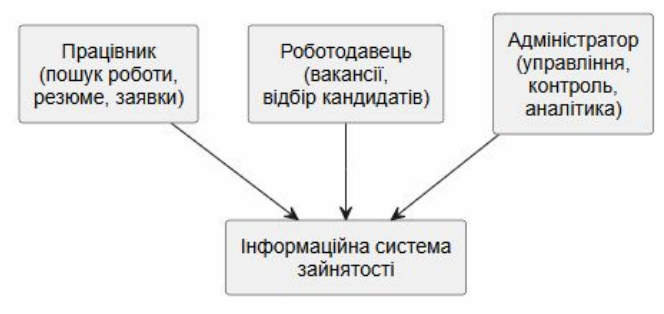

Рисунок 3 – Основні користувачі інформаційної системи зайнятості

АНАЛІЗ ІСНУЮЧИХ ПЛАТФОРМ ПРАЦЕВЛАШТУВАННЯ

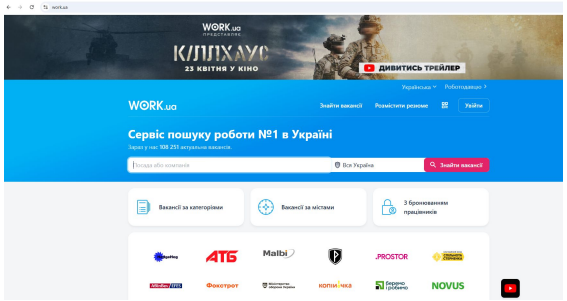

Рисунок 4 – Work.ua

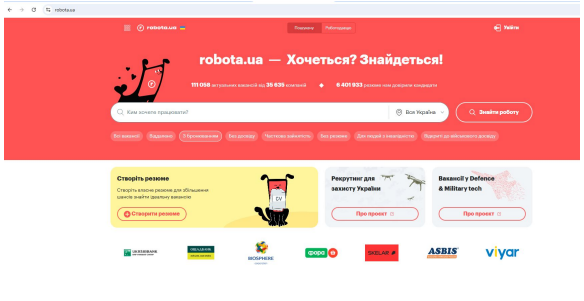

Рисунок 5 – Rabota.ua

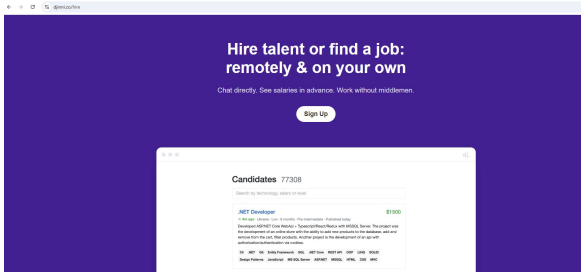

Рисунок 6 – Djinni

Таблиця 1 – Порівняння платформ працевлаштування

Платформа	Тип	Аналітика	Ролі	Переваги	Недоліки
Work.ua	Комерційна	Обмежена	Шукач/Роботодавець	Широка база, UI	Платна для роботодавців
Rabota.ua	Комерційна	Базова	Шукач/Роботодавець	Популярна в Україні	Менша аналітика
LinkedIn	Глобальна	Широка	Профілі/Компанії	Соціальний нетворкінг	Перевантажений UI
Djinni	IT-спеціалізована	IT-аналітика	Анонімний пошук	Фокус на IT-ринку	Вузька галузь
Розроблена СПР	Центр зайнятості	Повна (KPI)	Адмін/Шукач/Роботодавець	Рольова модель, СППР	Масштабування

ВИМОГИ ТА АРХІТЕКТУРА СИСТЕМИ

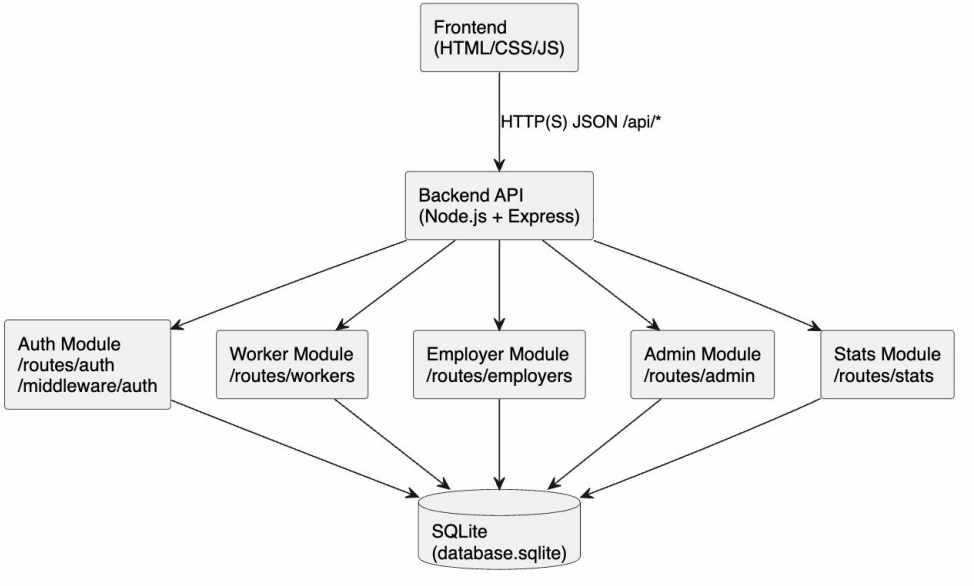

Рисунок 7 – UML компонентна діаграма архітектури системи

Функціональні вимоги:

• Автентифікація (реєстрація, вхід, JWT, рольовий доступ)

• Управління профілями (шукачі, роботодавці, адміністратор)

• Робота з вакансіями, резюме та заявками на роботу

• Адміністрування: призначення, моніторинг, журнал подій

• Аналітика: KPI, статистика, графіки (Chart.js)

Нефункціональні вимоги:

• Модульність, масштабованість, надійність

• JWT-автентифікація, HTTPS, захист даних

• Адаптивний інтерфейс, час відповіді < 2 с

СТРУКТУРА БАЗИ ДАНИХ ТА API-ВЗАЄМОДІЯ

```
CREATE TABLE IF NOT EXISTS worker_applications (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  worker_id INTEGER NOT NULL,
  ...
  status TEXT DEFAULT 'pending' CHECK(status IN (
    'pending','admin_assigned','employer_viewed','employer_accepted',
    'employer_rejected','completed','cancelled'
  )),
  FOREIGN KEY (worker_id) REFERENCES workers(id) ON DELETE CASCADE
)

CREATE TABLE IF NOT EXISTS admin_assignments (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  application_id INTEGER NOT NULL,
  employer_id INTEGER NOT NULL,
  admin_id INTEGER NOT NULL,
  ...
  UNIQUE(application_id, employer_id)
)
```

Рисунок 8 – UML-діаграма структури бази даних системи

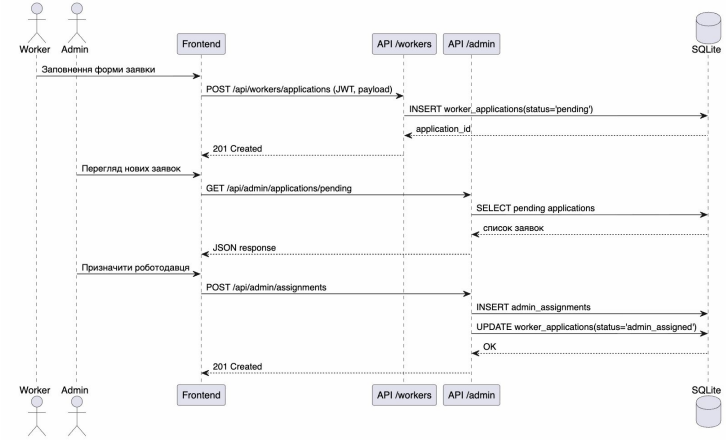

Рисунок 9 – UML-діаграма послідовності API-взаємодії

Таблиця БД	Основні поля	Призначення
users	id, role, email, password_hash	Облікові записи всіх ролей (worker/employer/admin)
workers	id, skills, experience, education, salary	Профіль шукача роботи
employers	id, company_name, industry, website	Профіль роботодавця
worker_applications	id, worker_id, status, assigned_at	Заявки на роботу + статусна модель
system_logs	id, entity_type, action, performed_by	Журнал усіх дій у системі (аудит)

ТЕХНОЛОГІЧНИЙ СТЕК РЕАЛІЗАЦІЇ

Таблиця 2 – Технологічний стек серверної та клієнтської частини системи

Рівень	Технологія	Версія	Призначення
Backend	Node.js + Express	18.x / 4.x	Серверна логіка, маршрутизація REST API
Backend	SQLite + better-sqlite3	—	Реляційна БД для зберігання всіх даних системи
Backend	JSON Web Token (JWT)	RFC 7519	Автентифікація та авторизація за ролями
Backend	bcrypt	—	Хешування паролів перед збереженням у БД
Backend	cors, dotenv	—	CORS-middleware та управління змінними середовища
Frontend	Vanilla JS + HTML5 + CSS3	—	SPA-інтерфейс без тяжких фреймворків
Frontend	Chart.js	—	Візуалізація аналітики (KPI, графіки, статистика)
Frontend	Fetch API	—	HTTP-запити до REST API з клієнтської частини
Архітектура	REST API	—	JSON-обмін між клієнтом та сервером
Безпека	JWT + HTTPS + Middleware	—	Рольовий доступ (admin/worker/employer)

REST API СИСТЕМИ «ЦЕНТР ЗАЙНЯТОСТІ»

Таблиця 3 – Основні REST API endpoints системи підтримки рішень

Модуль	Метод	Endpoint	Опис
Авторизація	POST	/api/auth/register	Реєстрація нового користувача (worker/employer)
Авторизація	POST	/api/auth/login	Вхід у систему, отримання JWT-токена
Працівник	GET	/api/workers/applications	Список заявок поточного шукача
Працівник	POST	/api/workers/applications	Подача нової заявки на роботу
Роботодавець	GET	/api/employers/applications	Перегляд заявок до вакансій
Адміністратор	POST	/api/admin/assign	Призначення заявки роботодавцю
Адміністратор	GET	/api/admin/users	Список усіх користувачів системи
Адміністратор	GET	/api/admin/logs	Журнал подій (аудит дій)
Статистика	GET	/api/stats/overview	KPI: кількість заявок, вакансій, користувачів
Статистика	GET	/api/stats/charts	Дані для графіків (Chart.js)

ПІДСИСТЕМА АНАЛІТИКИ ТА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

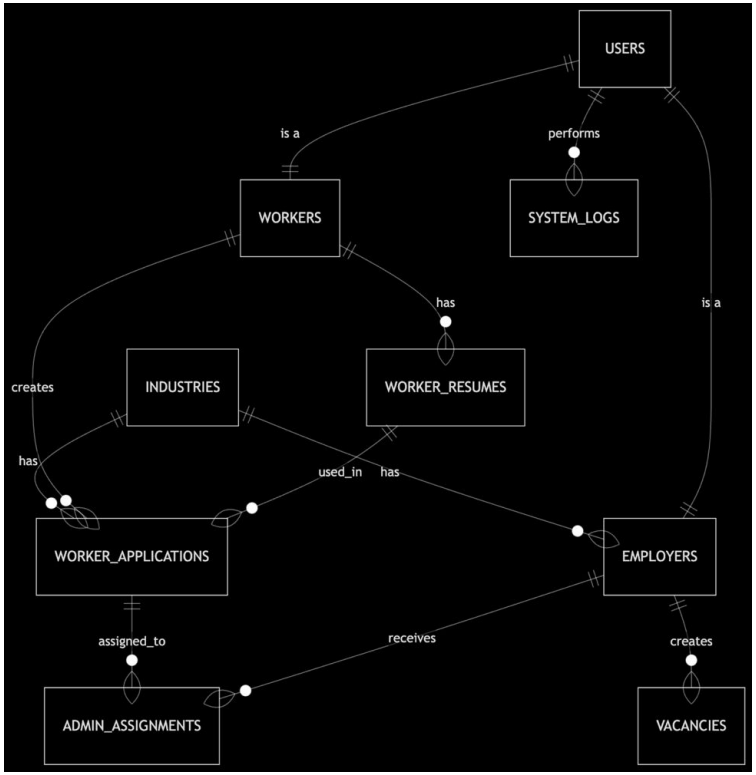

Рисунок 10 – UML-діаграма взаємодії модуля аналітики та СППР

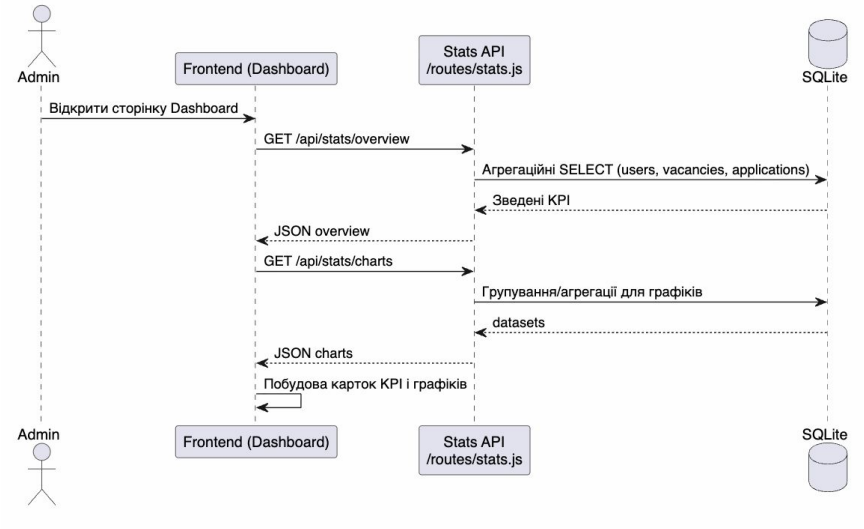

Рисунок 11 – Візуалізація аналітики: KPI та графіки на frontend

• KPI-показники: загальна кількість заявок, активних вакансій, зареєстрованих користувачів

• Статистичні зрізи: розподіл заявок за статусами (pending/assigned/accepted/rejected/completed)

• Графіки Chart.js: динаміка заявок, галузевий аналіз, ефективність підбору кандидатів

ДЕМОНСТРАЦІЯ ІНТЕРФЕЙСУ СИСТЕМИ «ЦЕНТР ЗАЙНЯТОСТІ»

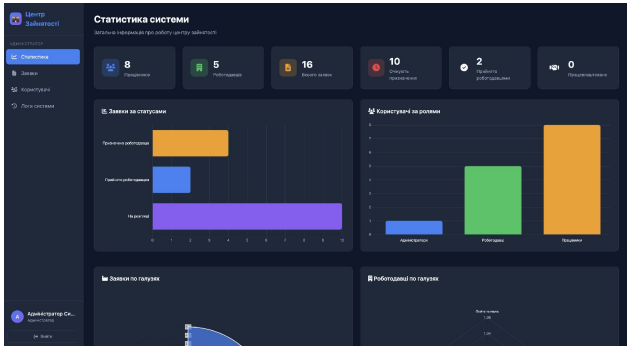

Рисунок 12 – Форма автентифікації

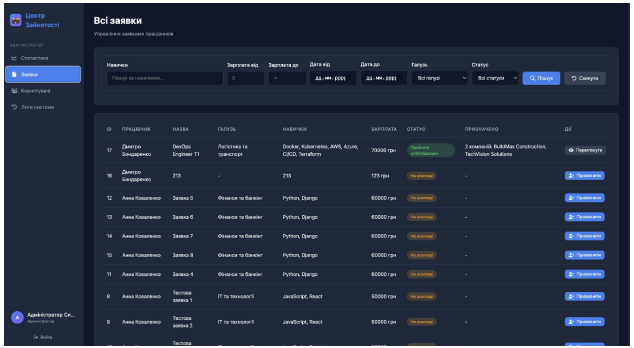

Рисунок 13 – Дашборд адміністратора

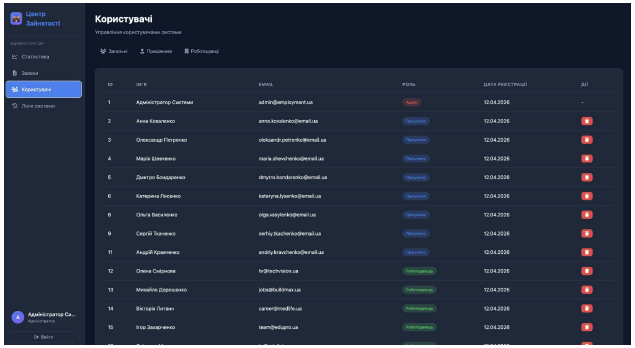

Рисунок 14 – Управління заявками

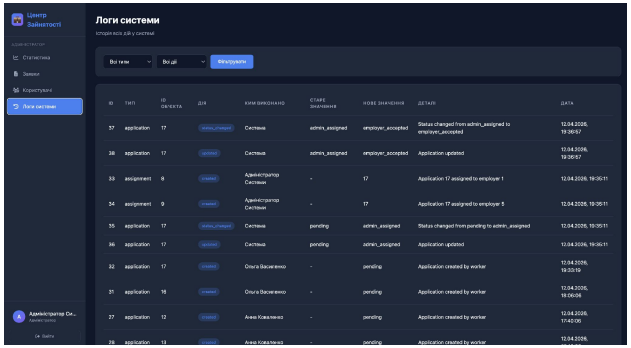

Рисунок 15 – Управління користувачами та логи

ВИСНОВКИ

1. Досліджено теоретичні засади СППР та інформаційних систем зайнятості; проаналізовано сучасні платформи (Work.ua, Rabota.ua, LinkedIn, Djinni) та визначено напрями розробки власної системи для центру зайнятості.

2. Сформовано функціональні та нефункціональні вимоги; спроєктовано трирівневу клієнт-серверну архітектуру з рольовим доступом (адміністратор / шукач роботи / роботодавець).

3. Розроблено реляційну модель бази даних SQLite: users, workers, employers, worker_applications, system_logs із визначеними зв'язками, статусною моделлю та механізмами аудиту.

4. Реалізовано REST API (Node.js/Express): модулі /auth, /workers, /employers, /admin, /stats із JWT-автентифікацією, bcrypt-хешуванням паролів та CORS-підтримкою.

5. Розроблено підсистему аналітики та СППР: KPI-показники, статистичні зрізи, графіки Chart.js — перехід від накопичення даних до управлінської підтримки рішень.

6. Реалізовано клієнтський інтерфейс (Vanilla JS + HTML5 + CSS3) з автентифікацією, управлінням заявками, аналітичною панеллю та журналом подій для адміністратора.

7. Практична цінність роботи — повнофункціональне програмне рішення для центру зайнятості, готове до подальшого розвитку: розширення аналітики, інтеграції зовнішніх сервісів та масштабування.

ДЯКУЮ ЗА УВАГУ!