

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ КИЇВСЬКИЙ
НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»**

Кафедра інформаційних систем в економіці

**ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА
«Інформаційні управляючі системи і технології»**

галузь знань
спеціальність

12 Інформаційні технології
122 Комп'ютерні науки

Форма навчання: заочна

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

**«Розробка HRM-системи для оптимізації управління персоналом і внутрішніх процесів
компанії»**

здобувача Селіверстова Петра Сергійовича

Науковий керівник: д.е.н., професор Устенко С.В.

(підпис)

Робота допущена до захисту передекзаменаційною
комісією з атестації здобувачів вищої освіти (ЕК)

Завідувач кафедри: к.е.н., доцент Тішков Б. О.

(підпис)

Київ 2025

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА**

**Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці**

**ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА
«Інформаційні управляючі системи і технології»**

галузь знань
спеціальність

12 Інформаційні технології
122 Комп'ютерні науки

ПОГОДЖЕНО:
Керівник проєктної групи (гарант)
освітньо-професійної програми

ЗАТВЕРДЖУЮ:
Завідувач кафедри
інформаційних систем в економіці

_____ Устенко С.В.
«23» січня 2024 р.

_____ Тішков
Б.О.
«23» січня 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти _____ **Селіверстова Петра Сергійовича**
(прізвище, ім'я, по батькові)
_____ заочної _____ форми навчання
очної (денної), заочної, дистанційної

на підготовку кваліфікаційної магістерської роботи
**на тему: «Розробка HRM-системи для оптимізації управління
персоналом і внутрішніх процесів компанії»**

Тему затверджено наказом ректора Університету від «07» березня 2025р. №465-ст

Кваліфікаційна магістерська робота виконується на матеріалах

План кваліфікаційної магістерської роботи

Розділ I. Дослідження і аналіз уже існуючих HRM-систем

Розділ II. Характеристика HRM-системи та визначення її архітектури

Розділ III. Реалізація функціональних компонентів системи та оцінка ефективності

Об’єкт дослідження: HRM-система як інструмент цифрового управління персоналом.

Предмет дослідження: методологія проєктування та реалізації HRM-систем із урахуванням сучасних архітектурних підходів

Мета кваліфікаційної магістерської роботи: розробка сучасної HRM-системи, здатної забезпечити повний цикл управління персоналом у компанії, включаючи рекрутинг, адаптацію, облік робочого часу, оцінку продуктивності, навчання та кадрове адміністрування. Система має відповідати сучасним вимогам до гнучкості, масштабованості, безпеки та інтеграційної сумісності, а також враховувати специфіку малого та середнього бізнесу. У роботі передбачено формування архітектури, проєктування функціональних модулів, визначення вимог до їхньої реалізації та опис сценаріїв впровадження для підвищення ефективності управлінських процесів компанії.

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі I здійснити огляд сучасного стану та тенденцій розвитку HRM-систем, охарактеризувати ключові функціональні компоненти, моделі впровадження та технологічні особливості існуючих рішень.

У розділі II визначити архітектурний підхід до побудови HRM-системи, обґрунтувати вибір інструментів, описати логічну структуру, функціональність та модель даних.

У розділі III реалізувати обрані модулі системи, представити UML-діаграми та ER-модель, а також надати приклади сценаріїв взаємодії користувачів із системою

**Завдання підготував
науковий керівник**

С.В. Устенко

(підпис)

«13» травня 2025 р.

**Завдання одержав
здобувач**

П.С. Селіверстов
«13» травня 2025 р

(підпис)

Реферат

Кваліфікаційна магістерська робота містить 119 сторінок, 9 таблиць, 25 рисунків, список використаних джерел із 59 найменувань.

«Розробка HRM-системи для оптимізації управління персоналом і внутрішніх процесів компанії»

Об'єкт дослідження: HRM-система як інструмент управління персоналом та бізнес-процесами в організації.

Предмет дослідження: включає комплексне дослідження методів, архітектурних рішень і підходів до проектування інформаційних систем управління персоналом у цифровому середовищі.

Мета та завдання дослідження: Основною метою цієї кваліфікаційної магістерської роботи є розробка HRM-системи, здатної автоматизувати ключові етапи управління персоналом – від підбору кадрів до оцінки їхньої ефективності, – з урахуванням вимог до масштабованості, безпеки, адаптивності та інтеграції. Для досягнення цієї головної мети були визначені такі завдання:

- здійснено аналіз сучасних HRM-рішень на ринку та їх функціональних особливостей;
- визначено вимоги до системи: функціональні, нефункціональні та нормативні;
- спроектовано архітектуру мікросервісної HRM-системи з використанням сучасних підходів і стандартів;
- побудовано модель даних, діаграми класів, компонентів, процесів та сценаріїв;
- визначено технічні засоби реалізації, зокрема засоби контейнеризації, API-шари, засоби інтеграції з зовнішніми системами;
- описано процеси рекрутингу, адаптації, обліку часу, оцінювання й адміністрування персоналу в межах розробленої системи.

Теоретичне, методологічне та практичне значення отриманих результатів. У роботі реалізовано сучасний підхід до побудови HRM-системи на основі мікросервісної архітектури з підтримкою масштабованості, безпеки та інтеграції з корпоративною інфраструктурою. Представлена концепція може бути використана як основа для впровадження цифрових HR-рішень у малих та середніх компаніях, які потребують автоматизації процесів управління персоналом.

Рік виконання кваліфікаційної магістерської роботи – 2025. Рік захисту роботи – 2025.

Ключові слова: HRM, управління персоналом, інформаційна система, мікросервісна архітектура, автоматизація бізнес-процесів, кадровий облік.

Відгук

про кваліфікаційну магістерську роботу

здобувача навчально-наукового інституту «Інститут інформаційних технологій в економіці»

освітньо-професійної програми «Інформаційні управляючі системи і технології»

Селіверстова Петра Сергійовича

(прізвище, ім'я, по батькові)

на тему «Розробка HRM-системи для оптимізації управління персоналом і внутрішніх процесів компанії»

1. Актуальність теми: Тема роботи є дуже актуальною через цифрову трансформацію бізнесу. Впровадження HRM-систем автоматизує управління персоналом, підвищує ефективність, знижує витрати та покращує управлінські рішення, що особливо важливо для малого та середнього бізнесу.

2. Позитивні риси кваліфікаційної магістерської роботи: Робота містить ґрунтовний аналіз сучасних HRM-систем, чітке визначення вимог та застосування мікросервісної архітектури. Автор успішно змодельював бізнес-процеси, створив UML- та ER-діаграми та обґрунтував вибір технологій. Практична частина включає реалізацію ключових модулів, що робить дослідження цінним для впровадження.

3. Наявність самостійних розробок автора: Автор самостійно здійснив аналіз існуючих HRM-рішень, розробив детальну архітектуру системи, включаючи компонентні та процесні діаграми, а також описав технічні аспекти реалізації. Значну увагу приділено безпеці, масштабованості та інтеграції системи.

4. Результати мають високу теоретичну і практичну значущість. Запропонована архітектура відповідає сучасним стандартам і може застосовуватися в різних організаціях, сприяючи підвищенню ефективності управління та прозорості бізнес-процесів.

5. Наявність недоліків: Недоліків у роботі не виявлено. Вона виконана на високому рівні з урахуванням усіх основних вимог до сучасних інформаційних систем управління.

6. Загальна оцінка кваліфікаційної магістерської роботи та її допущення до захисту перед ЕК: Робота виконана на високому науковому та практичному рівні, відповідає вимогам освітньої програми та стандартам підготовки магістерських робіт. Рекомендується для захисту перед екзаменаційною комісією.

Науковий керівник д.е.н., професор Устенко С.В.

(посада, учене звання, науковий ступінь)

(підпис)

(прізвище, ініціали)

«__» _____ 20__ р.

Рецензія

на кваліфікаційну магістерську роботу
здобувача вищої освіти

Селіверстова Петра Сергійовича

(прізвище, ім'я, по батькові)

Тема Розробка HRM-системи для оптимізації управління персоналом і внутрішніх процесів компанії

Актуальність теми кваліфікаційної магістерської роботи і доцільність її розроблення: Сучасний бізнес усе більше орієнтується на цифровізацію управлінських процесів, тому розробка ефективної HRM-системи є важливим кроком для підвищення конкурентоспроможності організації. Робота Петра має безпосереднє практичне значення для підприємства, де він проходив практику, оскільки впровадження автоматизованої системи управління персоналом здатне значно покращити внутрішні процеси.

1. Якість проведеного дослідження: Дипломна робота вирізняється ґрунтовним дослідженням актуальних рішень у сфері HRM, а також детальним технічним опрацюванням архітектури системи. Автор проявив високий рівень самостійності в аналізі вимог, моделюванні бізнес-процесів і виборі технологічного стеку. Практична частина відповідає реальним потребам підприємства, що свідчить про тісний зв'язок наукової роботи з виробничою діяльністю.

2. Позитивні риси кваліфікаційної магістерської роботи: Робота відзначається чіткою структурою, послідовністю викладу та продуманістю функціональних модулів системи. Запропоноване рішення враховує масштабованість, безпеку та інтеграцію з існуючими ІТ-інфраструктурами. Особливо цінним є практичний досвід, отриманий під час проходження практики, що відобразився у конкретних рекомендаціях для поліпшення роботи підприємства.

3. Зауваження: За результатами ознайомлення з роботою і практичним досвідом співпраці, суттєвих недоліків у роботі не виявлено. Всі поставлені завдання виконані якісно та вчасно.

4. Практична значимість висновків і рекомендацій: Результати роботи мають безпосереднє значення для підприємства, де здобувач проходив практику, та можуть бути використані для подальшого розвитку інформаційної системи управління персоналом. Впровадження запропонованих рішень сприятиме підвищенню ефективності кадрового обліку, оптимізації робочих процесів і покращенню комунікації між відділами.

Місце роботи та посада рецензента ФОП Селіверстов Петро Сергійович
Науковий ступінь, учене звання (за наявності) _____

(підпис, ПІБ)

Підпис засвідчую: _____

(посада, підпис)

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1. ДОСЛІДЖЕННЯ І АНАЛІЗ ВЖЕ ІСНУЮЧИХ HRM СИСТЕМ	5
1.1 Аналіз існуючих інформаційних систем	5
1.1.1 Дослідження предметної області	5
1.1.2 Дослідження інформаційних управляючих систем.....	7
1.2 Обґрунтування вибору підходів і технологій для створення інформаційної управляючої системи	13
РОЗДІЛ 2. ХАРАКТЕРИСТИКА HRM СИСТЕМ.....	18
2.1 Структура і характеристика системи	18
2.2 Архітектура системи.....	21
2.2.1 Функціональні характеристики	29
2.2.2 Нефункціональні характеристики	39
2.2.3 Технологічний стек.....	41
РОЗДІЛ 3. РОЗРОБКА ТА ПРОЄКТУВАННЯ HRM СИСТЕМ.....	59
3.1. Архітектура системи.....	59
3.2 Проєктування бази знань інформаційної управляючої системи та/або засоби інтелектуального аналізу даних.....	64
3.2.1 Структура бази знань.....	64
3.2.2 Функціональне моделювання системи	66
3.3. Проєктування інтерфейсу користувача	71
3.4. Реалізація інформаційної системи.....	75
ВИСНОВКИ.....	79
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	81
ДОДАТКИ.....	85

ВСТУП

Значимість цієї теми обумовлена швидким цифровим перетворенням, що охоплюють всі сфери діяльності сучасних організацій. Покращення ефективності управління людським капіталом мають вирішальне значення в конкурентному середовищі, оскільки автоматизація кадрових процесів – включаючи практики працевлаштування і адаптації – сприяють пришвидшенню процесу прийняття рішень, зниженню операційних витрат і позитивно впливають на прозорість внутрішнього механізму. Проте наявність на ринку рішень широко не задовольняють потреби малих та середніх бізнес-компаній згодом ставить питання перед необхідністю створити модульні, безпечні та масштабовані системи управління людськими ресурсами.

Метою цього дослідження є розробка концепційно нової архітектури та створення прототипу системи управління людськими ресурсами (HRM), яка об'єднуватиме принципи мікросервісного підходу разом з використанням хмарних технологій і високих стандартів безпеки з метою оптимізації управління персоналом та внутрішнього бізнесу компаній.

У рамках цілей цього проекту було спрямовано на комплексний аналіз існуючих рішень у сфері управління людськими ресурсами. Була проведена ідентифікація ключових функціональних та нефункціональних вимог. Розроблено загальну архітектуру з детальним описом топологічностей та точок інтеграцій. Були спроектовані основні модулі з використанням UML діаграм. Створено модель даних за допомогою ER діаграми. Також було обґрунтовано вибране технологічне стеко та засоби для розгортання. Подібний підхід дозволив виконати дослідження в унісон із сучасними стандартами ISO 25010 та OWASP Top 10, а також врахувати вимоги нормативно-правових актів України у сфері захисту персональних даних.

У роботі використано методи системного аналізу та синтезу теоретичних і практичних даних, UML-моделювання для формалізації структури та поведінки компонентів, кейс-стаді для вивчення бізнес-вимог середніх компаній, формальні специфікації API (JSON Schema, OpenAPI) та порівняльний аналіз технологій.

Застосування цих методів сприяло чіткій структуризації матеріалу та обґрунтуванню прийнятих технічних рішень.

Наукова новизна дослідження полягає в розробці адаптивної мікросервісно-BFF архітектури HRM-системи з підтримкою SCORM/xAPI та вбудованою BI-аналітикою, що враховує українські та міжнародні вимоги до безпеки та захисту даних. Практичне значення роботи полягає в готовій архітектурній моделі й рекомендаціях щодо впровадження, що можуть бути використані IT-відділами для модернізації наявних або створення нових HRM-рішень.

Структурно магістерська робота складається з чотирьох розділів. Перший розділ визначає актуальність, мету, завдання, методи та наукову новизну дослідження. Другий розділ аналізує та описує архітектурні, функціональні та нефункціональні характеристики системи, технологічний стек і модель даних. Третій розділ присвячений практичній частині — UML- та business-логіці сценаріїв, а також інтеграційним діаграмам. Четвертий розділ містить підсумкові висновки та рекомендації щодо подальшого розвитку HRM-платформи.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ І АНАЛІЗ ВЖЕ ІСНУЮЧИХ HRM СИСТЕМ

1.1 Аналіз існуючих інформаційних систем

1.1.1 Дослідження предметної області

HRM-системи (Human Resource Management Systems) виникли на початку 1990-х років як прості бази даних для обліку персоналу, але вже до 2000-х перетворилися на комплексні платформи, що об'єднували кадровий облік із бізнес-аналітикою та автоматизацією рутинних процесів [1]. Згідно з дослідженням Christopher S. (2018), сучасні HRM-системи інтегрують алгоритми штучного інтелекту та машинного навчання, аби прогнозувати потреби в навичках, оптимізувати набір співробітників і адаптувати програми розвитку під індивідуальні профілі [1]. Лук'янихін (2004) відзначав, що первинна мета автоматизації полягала у відмові від паперового документообігу: це скорочувало час обробки кадрових запитів на 30–40 % та зменшувало ймовірність помилок при введенні даних [2]. Однак сьогодні ключова цінність HRM-систем полягає не лише в оптимізації витрат часу, а передусім у стратегічній підтримці управління людським капіталом [3].

Історія та етапи розвитку

1. Перший етап (1990–2000): «революція баз даних» — впровадження модульних систем для обліку штатного розпису, відряджень, відпусток та лікарняних [2].

2. Другий етап (2000–2010): «розширена автоматизація» — поява аналітичних звітів, інтеграція з ERP і фінансовими модулями, перші хмарні рішення [4].

3. Третій етап (2010–дотепер): «інтелектуальні системи» — ML-алгоритми для аналізу продуктивності, платформи для дистанційного навчання, мобільні додатки та чат-боти для комунікації [3].

Ключові компоненти та ролі системи

HRM-система виконує три взаємопов'язані ролі:

- Операційна — автоматизація щоденних задач (облік часу, оформлення документів) і забезпечення законодавчого відповідності [5].
- Аналітична — збирання й обробка великих масивів HR-даних, побудова BI-дашбордів і прогнозних моделей для управлінських рішень [4].
- Стратегічна — підтримка планування кар'єрних шляхів, формування програм розвитку талантів та аналіз ефективності інвестицій у навчання [7].

Тенденції та виклики

- Діджиталізація процесів (Digital HRM): перехід до хмарних архітектур, інтеграція з мобільними платформами та чат-ботами для автоматизованого спілкування [9].
- Аналітика великих даних (HR-Analytics): використання алгоритмів для виявлення причин вибуття співробітників і прогнозування ключових показників [6].
- Гейміфікація та взаємодія: застосування елементів ігор для підвищення мотивації та залученості користувачів [11].
- Конфіденційність та безпека: з огляду на GDPR та локальні норми захисту даних, важливо забезпечити шифрування і контроль доступу на всіх рівнях системи [3].

HRM-система — це інтегрована IT-платформа, яка охоплює весь життєвий цикл співробітника: від рекрутингу до завершення кар'єри. Вона взаємодіє з іншими корпоративними системами (ERP, CRM, LMS) для створення єдиного інформаційного простору [10]. Основні переваги:

- Централізація та уніфікація даних: єдиний репозиторій для всіх HR-процесів, що забезпечує цілісність і достовірність інформації [2].
- Машинне навчання та AI: автоматичне розпізнавання шаблонів у поведінці співробітників, прогноз втрат кадрів та рекомендації щодо розвитку компетенцій [1].
- Підтримка мобільності: доступ до ключових модулів через мобільні додатки чи веб-інтерфейс, що забезпечує гнучкість та оперативність [3].

Типова HRM-система складається з наступних модулів [11]:

1. Рекрутинг - Автоматичне розміщення вакансій і пошук через Job-портали та соціальні мережі. А також AI-скринінг резюме, що знижує час на первинну обробку на 50 % [6].
2. Адаптація (onboarding) - Електронні check-list-и, навчальні курси з відстеженням прогресу, віртуальні тури по офісу.
3. Навчання й розвиток - Інтеграція з LMS, персоналізовані траєкторії навчання та сертифікація.
4. Оцінка продуктивності - Налаштування та відстеження KPI/OKR, 360°-фідбек, аналітика динаміки.
5. Кадровий облік та адміністрування - Управління договорами, дозвільними документами, розрахунок відпусток та лікарняних листків.
6. Аналітика й BI-звіти - Дашборди з показниками ефективності, предиктивна аналітика для планування навчання та ротації.

Завдяки поєднанню цих модулів компанія отримує єдину екосистему, де всі HR-операції синхронізовані, автоматизовані та прозорі, що сприяє підвищенню продуктивності й мотивації персоналу.

1.1.2 Дослідження інформаційних управляючих систем

SAP SuccessFactors є ядром екосистеми SAP, котра вже десятиліттями домінує серед глобальних ERP-рішень. Розгорнуте як multi-tenant SaaS із центрами обробки даних у кількох регіонах світу, SuccessFactors забезпечує безперебійний доступ і відповідність локальним законодавчим вимогам [12]. Платформа включають такі модулі як Основні кадрові ресурси (Core HR), Управління талантами (Talent Management), Виконання та Цілі (Performance & Goals), Послідовність та Розвиток (Succession & Development) та Аналітика робочою сили (Workforce Analytics), які інтегруються з SAP ERP через SAP Integration Suite. Впровадження успішно здійснене в компаніях Siemens та Coca-Cola де вдалося скоротити час на обробку кадрових процесів на 40-50 % та підвищити залученість співробітників за допомогою Career Marketplace з порадами на основі Machine Learning [12]. Шаблони SuccessFactors підтримують індивідуальне налаштування

робочих процес і вбудований SAP Analytics Cloud надає можливості генерувати детальні dashboard і прогнозувати показники кадрової мобільності.

Oracle HCM Cloud позиціонується як провідне корпоративне рішення зі здатністю адаптуватися до складних організаційних структур і великих обсягів даних. Завдяки хмарній архітектурі з мікросервісними компонентами та CI/CD-процесами через Oracle PaaS, оновлення відбуваються автоматично без простою системи. Основні функції охоплюють Global HR, Workforce Compensation, Talent Management та Workforce Predictions, доповнені HR Help Desk і цифровим помічником Oracle Digital Assistant. Впровадження в компаніях Johnson & Johnson та Barclays показало, що перехід на Oracle HCM дозволяє скоротити операційні витрати на HR до 30 % та підвищити точність прогнозів дефіциту навичок використовуючи Adaptive Intelligence Apps [13]. Ці аналітичні інструменти застосовують ML-моделі для виявлення ризиків втрати ключових співробітників і планування адаптивних програм розвитку.

Workday вирізняється інтуїтивним користувацьким інтерфейсом і швидкістю розгортання. Платформа є multi-tenant SaaS і отримує два оновлення на рік без зупинки сервісів, що забезпечує постійний доступ до найновіших функцій [13]. Workday HCM поєднує в собі модулі для управління персоналом, фінансового контролю, оплати праці та навчання з єдиною моделлю даних. Серед клієнтів платформи — Netflix та Airbnb, де впровадження Workday скоротило час налаштування кадрових схем на 60 % та дозволило автоматизувати процеси погодження запитів через мобільні додатки [13]. Workday Prism Analytics розширює можливості BI, а інтеграція з Microsoft Teams і Slack через готові конектори покращує комунікацію між HR та бізнес-підрозділами.

Незважаючи на те, що всі три рішення пропонують подібний базовий набір функцій для рекрутингу, адаптації, оцінки продуктивності та аналітики, вони відрізняються підходом до кастомізації, моделі інтеграції та загальної вартості впровадження. У випадку SAP SuccessFactors ключовим фактором є глибока інтеграція з ERP і аналітичний потенціал, Oracle HCM Cloud робить акцент на адаптивності та масштабованості, а Workday приваблює простотою використання

та швидкістю ROI. Тому вибір конкретного пакета залежатиме від розміру компанії, наявності власної ERP-інфраструктури, стратегічних цілей і бюджету на цифрову трансформацію. Для прикладу, можна розглянути порівняння основних характеристик у таблиці 1.1

Таблиця 1.1 Порівняльна таблиця основних характеристик

Характеристика	SAP SuccessFactors	Oracle HCM Cloud	Workday
Модель розгортання	SaaS	SaaS	SaaS
Інтеграція з ERP	Глибока (SAP ERP)	PaaS/SaaS API	Конектори
Аналітика	SAP Analytics Cloud	Oracle Analytics Cloud	Workday Prism Analytics
AI-функції	ML-рекомендації	Adaptive Apps	Аномалії та рекомендації
Мобільний клієнт	Так	Так	Так
Час впровадження	12–18 міс.	9–12 міс.	6–9 міс.
Модель ліцензування	Per-user	Per-user/Module	Per-user
Вартість початку (орієнт.)	Висока	Дуже висока	Середня

Джерело: сформовано автором на основі виконаного дослідження

Open-source HRM-системи активно набирають популярність у компаній малого та середнього бізнесу, котрі прагнуть гнучко адаптувати рішення під власні процеси без значних стартових витрат [15]. OrangeHRM, заснована у 2005 році [15], стала одним із першопрохідців на цьому ринку. Система побудована на фреймворку CakePHP [15] і використовує MySQL як основну СУБД [15], що забезпечує простоту інсталяції та мінімальні апаратні вимоги. Архітектура OrangeHRM спирається на модульність: ядро платформи відповідає за автентифікацію й управління сесіями, а додаткові пакети реалізують функціональні можливості — рекрутинг, облік відпусток і відвідуваності, оцінку продуктивності. Для корпоративних впроваджень передбачена інтеграція з LDAP та Active Directory

через стандартний SCIM-протокол [15], що дозволяє централізовано керувати обліковими записами співробітників.

Функціональні модулі OrangeHRM легко розширюються за допомогою плагінної системи, де кожен плагін містить набір PHP-класів та XML-конфігурацій. Доступні плагіни для інтеграції з OpenLDAP, корпоративними месенджерами (наприклад, Mattermost) та бухгалтерськими платформами (QuickBooks, Xero) [15], що автоматизують синхронізацію даних і сповіщення про кадрові події. Крім того, OrangeHRM пропонує готовий Docker-образ для швидкого тестування та розгортання в контейнеризованому середовищі [15], що спрощує CI/CD-процеси та забезпечує ізольовані тестові стенди.

Odoo HRM, що є частиною єдиного стеку Odoo ERP [16], виробляється на Python із фреймворком OpenObject та базою даних PostgreSQL. Ця система відзначається високим рівнем інтеграції з іншими модулями Odoo: CRM, MRP, фінанси та веб-сайт. Всередині архітектури Odoo кожен модуль реалізує бізнес-логіку через ORM-шар, який автоматично генерує REST-інтерфейс для CRUD-операцій над сутностями [16]. Для розробників доступні інструменти Odoo Studio та XMLView для конфігурування форм, дашбордів та автоматичних дій без необхідності правити вихідний код.

Odoo HRM підтримує розгортання в декілька кластерів PostgreSQL із використанням Patroni для забезпечення високої доступності та резервного копіювання [16]. Завдяки вбудованим можливостям роботи з заходами (Events) та планувальником Cron, модуль HR автоматично запускає завдання з розсилки нагадувань про закінчення випробувального терміну чи завершення сертифікацій [16]. Використання QWeb-шаблонів для звітності дає змогу створювати PDF-документи корпоративного рівня, а інтеграція з Odoo Slide та eLearning модулями забезпечує організацію дистанційного навчання.

Локальні ініціативи на базі OrangeHRM та Odoo HRM демонструють приклади адаптації системи під специфічні бізнес-кейси. Зокрема, кілька українських ІТ-компаній реалізували модуль контролю доступу, який використовує REST API OrangeHRM для синхронізації списку співробітників із

системою RFID-пропусків. Інший приклад — інтеграція Odoo HRM з SCORM LMS через Odoo Module Connector, що дозволило централізувати управління курсами та відстежувати результати навчання безпосередньо в HRM.

Незважаючи на всі переваги гнучкості та відсутності первинних витрат, open-source рішення вимагають від компанії наявності внутрішньої експертизи для конфігурування, адміністрування та забезпечення безпеки. Для мінімізації ризиків оновлень та втрати даних комерційні вендори пропонують Enterprise-версії зі SLA, регулярними оновленнями безпеки та професійною технічною підтримкою. Такий змішаний підхід дозволяє організаціям отримати контроль над процесами та водночас гарантувати стабільність роботи системи.

Комерційні HRM-платформи вирізняються глибинною функціональністю, готовими enterprise-модулями та гарантованою підтримкою сервісного рівня (SLA). Вони містять широкий набір аналітичних інструментів з AI та ML, інтегруються з ERP-системами і фінансовими платформами без додаткової розробки. Однак висока гнучкість і масштабованість до рівня десятків тисяч користувачів супроводжуються тривалими циклами впровадження (9–18 місяців) та значними ліцензійними витратами (200–400 \$ на користувача/рік).

Open-source HRM-рішення надають повну свободу кастомізації та відсутність ліцензійних платежів. Вони побудовані на популярних технологіях (PHP/MySQL у OrangeHRM і Python/PostgreSQL в Odoo HRM) та легко розгортаються в контейнерах. Основні витрати пов'язано з роботою IT-команди: налаштування модулів, інтеграція з корпоративними сервісами, забезпечення безпеки та резервування. Впровадження займає від 1 до 4 місяців, а загальний TCO у п'ятирічній перспективі може бути в 3–4 рази нижчим порівняно з комерційними рішеннями.

Для кращого розуміння, розглянутих програмних рішень можна звернути увагу на порівняння у таблиці 1.2

Таблиця 1.2. Аналітична таблиця порівняння програмних рішень

Критерій	Комерційні рішення	Open-source рішення
----------	--------------------	---------------------

Функціональна глибина	Широка (AI, BI, ERP-інтеграція)	Базова–середня (модулі HRM)
Час розгортання	9–18 місяців	1–4 місяці
Початкові ліцензійні витрати (\$/кор.)	200–400	0
Затрати на підтримку (\$/кор./рік)	20–30 % від ліцензії	50–100 (ІТ-ресурси)
TCO (5 років)	>400 000 \$	~100 000 \$
Масштабованість	>10 000 користувачів	до 1 000 користувачів
SLA	99.9 %	залежить від внутрішніх ресурсів
Гнучкість кастомізації	обмежена шаблонами вендора	повна, через вихідний код

Джерело: сформовано автором на основі виконаного дослідження

Комерційні HRM-продукти підходять організаціям із жорсткими вимогами до SLA та необхідністю глибокої аналітики без залучення власних технічних ресурсів. Вони ідеальні для великих компаній із розгалуженою ERP-інфраструктурою і ресурсами на довготривале впровадження.

Open-source системи є оптимальним вибором для малого й середнього бізнесу, який має внутрішню ІТ-команду та потребує швидкого старту з можливістю гнучкого розвитку продукту. Вони забезпечують значну економію витрат та контроль над кодом, але вимагають експертизи для підтримки безпеки та масштабування.

Формулювання чітких і детальних вимог стає фундаментом для розробки надійної, масштабованої та безпечної HRM-системи, що відповідає сучасним викликам бізнесу та законодавства. Вимоги поділяються на функціональні, нефункціональні та нормативні.

1.2 Обґрунтування вибору підходів і технологій для створення інформаційної управляючої системи

Функціональні вимоги

1. Управління рекрутингом: система має підтримувати завантаження резюме у форматах DOCX, PDF та TXT; автоматичний парсинг ключових полів (ім'я, контактна інформація, досвід); створення кастомізованих етапів відбору з призначенням відповідальних HR-менеджерів; планування інтерв'ю через інтеграцію з календарем (Google Calendar, Microsoft Outlook). Кандидатам має відправлятися автоматизоване підтвердження та нагадування про співбесіду через email і SMS.

2. Адаптація та онбординг: генерація індивідуальних чек-листів та навчальних планів на основі ролі співробітника; автоматичне призначення наставників і контроль виконання завдань через статуси "непочато", "в процесі", "завершено"; інтеграція з LMS для запуску онлайн-курсів і відстеження результатів.

3. Управління робочим часом і відпустками: ведення обліку робочого часу з точки входу/виходу; підтримка різних режимів робочого графіка (гнучкий, погодинний, змінний); автоматичний розрахунок кількості доступних днів відпусток, лікарняних та компенсацій; погодження заявок на відпустку з урахуванням навантаження команди та замінюваності.

4. Оцінка продуктивності: налаштування метрик KPI та OKR для співробітників і команд; підтримка багаторівневої оцінки (лідер — підлеглий, peer-to-peer, самооцінка); модулі для проведення 360°-фідбек із анонімними відгуками; формування графіків прогресу та звітів за встановленими періодами.

5. Навчання та розвиток: планування та реєстрація на внутрішні та зовнішні тренінги; календар навчальних подій з інтеграцією у загальний корпоративний календар; підтримка сертифікацій із автоматичним оновленням статусів; можливість створення індивідуальних траєкторій розвитку на підставі оцінок продуктивності.

6. Кадрове адміністрування: управління договорами, наказами та кадровими картками; версіонування документів із історією змін; автоматичне формування шаблонних документів (трудові договори, накази про відпустку) з опціональним підключенням цифрового підпису.

7. Аналітика та звітність: конструктор звітів із drag-and-drop інтерфейсом; інтеграція з BI-платформами (Tableau, Power BI) через REST API; налаштовувані дашборди з показниками обороту персоналу, середнього часу найму, рівня задоволеності; експорт звітів у PDF, XLSX.

Нефункціональні вимоги

1. Продуктивність: система повинна обробляти 1000 запитів за секунду, підтримуючи час відповіді на API-запити не більше 300 мс при навантаженні в 500 одночасних користувачів.

2. Надійність і доступність: рівень доступності 99,9 % щомісяця; впровадження кластеризації баз даних (PostgreSQL replication) та контейнеризації (Docker Swarm або Kubernetes) для безвідмовної роботи.

3. Масштабованість: горизонтальне масштабування сервісних компонентів; автоматичне балансування навантаження (NGINX, HAProxy); можливість підтримки до 10 000 активних користувачів.

4. Безпека: шифрування даних у спокої (AES-256) та під час передачі (TLS 1.3); аутентифікація OAuth2 та SAML 2.0; рольова модель доступу з правами на рівні сутностей та полів; аудиту журнал дій користувачів із можливістю відновити стан до будь-якого моменту.

5. Зручність використання та доступність: інтерфейс з адаптивною вёрсткою (Mobile-First), багатомовний (Ukr, Eng, Рус); відповідність стандарту WCAG 2.1 Level AA; інтуїтивні workflow з підказками та швидким доступом до найпопулярніших функцій.

6. Інтеграція: готові конектори для LDAP/Active Directory, корпоративних месенджерів (Slack, Microsoft Teams) та месенджерів (Telegram); webhook-и для сповіщень; можливість підключення до ERP/CRM систем через REST/SOAP API.

7. Конфігурованість: налаштування бізнес-процесів через BPM-движок із drag-and-drop редактором; параметризовані шаблони документів та звітів; система оновлень без простою.

Законодавчі та нормативні аспекти

1. Захист персональних даних: відповідність Закону України «Про захист персональних даних» та GDPR: механізми отримання та відкликання згоди; шифрування та анонімізація даних; права на доступ, виправлення та видалення.

2. Аудит і звітність: формування звітів для Державної податкової служби (ЄСВ, ПДФО) та Пенсійного фонду; зберігання кадрових документів не менше 75 років відповідно до законодавчих норм.

3. Електронний цифровий підпис: інтеграція з довіреними посередниками для накладання КЕП на кадрові документи; підтримка стандартів XAdES та PAdES.

4. Відповідність трудовому законодавству: автоматична генерація наказів та договорів згідно з КЗпП України; механізми контролю випробуваль

Комплексний аналіз потреб компаній при впровадженні HRM-системи охоплює не лише функціональні вимоги, але й оцінку організаційної культури, технологічної готовності та економічної доцільності. Такий аналіз спирається на принципи бізнес-аналізу, закладені в BABOK® Guide [17], стандарти ISO 30409 з опису HRM-процесів [28] та найкращі практики CIPD та SHRM [29][27]. Усі етапи роботи слід проводити з урахуванням багатовимірного підходу: соціального, процесного, технологічного та стратегічного.

Аналіз розпочинається з визначення цілей проєкту через включення ключових зацікавлених сторін (stakeholder analysis). Застосовують техніку PESTLE для розуміння макроекономічного та нормативного середовища [20], а також 5 Why-методу для виявлення кореневих причин проблем. Після узгодження цілей проводяться:

- Інтерв'ю та опитування: структуровані бесіди з HR-менеджерами, керівниками та співробітниками для визначення болючих точок, повторюваності

рутинних операцій та ступеня задоволеності наявними інструментами (Fowler, 2014) [18].

- Фокус-групи: панельні дискусії, спрямовані на уточнення сценаріїв використання (use cases) та збір неформальних зауважень щодо UX і процесів (Armstrong, 2010) [22].
- Процесне моделювання: застосування нотації BPMN 2.0 для картування as-is процесів та побудови to-be моделей із виділенням точок автоматизації (Davenport, 1993) [19].

Наступний крок — gap analysis, який порівнює існуючі процеси з ідеальними моделями. Дослідження Becker & Huselid (2001) [23] свідчить, що неправильно спроектовані процеси оцінки продуктивності можуть збільшувати адміністративне навантаження на 25 %. При аналізі виявляють:

- повторювані ручні операції (облік відпусток, складання графіків зйомок часу) — 30–40 % часу HR-фахівців;
- відсутність єдиної системи реєстрації результатів оцінювання та навчання;
- фрагментарне використання даних із різних систем (Excel, локальні бази), що призводить до розривів у аналітиці.

Аудит охоплює аналіз наявної інфраструктури, мережових потужностей, серверних ресурсів, політик безпеки та підходів DevOps [21]. За рекомендаціями Boxall & Purcell (2003) [26], важливо оцінити готовність до інтеграції через REST/SOAP API, підтримку SSO/LDAP та можливість контейнеризації (Docker, Kubernetes). Технічні сценарії включають:

- перевірку навантаження на бази даних (тестування до 500 з'єднань);
- оцінку latency між сервісами (<50 мс);
- audit-сканування на вразливості OWASP Top 10;
- підготовку резервних середовищ та disaster recovery планів.

Аналіз потреби у змінах культурного середовища ґрунтується на роботах Kotter (1996) [24] та Ulrich (1997) [22]. Визначають такі аспекти:

- ступінь залученості топ-менеджменту;
- готовність працівників до цифровізації процесів;
- рівень довіри до IT-рішень та історія попередніх впроваджень.

Застосовують опитування за шкалою ADKAR (Awareness, Desire, Knowledge, Ability, Reinforcement) для оцінки readiness.

Human Resource SWOT-аналіз дозволяє виділити внутрішні сильні (S) та слабкі (W) сторони, а також зовнішні можливості (O) та загрози (T) [20]. PESTLE-аналіз оцінює вплив політичних, економічних, соціальних, технологічних, правових і екологічних факторів [25]. Результати комбінують у матрицю ризиків та можливостей для проєкту.

Зіставлення важливості функцій і складності реалізації на основі методу MoSCoW (Must, Should, Could, Would) [23] та мапування benefit vs effort (Lawler, 2008) [25] дозволяє сформувану фазовану дорожню карту адаптації системи.

Розрахунок ROI базується на зниженні витрат на ручну роботу, скороченні часу заповнення вакантних позицій (Time-to-Fill) і зниженні плинності кадрів (Turnover Rate) [30]. Приблизні показники:

- зменшення адміністративних витрат на 20–30 %;
- скорочення Time-to-Fill на 15–25 %;
- підвищення задоволеності співробітників на 10–15 %.

Системний аналіз потреб компаній показує, що успіх впровадження HRM-системи залежить від збалансованого підходу до дослідження бізнес-процесів, технологічної та організаційної готовності, а також чіткого планування фаз впровадження із врахуванням економічних показників.

РОЗДІЛ 2. ХАРАКТЕРИСТИКА HRM СИСТЕМ

2.1 Структура і характеристика системи

Основною ціллю розроблюваної HRM-системи є значне підвищення ефективності та прозорості кадрових процесів шляхом створення єдиного інформаційного простору для всіх HR-функцій. Виконання цієї цілі передбачає реалізацію наступних ключових компонентів:

1. Централізація даних та автоматизація процесів. Впровадження єдиного репозиторію співробітників, що об'єднує інформацію про резюме, контракти, оцінювання та навчання, дозволить уникнути фрагментації даних і дублювання інформації [11]. Автоматизація перевірок, генерації документів та електронного підпису зменшить кількість ручних операцій, скоротивши час на адміністрування до 30–40 % [6].

2. Прозорість і контроль. Детальне логування усіх дій користувачів та зміни даних із зазначенням часу, ролі та контексту виконаної операції забезпечить відповідність вимогам аудиту (ISO 27001) та внутрішньої політики безпеки. Інтерактивні дашборди з KPI зведуть у єдине вікно показники обороту персоналу, тривалості рекрутингу й адаптації новачків, що сприятиме оперативному моніторингу та аналізу ефективності HR-функцій [5].

3. Оптимізація витрат і ресурсів. Система дозволить проводити розрахунки вартості найму, навчання та утримання персоналу в режимі реального часу, що створить передумови для точного планування бюджету HR-відділів. Використання аналітичних модулів із можливостями прогнозування (HR-Analytics) допоможе знижувати непередбачені витрати, пов'язані з плинністю кадрів, на 15–20 % [17].

4. Підвищення залученості та розвитку талантів. Інтеграція модулів адаптації й навчання з гейміфікацією та персоналізованими траєкторіями розвитку забезпечить індивідуальний підхід до кожного співробітника. Механізми зворотного зв'язку (360°-фідбек) та регулярні опитування задоволеності допоможуть формувати культуру зворотного зв'язку та безперервного покращення.

5. Гнучкість та масштабованість архітектури. Використання мікросервісного підходу, контейнеризації (Docker, Kubernetes) та REST-API гарантує можливість горизонтального масштабування під зростання штату до 10 000 користувачів і легку інтеграцію з ERP, CRM та сторонніми сервісами [21].

У результаті виконання основної цілі очікується досягнення скорочення операційних витрат на HR у середньому на 25–30 %, підвищення точності звітності та прогнозів, а також формування прозорих і адаптивних процесів управління персоналом, що відповідатимуть стратегічним цілям організації.

Автоматизація кадрового обліку є фундаментальним завданням системи, що передбачає створення єдиного сховища електронних карток співробітників з історією всіх змін і версій документів. Це дозволить відстежувати рух кадрів від моменту прийому на роботу до звільнення, зберігаючи інформацію про укладені трудові договори, накази, переведення між підрозділами та зміни посадових окладів відповідно до КЗпП України. Інтеграція з модулем «Платіжна відомість» надасть можливість безшовного обміну даними з бухгалтерією та автоматичного формування розрахункових відомостей.

Розбудова аналітичних звітів є другим ключовим завданням. Система повинна містити конструктор звітів із drag-and-drop інтерфейсом, що працює з гнучким набором KPI, таких як «Time-to-Fill», «Turnover Rate», «Cost-per-Hire» та «Time-to-Promotion». Ці показники мають завантажуватися в реальному часі через REST API до зовнішніх BI-платформ (Power BI, Tableau), що забезпечить HR-аналітикам можливість застосовувати додаткову візуалізацію, виконувати drill-down аналіз та створювати дашборди з динамічними фільтрами [11].

Третє завдання пов'язане з управлінням навчанням і розвитком персоналу. Розробка модуля передбачає підтримку LMS-протоколів SCORM та xAPI, що дозволить організувати онлайн-курси, відстежувати прогрес учасників та автоматично оновлювати статуси сертифікацій у профілях співробітників. Додатково слід реалізувати механізм рекомендацій навчальних програм на основі даних оцінки продуктивності та аналізу потреб у компетенціях, що змінюються відповідно до стратегічних цілей компанії.

Четвертий компонент — гнучкий облік робочого часу та відпусток. Система має підтримувати різні моделі робочого графіка: змінний, гнучкий та погодинний, а також автоматичний розрахунок залишків днів відпусток, лікарняних листів та компенсацій. Інтерактивний календар із можливістю колективного планування дозволить менеджерам координувати заявки на відпустки з урахуванням завантаженості команди та забезпечувати безперервність бізнес-процесів.

П'ятим завданням є впровадження комплексної системи оцінки продуктивності, що враховує методології KPI та OKR, а також механізми 360°-фідбек, peer-review та самооцінки. Зібрані дані повинні аналізуватися для формування динамічних звітів та рекомендацій для менеджменту щодо розвитку талантів і коригування індивідуальних планів розвитку [3].

Інтеграція з зовнішніми сервісами становить завершальний етап: передбачено підключення LDAP або Active Directory для єдиного входу (SSO), використання webhook-ів та push-сповіщень у корпоративних месенджерах (Slack, Microsoft Teams) для миттєвого інформування користувачів, а також синхронізацію з календарними системами для автоматизації планування інтерв'ю, онбордингу та навчальних заходів. Крім того, взаємодія з модулем «Управління активами» дозволить відслідковувати видачу та повернення обладнання співробітниками протягом усього періоду їхнього працевлаштування.

2.2 Архітектура системи

Проектування архітектури HRM-системи спирається на сучасні підходи до побудови розподілених додатків з високою надійністю та масштабованістю. У якості базового шаблону обрано архітектуру клієнт-сервер, розділену на три основні рівні: презентаційний, бізнес-логіки та зберігання даних. Презентаційний шар включає веб-додаток та мобільні клієнти, які забезпечують користувачам доступ до ключових функцій системи з будь-якого пристрою. Вже на цьому рівні реалізується принцип «Back-end for Front-end» (BFF), що дозволяє оптимізувати UX різних платформ — веб, iOS та Android — за допомогою окремих проксі-сервісів, які агрегають запити та адаптують відповіді.

У прикладному шарі бізнес-логіка розподілена між мікросервісами, кожен з яких виконує вузьке коло функцій: аутентифікація, управління користувачами, рекрутинг, онбординг, облік часу, оцінка продуктивності, навчання, звітність. Комунікація між сервісами відбувається через легковагові RESTful API та асинхронні повідомлення в брокері подій (Apache Kafka). Для забезпечення надійності та відмовостійкості застосовано шаблони Circuit Breaker та Bulkhead, а для централізованого конфігурування — окремий Config Server з Git-backed сховищем параметрів.

Шар зберігання даних базується на кластері PostgreSQL з використанням реплікації master-slave та автоматичного failover для гарантування безперервного доступу. В окремий NoSQL-бекенд (Redis) винесено кешування часто використовуваних даних і сесій, що знижує затримки доступу до інформації. Для забезпечення повного текстового пошуку та аналітичних запитів планується інтеграція з Elasticsearch, що дозволить швидко здійснювати фільтрацію та агрегацію інформації про співробітників.

Інфраструктурно система розгортається в контейнерному середовищі Kubernetes з використанням Helm-чартів для декларативного опису ресурсів. У ролі API Gateway застосовано Istio Service Mesh, що надає можливості маршрутизації, балансування навантаження, захищеного з'єднання mTLS між сервісами та збір телеметрії. Моніторинг і логування реалізовано за допомогою Prometheus, Grafana

та стека ELK (Elasticsearch, Logstash, Kibana), що забезпечує централізоване збирання метрик, логів та трасування запитів (Jaeger).

Архітектурна топологія розроблюваної HRM-системи спирається на трирівневу модель «клієнт-сервер», розділену на презентаційний, логічний і шар даних. Такий підхід відповідає загальноприйнятій системній ієрархії, де кожен рівень чітко окреслений та масштабований незалежно.

Презентаційний шар забезпечує інтерфейси для користувачів через веб- та мобільний клієнти. Веб-додаток реалізовано на сучасних фреймворках (React/Angular) з підходом Single Page Application (SPA), що забезпечує інтуїтивну навігацію та мінімальні затримки оновлення даних. Кожна сторінка завантажується через RESTful API або WebSocket для push-повідомлень, а захист реалізовано за допомогою Content Security Policy та HTTPS/TLS. Мобільні додатки (iOS/Android) використовують патерн Back-end for Front-end (BFF), тобто окремий проксі-сервіс для агрегування та оптимізації запитів відповідно до платформи, а також підтримують офлайн-режим з локальним кешем SQLite або Realm і синхронізацією через background tasks.

Прикладний шар складається з набору мікросервісів, кожен із яких відповідає за вузьке коло бізнес-логіки: аутентифікацію, управління користувачами, рекрутинг, облік часу, оцінку продуктивності, навчання та звітність. Ці сервіси розгортаються в контейнерному середовищі Kubernetes із використанням Helm-чартів для декларативного опису, а зв'язок між ними відбувається через API Gateway (Istio Service Mesh) з підтримкою маршрутизації, балансування навантаження, mTLS і телеметрії. Для обміну повідомленнями в реальному часі застосовано брокер Kafka, що гарантує надійний асинхронний обмін подіями між сервісами.

Шар зберігання даних базується на кластері PostgreSQL із реплікацією master-standby і автоматичним failover, що забезпечує високу доступність та цілісність даних. Для кешування сесій і часто запитуваних об'єктів використовується Redis, а аналітичні запити та повнотекстовий пошук реалізуються через Elasticsearch. Така комбінація дозволяє забезпечити швидку

відповідь API (≤ 200 мс) та одночасну обробку аналітичних навантажень без блокування транзакційного середовища.

Рівень	Компоненти	Відповідальність	Технології
Презентаційний шар	Веб-клієнт, мобільні SPA/нативні додатки	Візуалізація даних, взаємодія з користувачами, безпека	React, Angular, Swift, Kotlin, TLS, CSP
Прикладний шар	API Gateway, Auth, User, Recruitment, Time, Performance, Learning, Reporting	Обробка бізнес-логіки, маршрутизація, інтеграція	Kubernetes, Istio, Kafka,
Шар даних	PostgreSQL, Redis, Elasticsearch	Зберігання, кешування, пошук, реплікація	PostgreSQL HA, Redis, Elasticsearch

Джерело: сформовано автором на основі виконаного дослідження

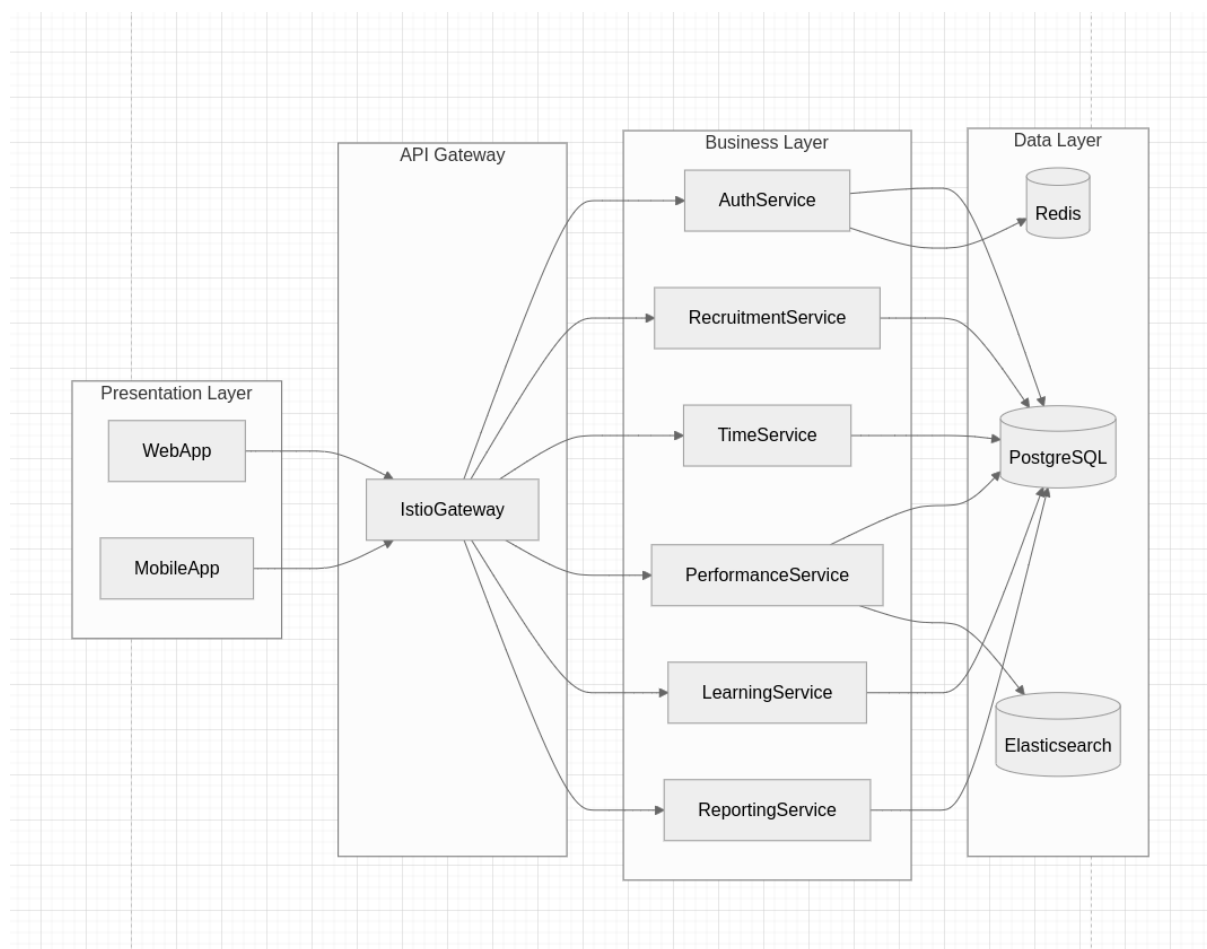


Рисунок 2.1 Компонента діаграма

Джерело: сформовано автором на основі виконаного дослідження

Таблиця опису компонентів 2.1 та компонентна діаграма 2.1, представлена вище, відображає ключові взаємозв'язки між шарами системи та окремими модулями, що базуються на трирівневій мікросервісній архітектурі з BFF-підходом для оптимізації UX на різних платформах [34]. Презентаційний шар, реалізований у вигляді веб-додатку та мобільних клієнтів, відповідає за взаємодію з користувачем і відображення бізнес-даних. Веб-клієнт і мобільний додаток направляють запити до API Gateway, який виступає єдиною точкою входу, забезпечуючи централізовану маршрутизацію, авторизацію за протоколом OAuth2/JWT [28] та моніторинг трафіку.

API Gateway (IstioGateway) виконує роль платформи сервіс-меш, що надає можливості маршрутизації, балансування навантаження, mTLS та збір телеметрії [36]. Він перенаправляє запити до відповідних мікросервісів у бізнес-шарі: AuthService обробляє аутентифікацію та авторизацію користувачів; RecruitmentService управляє процесами рекрутингу й адаптації новачків; TimeService займається обліком робочого часу та відпусток; PerformanceService відповідає за збір метрик продуктивності і ініціює обчислення KPI/OKR.

LearningService координує курси та сертифікації, інтегруючи протоколи SCORM та xAPI [29, 30], тоді як ReportingService акумулює дані з усіх компонентів і формує агреговані звіти. Комунікація між мікросервісами реалізована через синхронні HTTP-запити та асинхронний обмін подіями через Apache Kafka [35], що забезпечує стійкість і масштабованість системи.

Шар даних складається з трьох основних сховищ: PostgreSQL відповідає за транзакційні операції та зберігає критичні HR-дані з гарантією цілісності й доступності завдяки реплікації; Redis використовується для кешування сеансових даних і частих запитів, знижуючи затримки доступу [39]; Elasticsearch слугує для швидкого повнотекстового пошуку та аналітики, дозволяючи реалізувати складні запити фільтрації в режимі реального часу [38].

Таким чином, компоненти та їхні зв'язки забезпечують модульність і слабку зв'язаність, що відповідає принципам мікросервісної архітектури та полегшує

незалежне тестування, розгортання і масштабування кожного сервісу у кластері Kubernetes [31].

У серці HRM-системи лежать чотири ключові компоненти, кожен з яких реалізовано як окремий мікросервіс із власним життєвим циклом, API та механізмами масштабування. Така мікросервісна архітектура забезпечує розширюваність, стійкість до збоїв та незалежне розгортання оновлень.

Модуль рекрутингу відповідає за повний цикл від публікації вакансії до приймання кандидата на посаду. Він складається з підмодулів:

- Job Posting Service: генерує і розміщує вакансії на внутрішньому порталі і зовнішніх job-сайтах через REST API.
- Applicant Tracking Service: обробляє резюме, парсить дані (імена, контакти, досвід) за допомогою NLP-бібліотек, формує рейтинги кандидатів на основі multi-criteria scoring.
- Interview Scheduler: інтегрується з корпоративним календарем (Google Calendar, Outlook) і надсилає підтвердження та нагадування.

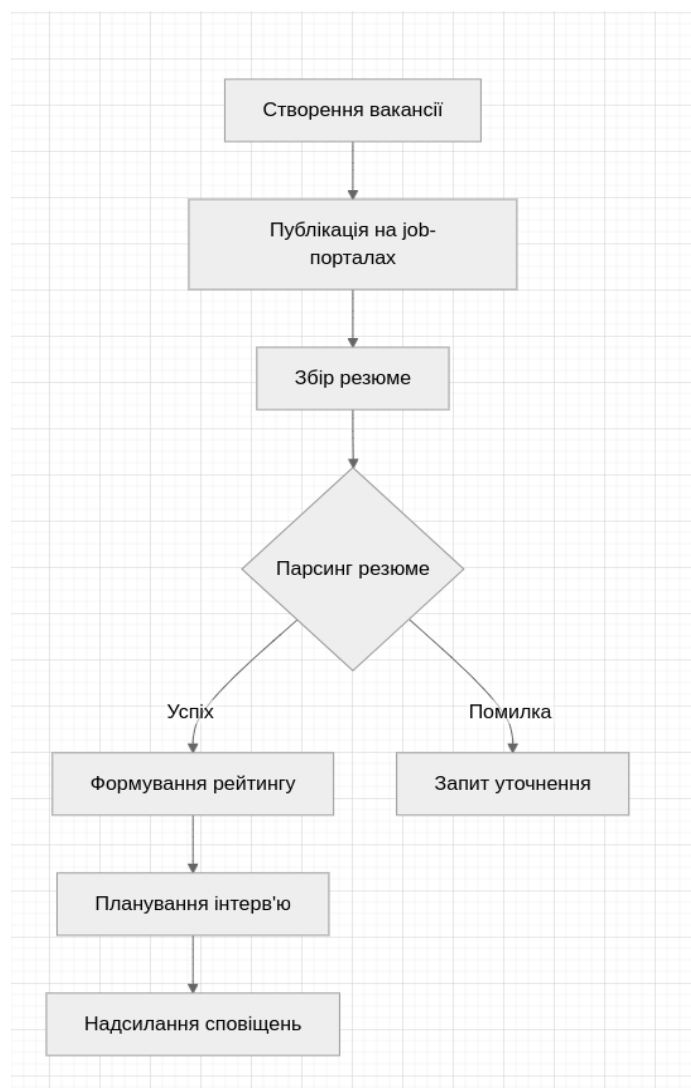


Рисунок 2.2 Діаграма модульного процесу

Джерело: сформовано автором на основі виконаного дослідження

Діаграма 2.2 відображає ключові етапи модульного процесу рекрутингу. На початку створюється вакансія (А), яка потім транслюється на зовнішні job-портали (В). Надходження резюме (С) ініціює автоматичний парсинг (D) з використанням NLP-алгоритмів; у разі успішної обробки формується рейтинг кандидатів (Е), а у випадку некоректних даних генерується запит на уточнення (F). Далі кандидати з найвищим рейтингом запрошуються на співбесіду, що планується через модуль Interview Scheduler (G), після чого система надсилає сповіщення учасникам процесу (H). Така візуалізація дозволяє чітко простежити логіку обробки кандидатів і місця для інтеграцій та кастомізації.

Модуль обліку робочого часу та відпусток відстежує присутність співробітників і управляє заявками:

- Time Logging: фіксація входу/виходу через веб-інтерфейс або RFID-термінали.
- Leave Management: автоматичний розрахунок залишків днів, обробка заявок з workflow-погодженням.

Надано діаграму зображену на рисунку 2.3

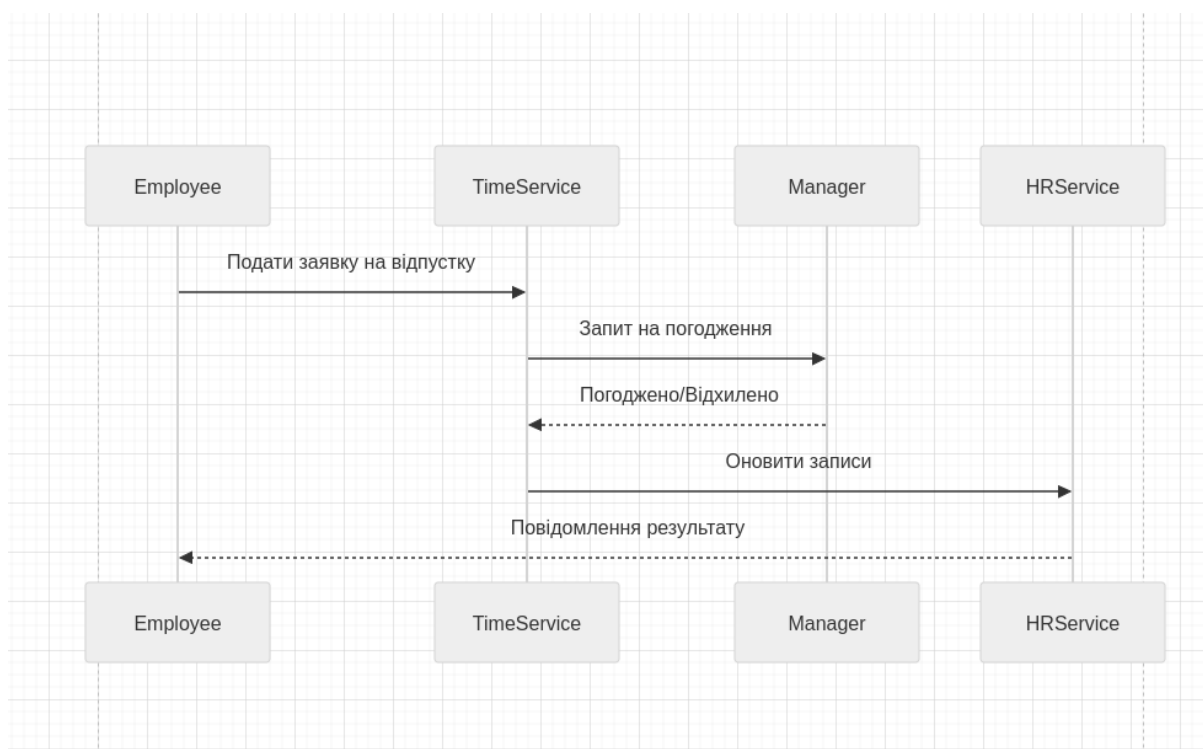


Рисунок 2.3 Діаграма бізнес процесу

Джерело: сформовано автором на основі виконаного дослідження

Діаграма демонструє класичний бізнес-процес обробки заявки на відпустку: працівник організації звертається до сервісу обліку часу з проханням оформити відпустку. Після отримання запиту сервіс обліку формує внутрішнє повідомлення й надсилає його безпосередньо керівникові для погодження.

Керівник приймає рішення і результат — чи відпустка схвалена, чи відхилена — надсилає назад у сервіс обліку часу. У разі позитивного рішення сервіс обліку передає команду HR-сервісу оновити відповідні записи в кадровій системі, щоб зміни збереглися в базі даних.

Після того, як HR-сервіс підтвердить внесення правок, сервіс обліку часу формує фінальне повідомлення для працівника з інформацією про статус заявки.

Завершальна частина діаграми показує, що всі чотири компоненти завершили свої дії і життєві лінії переходять у стан очікування нових запитів.

Модуль аналітики та звітності агрегує дані з усіх сервісів та надає можливості BI:

- Data Ingestion: збір метрик через Kafka Topics.
- Data Warehouse: зберігання нормалізованих даних у аналітичному сховищі.
- Report Builder: drag-and-drop інтерфейс для побудови кастомних дашбордів.

Модуль інтеграційних API забезпечує зв'язок з зовнішніми системами:

- SSO Service: LDAP/AD інтеграція та OAuth2/JWT авторизація.
- ERP/CRM Connectors: обмін фінансовими та клієнтськими даними.
- Notification Service: інтеграція з Slack, Teams, SMS шлюзами.

Таблиця 2.2. Таблиця відповідальностей та технологій

Модуль	Підмодулі	Технології	Інтеграції
Рекрутинг	Job Posting, Applicant Tracking, Interview Scheduler	Spring Boot, NLP, Scheduler	Job-портали, Calendar API, Email/SMS
Облік часу та відпусток	Time Logging, Leave Management	Node.js, RabbitMQ, RFID	RFID-системи, HR-адміністрування, Email
Аналітика та звітність	Data Ingestion, Data Warehouse, Report Builder	Kafka, PostgreSQL OLAP, React	Power BI, Tableau
Інтеграційні API	SSO Service, ERP Connector, Notification Service	OAuth2, SOAP/REST, Webhooks	LDAP, SAP, Slack, Teams, SMS, CRM

Джерело: сформовано автором на основі виконаного дослідження

Ця діаграма 2.1 та таблиця 2.2 демонструють внутрішню структуру модулів, їхні взаємозв'язки та технологічний стек, що підкреслює модульність, масштабованість та легкість підтримки системи.

2.2.1 Функціональні характеристики

У цьому розділі розглядаються деталізовані функціональні компоненти HRM-системи, які відповідають за забезпечення ключових кадрових операцій. Кожен модуль реалізовано як мікросервіс із чітко визначеним API та бізнес-логікою, що гарантує незалежне тестування та масштабування.

Модуль управління персоналом (PIM — Personnel Information Management) є центральним компонентом HRM-системи, відповідальним за збирання, зберігання та обробку всіх даних, пов'язаних із життєвим циклом співробітника. Архітектурно PIM реалізовано як мікросервіс із власним REST API, schema-first підходом та event sourcing для audit-логу [40].

Основні функції та підсервіси

1. Реєстрація співробітників:

- Імпорт даних з модулю рекрутингу через Kafka-топік `employee.hired`;
- Валідація даних за бізнес-правилами (перевірка унікальності ПІБ та email);
- Створення первинного профілю в базі даних та відправка webhook-повідомлень до BillingService і AccessControlService [41].

2. Управління профілями:

- CRUD-операції над атрибутами профілю (посада, підрозділ, зміна, менеджер);
- Динамічні custom fields із конфігурацією через Config Server;
- Підтримка локалізації та багатомовності полів профілю згідно з ISO 639-1 [42].

3. Історія змін (Audit Log):

- Збір подій (create, update, delete) та збереження у хранилищі подій (EventStoreDB) для CQRS-підходу [43];
- Можливість replay подій та відновлення стану профілю на будь-яку моментну точку часу через snapshot-версії;
- Інтеграція з SIEM-системою (Splunk, ELK) для аналізу безпеки.

Надано діаграми зображені на рисунках 2.4 та 2.5

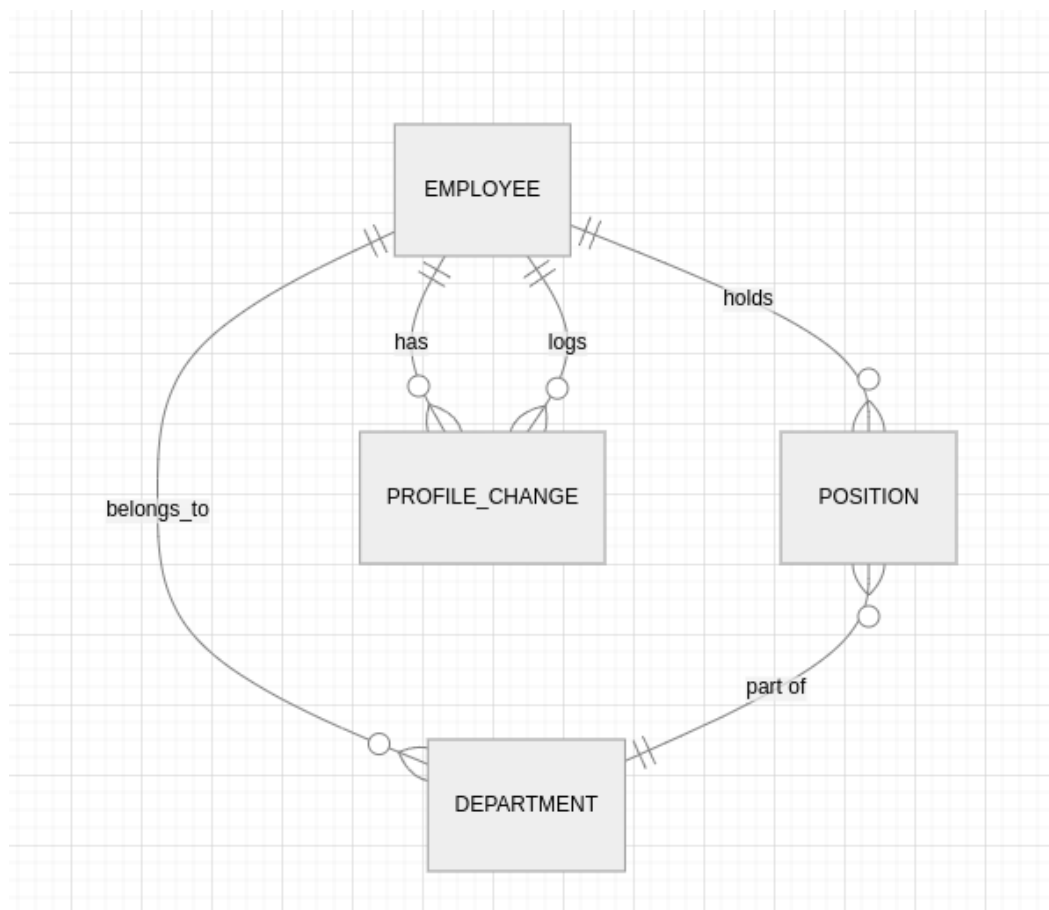


Рисунок 2.4 ER-діаграма сутностей

Джерело: сформовано автором на основі виконаного дослідження

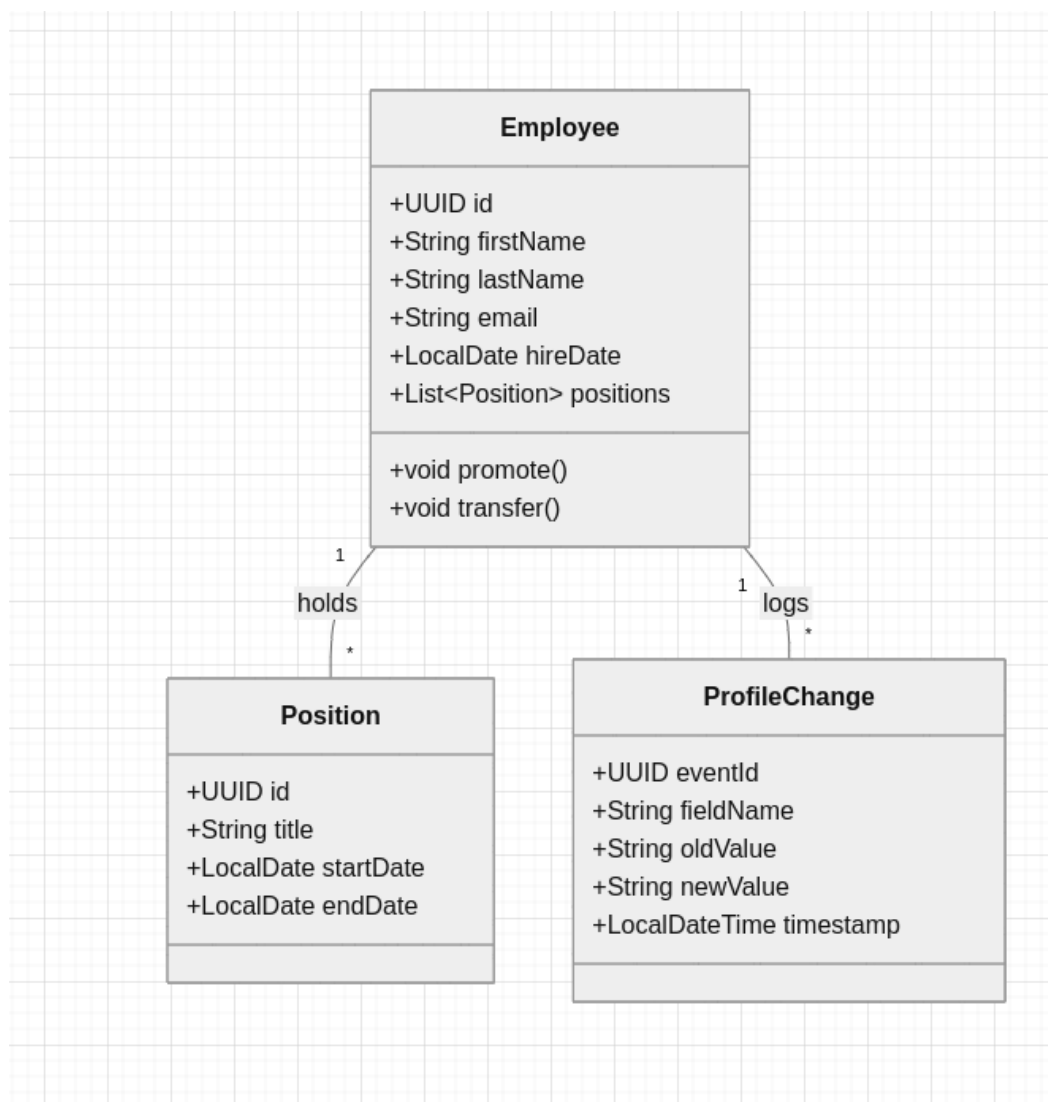


Рисунок 2.5 - Класова діаграма PIM-модуля

Джерело: сформовано автором на основі виконаного дослідження

Після успішної реєстрації негайні дані потрапляють до AuthService для створення облікового запису, а потім до AccessControlService для налаштування прав доступу. Уся комунікація між модулями відбувається через асинхронні події в Kafka та синхронні REST-виклики з коректним таймаутом та circuit breaker [44].

Модуль оцінки продуктивності та атестацій є критично важливим для стратегічного управління людським капіталом, оскільки він забезпечує автоматизований підхід до встановлення цілей, вимірювання результатів та підтримки безперервного розвитку співробітників. Цей сервіс реалізовано як незалежний мікросервіс із власним REST API, підтримкою асинхронних черг повідомлень та інтеграцією з BI-платформами для формування інтелектуальних дашбордів.

Модуль складається з чотирьох основних підсервісів:

- **KPI/OKR Setup Service:** відповідає за конфігурування метрик KPI та формування об'єктивів OKR на рівнях компанії, підрозділів і співробітників. Використовується JSON Schema для валідації та Git-backed Config Server для збереження версій шаблонів [45].
- **Feedback Engine:** забезпечує збори 360°-фідбек та peer-review, управління опитуваннями та анонімізацію відповідей. Інтегрується з Email API та корпоративними месенджерами через webhook-и [48].
- **Analytics Processor:** агрегує дані з KPI/OKR Setup та Feedback Engine, проводить нормалізацію й обчислення синтетичних показників, зберігає результати в ClickHouse для OLAP-запитів та передає дані в Grafana [47].
- **Assessment Scheduler:** відповідає за планування атестаційних кампаній, розсилання запрошень та обробку результатів. Використовує Apache Kafka для черг і Kubernetes CronJobs для періодичних завдань [35].

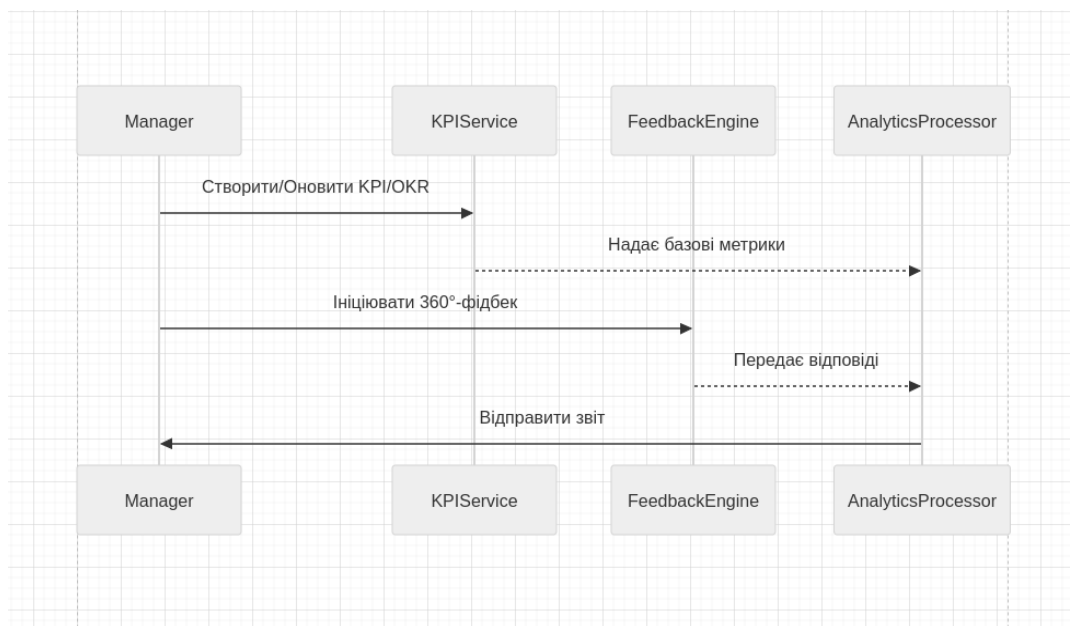


Рисунок 2.6 - Sequence Diagram: взаємодія з підсервісами

Джерело: сформовано автором на основі виконаного дослідження

Діаграма 2.6 ілюструє послідовність взаємодії між керівником, сервісом KPI, модулем збору зворотного зв'язку та аналітичним процесором у процесі встановлення та оцінки ключових показників ефективності й проведення 360°-фідбеку. Спочатку керівник надсилає до KPIService запит на створення або

оновлення KPI/OKR, після чого сервіс передає аналітичному процесору базові метрики для подальшої обробки.

Далі керівник ініціює 360°-фідбек через FeedbackEngine, який збирає відгуки учасників і в потрібний момент відправляє їх аналітичному процесору для поглибленого аналізу. Після обробки даних FeedbackEngine формує звіт і надсилає його назад керівнику, завершуючи цикл оцінки та забезпечуючи менеджера деталізованою інформацією про результати.

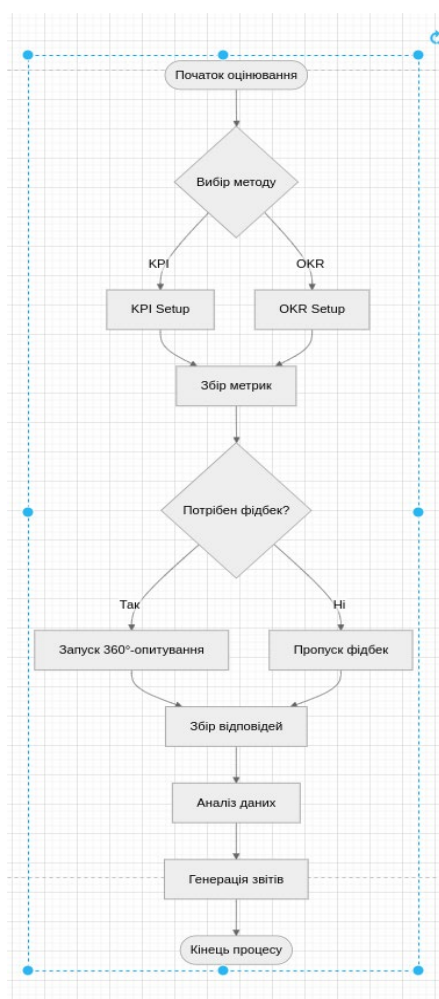


Рисунок 2.7 - Activity Diagram: процес оцінки продуктивності

Джерело: сформовано автором на основі виконаного дослідження

Діаграма 2.7 ілюструє послідовний процес оцінювання, який починається з ініціалізації процедури та вибору підходу — чи використовувати традиційні ключові показники ефективності (KPI), чи сучасну методологію OKR. Залежно від зробленого вибору відбувається відповідна підготовка: у випадку KPI проводиться налаштування цільових показників, а при OKR — визначаються об’єкти та ключові

результати. Після цього система переходить до етапу збору метрик, де фіксуються всі необхідні дані для подальшого аналізу.

Таблиця 2.3 Підсервісів та технологій

Підсервіс	Опис	Технології
KPI/OKR Setup Service	Управління шаблонами та ієрархічним зв'язуванням цілей	Java Spring Boot, JSON Schema, Git
Feedback Engine	360°-фідбек, peer-review, анонімність	Node.js, MongoDB, Email API, Slack SDK
Analytics Processor	Нормалізація, агрегування, ML-аналітика	Python, Pandas, TensorFlow, Grafana, ClickHouse
Assessment Scheduler	Планування атестацій, CRON, обробка черг	Go, Apache Kafka, Kubernetes CronJobs

Джерело: сформовано автором на основі виконаного дослідження

Таблиця 2.3 містить перелік чотирьох ключових підсервісів системи (KPI/OKR Setup-Service, Feedback-Engine, Analytics-Processor, Assessment-Scheduler), дає короткий опис їхньої функціональності та зазначає основні технології і фреймворки, які використовуються для реалізації кожного з них.

Модуль навчання та розвитку (Learning & Development) забезпечує планування, доставку та відстеження навчальних програм, спрямованих на підвищення компетенцій співробітників. Архітектурно реалізовано як мікросервіс із підтримкою LMS-протоколів (SCORM, xAPI) та інтеграцією з зовнішніми платформами e-learning [29, 30].

Структура підсервісів

- Course Catalog Service: керує реєстром курсів, тренінгів та сертифікацій з метаданими (тривалість, категорії, провайдери).
- Enrollment Service: обробляє заявки на навчання, здійснює перевірку попередніх кваліфікацій та доступу до контенту.
- Content Delivery Service: відповідає за трансляцію відео, інфографіки та інтерактивних занять, кешує ресурси через CDN та забезпечує offline access.

- Certification Service: автоматично генерує і видає сертифікати після проходження курсу, перевіряє успішність завершення за результатами тестів.

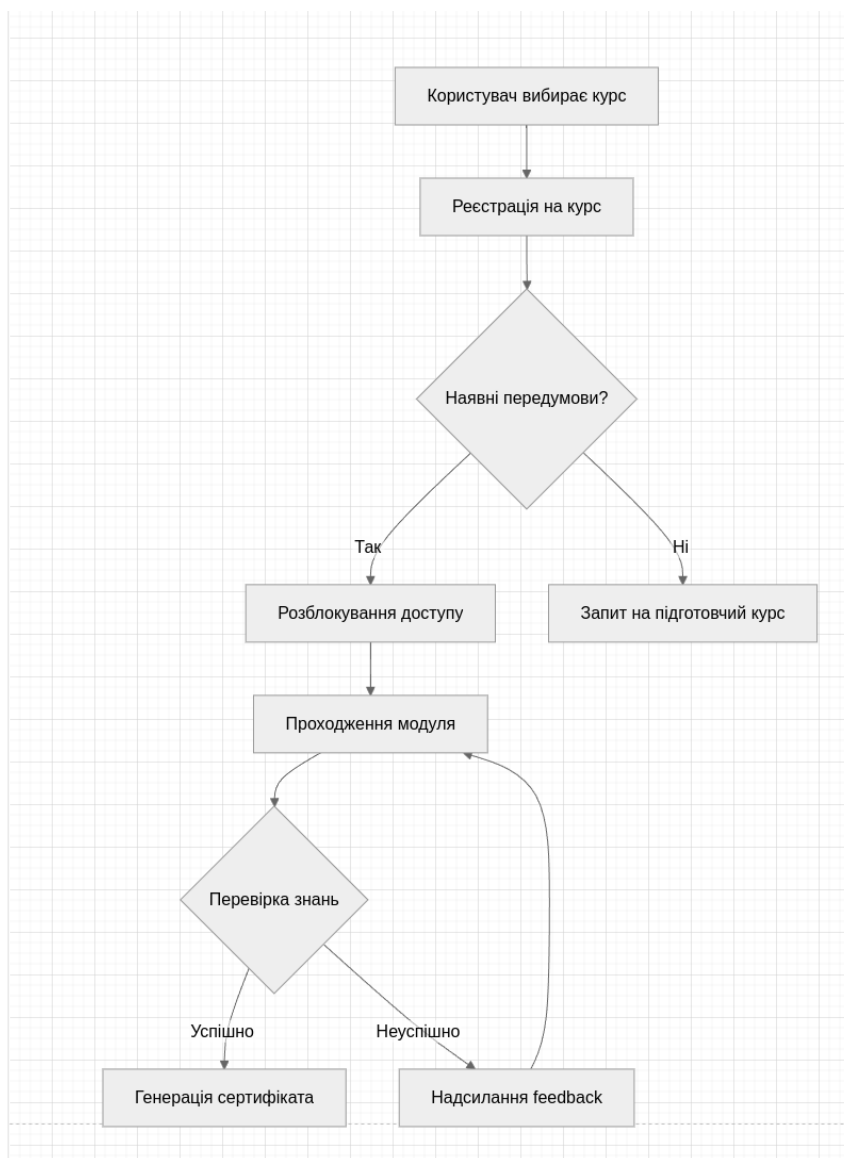


Рисунок 2.8 Activity діаграма «навчальний процес»

Джерело: сформовано автором на основі виконаного дослідження

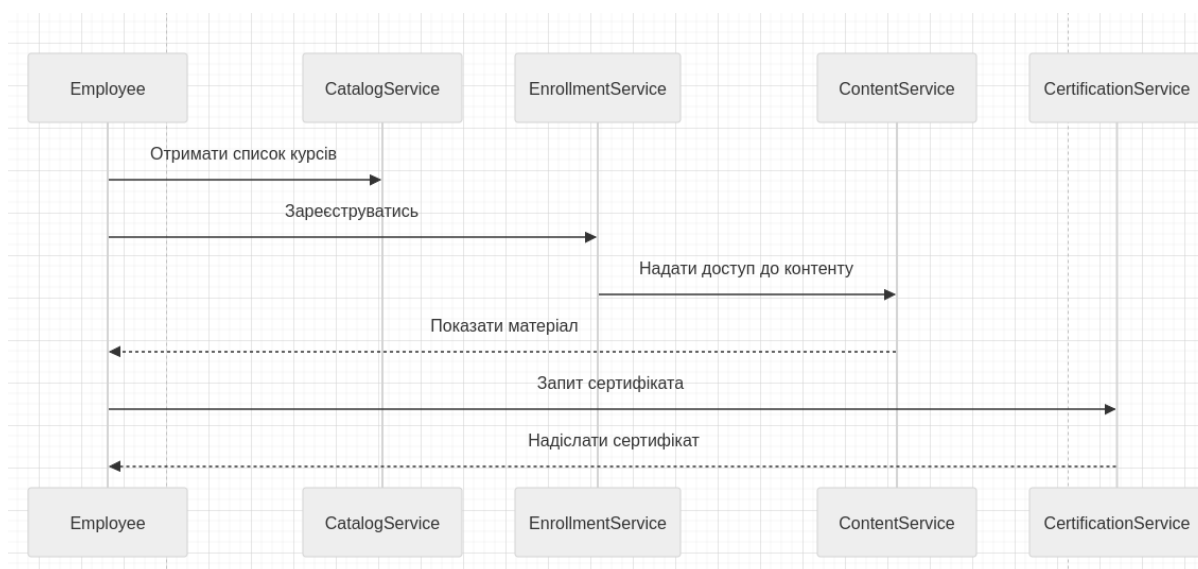


Рисунок 2.9 Sequence Diagram взаємодія компонентів

Джерело: сформовано автором на основі виконаного дослідження

Діаграми 2.8 та 2.9 ілюструють стандартний цикл взаємодії працівника з компонентами навчальної платформи: спочатку працівник (Employee) відправляє запит до сервісу каталогів (CatalogService) для отримання списку доступних курсів, після чого реєструється через EnrollmentService. Сервіс реєстрації обробляє заявку та передає команду ContentService надати доступ до навчальних матеріалів, які потім відображаються користувачеві. Завершальним етапом є ініціація запиту на сертифікат — працівник звертається до CertificationService, котрий генерує та надсилає йому готовий сертифікат.

Таблиця 2.4 Таблиця підсервісів та технологій

Підсервіс	Опис	Технології
Course Catalog Service	Управління реєстром курсів із метаданими	Java Spring Boot, PostgreSQL
Enrollment Service	Обробка заявок, валідація попередніх курсів	Node.js, MongoDB
Content Delivery	Трансляція і кешування навчальних матеріалів	Python, CDN, WebRTC
Certification Service	Генерація сертифікатів та видача	Go, PDFKit, QR-коди

Джерело: сформовано автором на основі виконаного дослідження

Таблиця 2.4 служить зведеним оглядом чотирьох ключових підсервісів системи, кожен із яких характеризується своїми функціями та технологіями реалізації: Course Catalog Service відповідає за управління реєстром курсів із

метаданими (Java Spring Boot, PostgreSQL), Enrollment Service — за обробку реєстрацій та валідацію попередніх курсів (Node.js, MongoDB), Content Delivery займається трансляцією і кешуванням навчальних матеріалів (Python, CDN, WebRTC), а Certification Service забезпечує генерацію й видачу сертифікатів із використанням Go, PDFKit та QR-кодових рішень.

Модуль звітності та аналітики є ключовим для моніторингу ефективності HR-процесів і прийняття обґрунтованих управлінських рішень. Архітектурно цей компонент реалізовано як окремий мікросервісний підпроект — ReportingService — із власним OLAP-сховищем, конектором до BI-інструментів та вбудованим конструктором дашбордів.

Архітектура та компоненти

- Data Ingestion Pipeline: збір подій і метрик із Kafka Topics (`employee.updates`, `time.logs`, `performance.metrics`, `learning.events`) та завантаження в Data Lake на базі Amazon S3 або Hadoop HDFS для довгострокового зберігання [55].
- ETL/ELT-процеси: перетворення сирих даних за допомогою Apache Spark та Airflow для наповнення Data Warehouse (наприклад, ClickHouse або Amazon Redshift) та OLAP-кубів [56].
- Dashboards Builder: веб-інтерфейс з drag-and-drop редактором, реалізований на React із використанням бібліотеки Recharts, який генерує інтерактивні візуалізації KPI: оборот кадрів, середній час найму, рівень залученості, ефективність навчання.
- BI-конектори: стандартизовані ODBC/JDBC драйвери для підключення зовнішніх BI-платформ (Power BI, Tableau, Qlik), а також REST-API для отримання JSON/XML-запитів.
- Alerts & Notifications: сервіс моніторингу ключових порогів KPI з можливістю генерації email- та Slack-сповіщень при досягненні критичних значень.

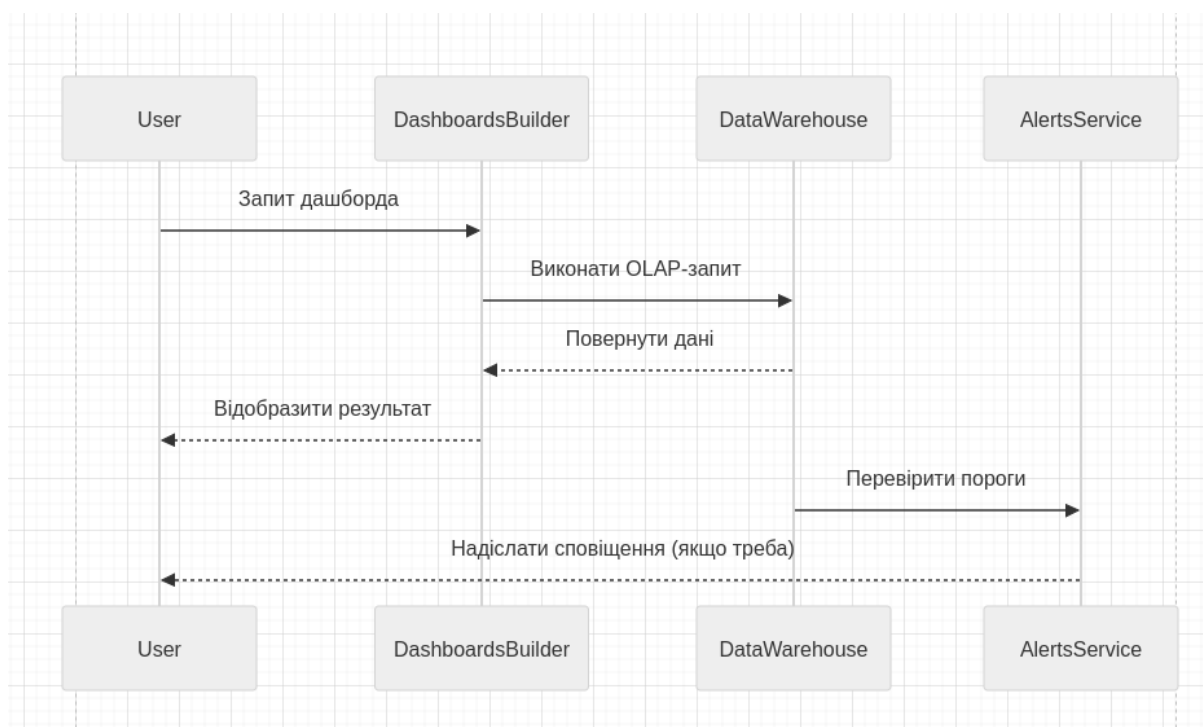


Рисунок 2.10 Sequence Diagram генерація звіту

Джерело: сформовано автором на основі виконаного дослідження

Діаграма 2.10 показує, як користувач взаємодіє з підсистемою побудови дашбордів і суміжними сервісами для генерації звіту. Спочатку користувач відправляє запит на формування дашборда до DashboardsBuilder, який у свою чергу виконує OLAP-запит у сховище даних (DataWarehouse). Після обробки запиту сховище повертає агреговані результати назад до DashboardsBuilder, а той відображає їх користувачеві. Одночасно DataWarehouse ініціює перевірку порогових значень через AlertsService, і в разі виявлення аномалій або досягнення певних порогів AlertsService надсилає сповіщення назад до DashboardsBuilder, який, за потреби, передає його користувачеві як додаткове повідомлення. Завершальні штрихові життєві лінії підкреслюють, що всі компоненти повертаються в очікування нових запитів. Опис використаних технологій виконано в таблиці 2.5

Таблиця 2.5 Опис технологій

Компонент	Технології	Опис
Data Lake	AWS S3 / HDFS	Зберігання сирих даних
ETL	Apache Spark, Airflow	Обробка та перетворення даних

Data Warehouse	ClickHouse / Redshift	OLAP-запити та зберігання агрегованих метрик
Dashboards Builder	React, Recharts	Інтерактивні графіки та візуалізації
BI-конектори	ODBC/JDBC, REST API	Підключення до Power BI, Tableau
Alerts Service	Node.js, Slack SDK, SMTP	Сповіщення за порогамі KPI

Джерело: сформовано автором на основі виконаного дослідження

2.2.2 Нефункціональні характеристики

Нефункціональні вимоги визначають критичні якісні атрибути HRM-системи, що формують її поведінку в експлуатації. У цьому підрозділі розглядаються основні аспекти — безпека, продуктивність і масштабованість, зручність використання та доступність, а також інтеграція з зовнішніми системами.

Для захисту конфіденційних HR-даних система реалізує багаторівневий підхід до безпеки. Аутентифікація здійснюється через OAuth2 з використанням JWT-токенів, які підписуються та перевіряються за допомогою асиметричних ключів [28]. Всі сервіси в мережі взаємодіють через mTLS (TLS 1.3) для гарантування цілісності та конфіденційності переданих повідомлень. Зберігання секретів і ключів реалізовано через HashiCorp Vault з ротацією ключів за розкладом. Рольова модель доступу (RBAC) впроваджена на рівні API Gateway та окремих мікросервісів, з правами CRUD на сутності та поля згідно з принципом найменших привілеїв. Регулярне сканування вразливостей відбувається за стандартом OWASP Top 10, а події безпеки передаються у SIEM-систему (Splunk) для кореляції й оповіщення.

HRM-система повинна підтримувати обробку не менше ніж 1 000 REST-запитів за секунду з середнім часом відповіді $API \leq 300$ мс при 95 % навантаженні [5]. Горизонтальне масштабування здійснюється Kubernetes Horizontal Pod Autoscaler, який реагує на метрики CPU та latency, забезпечуючи підтримку до 5 000 одночасних користувачів [31][32]. Для аналітичних OLAP-запитів у ClickHouse обмежено час виконання складних агрегацій до 2 с при обсягах до 10 млн записів

[56]. Кешування на рівні API реалізовано через Redis із TTL для зменшення навантаження на бази даних.

Інтерфейси системи розроблені з урахуванням принципів адаптивного дизайну та доступності WCAG 2.1 Level AA. Веб- та мобільні клієнти підтримують мультимовність (i18n) згідно з ISO 639-1, що дозволяє динамічно перемикаати локалізацію не менше ніж трьома мовами (Ukr, Eng, Rus). Навігація підтримує клавіатурний доступ, масштабовані шрифти та альтернативні описи для компонентів UI. Час завантаження стартового екрану не перевищує 1 с на пристроях із з'єднанням 5 Mbps [3].

Система передбачає широку інтеграцію з корпоративним ландшафтом:

- LDAP/Active Directory забезпечує єдиний вхід (SSO) та синхронізацію користувацьких груп [41].
- ERP/CRM: конектори SOAP/REST для обміну фінансовими даними та актуалізації записів про заробітну плату (SAP, Odoo) [42].
- Системи електронного документообігу (DocuSign, «Електронний підпис») через API для підписання кадрових документів.
- Webhook та ESB-шлюзи за паттернами Enterprise Integration Patterns (EIP) для обміну подіями з іншими бізнес-процесами [43].

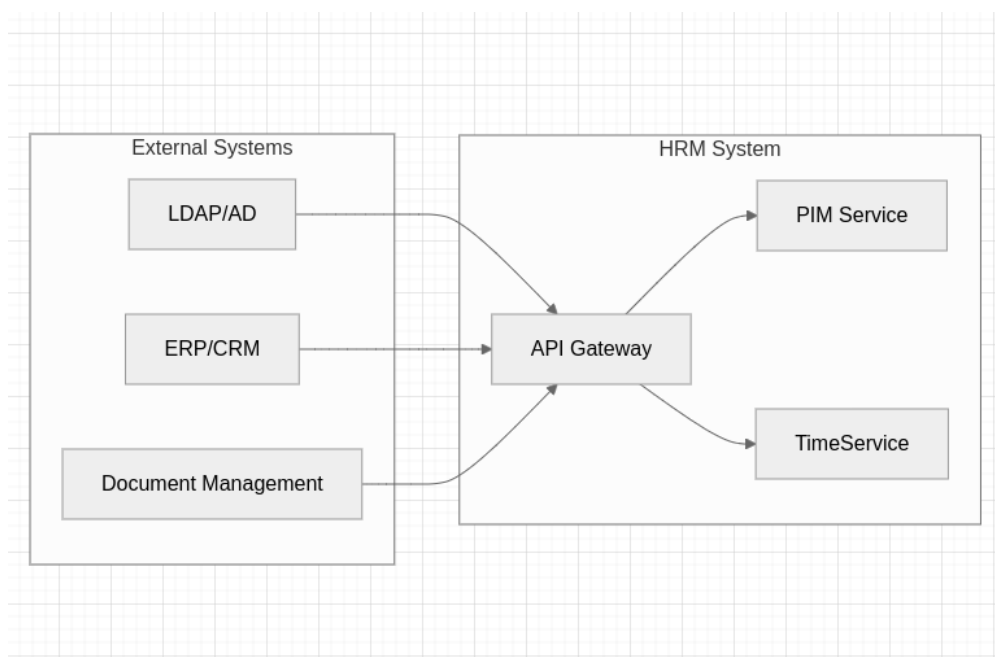


Рисунок 2.11 Інтеграційна діаграма

Джерело: сформовано автором на основі виконаного дослідження

На Рисунку 2.11 зображено інтеграційну архітектуру системи управління людськими ресурсами (HRM System) із зовнішніми системами. Діаграма ілюструє, як різні компоненти взаємодіють між собою для забезпечення обміну даними та функціональності.

2.2.3 Технологічний стек

Серверна частина HRM-системи реалізована на сучасних та перевірених технологіях, що гарантують високий рівень продуктивності, стійкості та зручності обслуговування. Основою бекенду є мікросервісна архітектура із розмежуванням відповідальності між сервісами. Ключові рішення:

Мови програмування та фреймворки. Головний стек обробки транзакційних запитів побудований на Java 17 (OpenJDK) із використанням Spring Boot 2.x, що забезпечує легку налаштування REST API, автоматичну конфігурацію та інтеграцію з Spring Security для OAuth2/JWT-автентифікації [42]. Для реалізації реактивного програмування та WebFlux-підходу застосовується Spring WebFlux, що дозволяє обробляти велику кількість одночасних з'єднань з малим споживанням ресурсів.

Для аналітичних мікросервісів та обробки даних використовується Python 3.9 у поєднанні з FastAPI (Uvicorn + Starlette), що підтримує асинхронний I/O та високу швидкість розробки. Для фонових завдань та ETL-процесів застосовують Celery з брокером RabbitMQ та бібліотеки Pandas/NumPy для обробки табличних даних.

Бази даних та сховища.

- PostgreSQL 14 із master-standby реплікацією та автоматичним failover у кількох зонах доступності гарантує цілісність транзакцій та швидкий відкат [56].
- Redis 6 використовується як ін-меморі кеш для сесій, частих результатів запитів та блокувань (distributed locks), з TTL-настройками для автоматичного очищення.
- ClickHouse 21.x слугує аналітичним OLAP-сховищем, що обробляє великі об'єми даних з низькою затримкою (до 2 с для 10 млн записів) [56].

- Elasticsearch 7.x забезпечує повнотекстовий пошук та агрегацію за параметрами співробітників і вакансій [9].

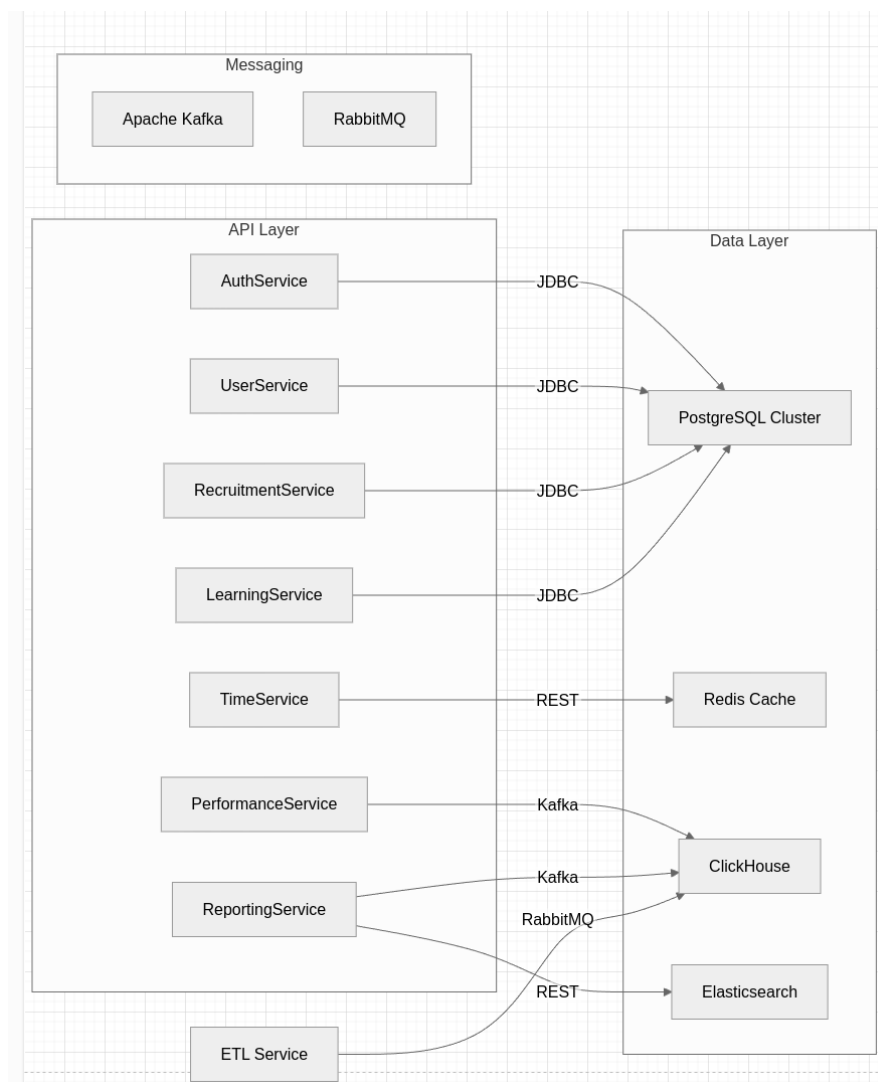


Рисунок 2.12 Компонентна діаграма серверної частини

Джерело: сформовано автором на основі виконаного дослідження

На Рисунку 2.12 представлена деталізована компонентна діаграма серверної частини HRM-системи, що ілюструє внутрішню структуру бекенду, реалізованого на основі мікросервісної архітектури, як було зазначено раніше. Діаграма розділяє компоненти на логічні шари для кращого розуміння їхніх ролей та взаємозв'язків.

Основними шарами архітектури є:

1. Messaging (Шар обміну повідомленнями): Цей шар відповідає за асинхронну взаємодію між сервісами та обробку подій. Він включає:

- Apache Kafka: Використовується як високопродуктивний розподілений брокер повідомлень для потокової обробки даних та забезпечення надійної

комунікації між мікросервісами, особливо для сценаріїв, що вимагають високої пропускної здатності та стійкості до збоїв. Згідно з попереднім текстом, він може задіюватися аналітичними мікросервісами та для ETL-процесів.

- RabbitMQ: Інший брокер повідомлень, який, як зазначено в тексті, використовується Celery для фонових завдань та ETL-процесів. Він може бути більш придатним для сценаріїв, де важлива надійна доставка окремих повідомлень та складніші патерни маршрутизації.

2. API Layer (Шар API): Цей шар містить набір мікросервісів, які реалізують основну бізнес-логіку HRM-системи та надають функціональність через API. Відповідно до мікросервісної архітектури, кожен сервіс має розмежовану відповідальність:

- AuthService: Відповідає за автентифікацію та авторизацію користувачів, ймовірно, використовуючи підходи на основі OAuth2/JWT, як вказано в попередньому тексті з прив'язкою до Spring Security.

- UserService: Керує даними користувачів та їхніми профілями.

- RecruitmentService: Реалізує функціональність, пов'язану з процесами найму, вакансіями, кандидатами.

- LearningService: Забезпечує функціональність для управління навчанням та розвитком співробітників.

- TimeService: Відповідає за облік робочого часу, відпусток, відгулів тощо, що відповідає попередній інтеграційній діаграмі.

- PerformanceService: Керує процесами оцінки ефективності співробітників.

- ReportingService: Генерує різноманітні звіти на основі даних із різних джерел.

- ETL Service: Сервіс, відповідальний за процеси вилучення, перетворення та завантаження даних (ETL), зокрема для аналітичних потреб, що узгоджується з використанням Python, Celery та RabbitMQ для цих цілей.

3. Data Layer (Шар даних): Цей шар включає різні сховища даних, оптимізовані для різних типів інформації та завдань:

- PostgreSQL Cluster: Основна реляційна база даних, що використовується для зберігання транзакційних даних з високою цілісністю та підтримкою складних запитів. Наявність master-standby реплікації та автоматичного failover, як зазначено в тексті, гарантує високу доступність та стійкість.
- Redis Cache: Ін-меморі сховище даних, яке використовується як кеш для швидкого доступу до часто запитуваних даних, сесій користувачів та розподілених блокувань. Це забезпечує високу продуктивність системи за рахунок зменшення навантаження на основну базу даних.
- ClickHouse: Аналітичне OLAP-сховище, спеціалізоване для швидкої обробки великих обсягів даних та виконання аналітичних запитів з низькою затримкою. Це відповідає згадці про його використання для аналітичних мікросервісів.
- Elasticsearch: Повнотекстовий пошуковий рушій, що використовується для реалізації функціональності пошуку по тексту та агрегації даних, зокрема для пошуку співробітників або вакансій.

Взаємозв'язки та потоки даних:

Діаграма показує, як мікросервіси (API Layer) взаємодіють з шаром даних. Більшість сервісів (AuthService, UserService, RecruitmentService, LearningService) підключаються до PostgreSQL Cluster за допомогою протоколу JDBC, що є стандартним для взаємодії з реляційними базами даних. TimeService та ReportingService взаємодіють з Redis Cache та Elasticsearch, ймовірно, через REST API, оскільки ці сховища часто надають такий інтерфейс. ReportingService також взаємодіє з ClickHouse, що логічно для аналітичних звітів.

Взаємодія між сервісами та шаром обміну повідомленнями відбувається через Kafka та RabbitMQ. ETL Service показано як джерело даних для ClickHouse та Elasticsearch, що підкреслює його роль у підготовці даних для аналітики та пошуку.

Таблиця 2.6 Технології серверної частини

Компонент	Технологія	Версія	Призначення
HTTP API	Spring Boot	2.x	Побудова REST/Reactive сервісів
Реактивне програмування	Spring WebFlux	2.x	Неблокуючий I/O для високої пропускної здатності
Аналітичний бекенд	FastAPI	0.70	Asynchronous REST, ETL
Сесійне кешування	Redis	6.x	Швидкий кеш, блокування
OLAP-сховище	ClickHouse	21.x	Обробка великих обсягів даних
Транзакційне сховище	PostgreSQL	14	ACID, реплікація, failover
Пошуковий движок	Elasticsearch	7.x	Повнотекстовий пошук, агрегація
Меседжинг	Apache Kafka	2.x	Асинхронний обмін подіями
Фонові завдання	RabbitMQ + Celery	3.x	Обробка ETL та завдань за розкладом

Джерело: сформовано автором на основі виконаного дослідження

Клієнтська частина HRM-системи складається з двох основних каналів доступу: веб-інтерфейсу для десктопних користувачів та мобільного додатку для співробітників на ходу.

Фреймворк: React 18 з TypeScript. Використовується для побудови компонентів у стилі SPA (Single Page Application), що забезпечують швидку навігацію без перезавантаження сторінки.

UI-бібліотека: Ant Design 5, що надає готові компоненти (таблиці, форми, дашборди) із підтримкою темізації.

Управління станом: Redux Toolkit із RTK Query для оптимізованих запитів до API та створення кешу даних [60].

Мультимовність: i18next з динамічною підвантаженням перекладів згідно з ISO 639-1 [42].

Безпека: захищені HTTP-only куки для JWT, Content Security Policy та Subresource Integrity.

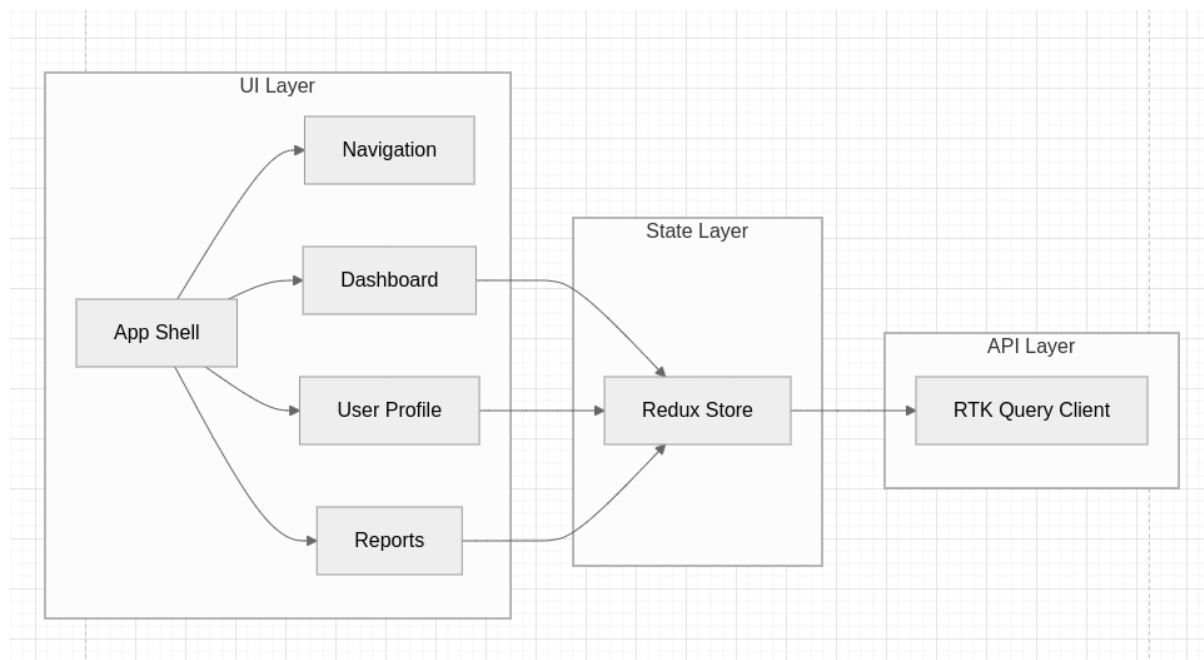


Рисунок 2.13 Компонентна діаграма веб-клієнта

Джерело: сформовано автором на основі виконаного дослідження

На Рисунку 2.13 представлена компонентна діаграма веб-клієнта HRM-системи, яка візуалізує внутрішню архітектуру клієнтської частини, реалізованої як односторінковий додаток (SPA) з використанням фреймворку React 18 та мови TypeScript, як було зазначено в попередньому описі. Діаграма розділена на логічні шари, що відображають організацію коду та потоки даних.

Основними шарами архітектури веб-клієнта є:

1. UI Layer (Шар користувацького інтерфейсу): Цей шар відповідає за візуальне представлення даних та взаємодію з користувачем. Він включає:
 - App Shell: Базова оболонка додатку, яка містить спільні елементи інтерфейсу, такі як хедер, футер, бічна навігація та область для відображення основного контенту.
 - Navigation: Компонент або набір компонентів, що реалізують навігацію по додатку, дозволяючи користувачеві переходити між різними розділами (наприклад, Дашборд, Профіль користувача, Звіти).
 - Dashboard: Компонент або сторінка, що надає загальний огляд ключових показників та швидкий доступ до основних функцій.

- User Profile: Компонент або сторінка для перегляду та редагування інформації про поточного користувача.
- Reports: Компонент або розділ для доступу та відображення звітів, які генеруються серверною частиною.
- Для побудови цих компонентів, як вказано в попередньому тексті, використовується UI-бібліотека Ant Design 5, яка надає готові та стилізовані елементи інтерфейсу (таблиці, форми тощо), прискорюючи розробку та забезпечуючи узгоджений зовнішній вигляд.

2. State Layer (Шар управління станом): Цей шар відповідає за централізоване зберігання та управління станом додатка, тобто даними, які використовуються різними компонентами. Він представлений:

- Redux Store: Єдине сховище стану додатку. Використання Redux Toolkit, згаданого в попередньому тексті, спрощує роботу з Redux, надаючи зручні інструменти для створення редюсерів, екшенів та конфігурації стору. Шар стану є центральним вузлом, з яким взаємодіють компоненти UI для отримання необхідних даних та оновлення стану.

3. API Layer (Шар API): Цей шар відповідає за взаємодію веб-клієнта із серверною частиною (бекендом) через API. Він містить:

- RTK Query Client: Компонент або модуль, що є частиною Redux Toolkit (RTK Query). Він спеціально розроблений для оптимізації запитів до API, кешування даних, автоматичного оновлення даних при змінах та управління станом завантаження/помилки. RTK Query Client отримує дані від серверного API (ймовірно, через API Gateway, як показано на інтеграційній діаграмі) та надає їх до Redux Store для подальшого використання компонентами UI.

Взаємозв'язки та потоки даних:

Діаграма ілюструє наступні ключові взаємозв'язки:

- App Shell містить та організовує різні UI компоненти (Navigation, Dashboard, User Profile, Reports).

- Компоненти UI (Navigation, Dashboard, User Profile, Reports) взаємодіють з Redux Store для отримання даних, необхідних для відображення, та для відправлення дій, що змінюють стан додатка (наприклад, введення даних у форму, натискання кнопки).

- Redux Store використовує RTK Query Client для виконання асинхронних запитів до серверного API (наприклад, отримати дані користувача, завантажити список звітів). Результати запитів від RTK Query Client зберігаються в Redux Store, звідки їх можуть отримати UI компоненти.

Така архітектура, заснована на розподіленні відповідальності між шарами (UI, State, API), використанні React для побудови компонентів, Redux Toolkit для управління станом та RTK Query для взаємодії з API, забезпечує модульність, зручність розробки, тестування та підтримки клієнтської частини, а також оптимізує роботу з даними, що надходять із сервера.

Мобільний додаток складається з таких компонентів:

- Технологія: Flutter 3.3, що дозволяє створювати нативні iOS та Android додатки з єдиної кодової бази [61].
- Стейт-менеджмент: Riverpod 2.0 для розподілу залежностей та реактивних оновлень UI.
- Навігація: GoRouter для декларативного опису маршрутів.
- Локальне сховище: Hive для збереження налаштувань та кешу в офлайн-режимі.
- Інтеграція: взаємодія з камерою, push-сповіщення через Firebase Cloud Messaging.
- Безпека: зберігання токенів у Flutter Secure Storage, SSL Pinning.

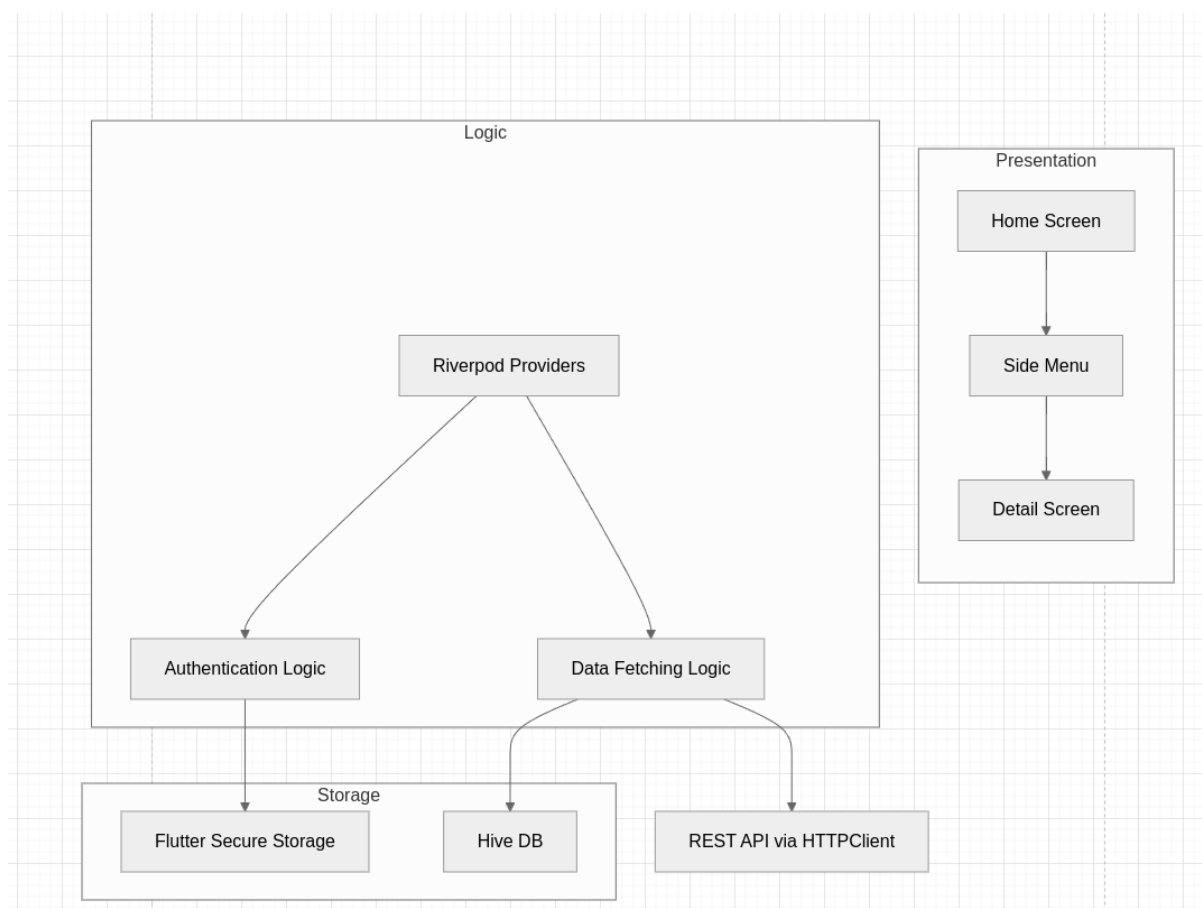


Рисунок 2.14 Архітектурна діаграма мобільного додатку

Джерело: сформовано автором на основі виконаного дослідження

На Рисунку 2.14 зображено архітектурну діаграму мобільного додатку HRM-системи, розробленого за допомогою Flutter 3.3, що забезпечує кросплатформну розробку для iOS та Android з єдиної кодової бази. Діаграма представляє логічну структуру додатку, розподілену на три ключові шари: Presentation, Logic та Storage.

1. Presentation (Шар представлення): Цей шар відповідає за інтерфейс користувача та відображення інформації. Він складається з екранів додатку, з якими безпосередньо взаємодіє користувач:

- Home Screen: Головний екран додатку, який може містити основну інформацію або відправні точки для навігації.
- Side Menu: Бічне меню навігації, що забезпечує доступ до різних розділів додатку. Використання GoRouter, згаданого в тексті, ймовірно, реалізує декларативний підхід до управління маршрутами між цими екранами.

- **Detail Screen:** Екрани, що відображають детальну інформацію про певний об'єкт (наприклад, профіль співробітника, деталі завдання). Цей шар відповідає за візуалізацію даних, отриманих із шару логіки, та обробку дій користувача.

2. **Logic (Шар логіки):** Цей шар містить бізнес-логіку додатку та управляє станом. Він є "мозком" мобільного додатку:

- **Riverpod Providers:** Використання Riverpod 2.0, як зазначено в тексті, централізує управління станом та залежностями. Провайдери Riverpod виступають посередниками, надаючи доступ до даних та логіки компонентам шару представлення та дозволяючи реактивно оновлювати UI при зміні даних.

- **Authentication Logic:** Модуль, відповідальний за процеси автентифікації користувача (вхід, вихід, реєстрація). Він взаємодіє зі сховищем для безпечного зберігання токенів.

- **Data Fetching Logic:** Модуль, що відповідає за отримання, обробку та кешування даних з різних джерел. Він взаємодіє як з локальним сховищем, так і з віддаленим API. Цей шар отримує запити від шару представлення, обробляє їх, взаємодіє із шаром сховища та надає оновлені дані назад шару представлення через Riverpod Providers.

3. **Storage (Шар сховища):** Цей шар відповідає за доступ до даних, незалежно від їхнього типу та розташування:

- **Flutter Secure Storage:** Використовується для безпечного зберігання чутливих даних, таких як токени автентифікації, як вказано в попередньому тексті. Це критично важливо для забезпечення безпеки додатку.

- **Hive DB:** Локальна нереляційна база даних, яка використовується для збереження налаштувань та кешу даних в офлайн-режимі. Це дозволяє додатку працювати навіть без доступу до мережі, покращуючи користувацький досвід.

- **REST API via HTTPClient:** Механізм взаємодії із серверною частиною через REST API, ймовірно, використовуючи вбудований HTTPClient або інші бібліотеки Flutter для виконання HTTP-запитів. Цей компонент відповідає за

отримання даних з бекенду та відправлення даних на сервер. Заходи безпеки, такі як SSL Pinning, які згадано в тексті, реалізуються на цьому рівні для захисту комунікації з сервером.

Взаємозв'язки та потоки даних:

Діаграма ілюструє такі ключові взаємодії:

- Шар Presentation залежить від шару Logic через Riverpod Providers. Компоненти UI "слухають" провайдери для отримання стану та даних.
- Riverpod Providers направляють виклики до відповідних модулів логіки: Authentication Logic та Data Fetching Logic.
- Authentication Logic взаємодіє з Flutter Secure Storage для збереження/читання токенів.
- Data Fetching Logic взаємодіє з Hive DB для кешування/читання локальних даних та з REST API via HTTPClient для отримання/відправлення даних на сервер.

Така архітектура, заснована на розподіленні відповідальності між шарами та використанні Riverpod для управління станом та залежностями, забезпечує чистоту коду, зручність тестування, модульність та можливість легкої підтримки додатку. Вона також враховує специфіку мобільних додатків, надаючи механізми для роботи з локальним сховищем та забезпечення безпеки.

Таблиця 2.7 Клієнтських технологій

Компонент	Технологія	Версія	Опис
Веб-фреймворк	React + TypeScript	18	SPA, компонентна архітектура
UI-бібліотека	Ant Design	5	Готові компоненти з підтримкою темізації
Мобільний фреймворк	Flutter	3.3	Кросплатформений нативний рендеринг
Стейт-менеджмент (WEB)	Redux Toolkit	1.x	Централізоване управління станом
Стейт-менеджмент (MOB)	Riverpod	2.0	Dependency Injection, реактивність

Локальне сховище	Hive	2.x	Швидке ключ-значення сховище
Безпека	Secure Storage	—	Захист токенів, SSL Pinning

Джерело: сформовано автором на основі виконаного дослідження

Розгортання HRM-системи організовано за принципами Infrastructure as Code, Continuous Integration/Continuous Deployment (CI/CD) та контейнеризації, що забезпечує швидкість, повторюваність і якість поставок.

Контейнеризація та оркестрація

- Docker 20.x використовується для пакування кожного мікросервісу у легковагові контейнери з чітко визначеними залежностями та конфігураціями [62].
- Kubernetes 1.23+ служить платформою для оркестрації контейнерів, забезпечуючи автоматичне масштабування, балансування навантаження та self-healing. Пайплайни розгортання базуються на Helm-чартах для декларативного опису ресурсів.
- Helm 3.x дозволяє управляти версіями релізів та параметризувати конфігурації середовища (development, staging, production).

CI/CD-процеси

- GitLab CI/CD та Jenkins використовуються як основні рушії для автоматизації збірки, тестування та розгортання.
- Pipeline Stages:
 1. Build: збірка образів Docker, статичний аналіз коду (SonarQube) та npm/Gradle/Yarn залежності.
 2. Test: unit-тести (JUnit, Jest, pytest), інтеграційні тести у Docker Compose середовищі.
 3. Security Scan: SAST (Static Application Security Testing) за допомогою OWASP Dependency Check та Trivy.
 4. Package: завантаження образів у Docker Registry (Harbor або GitLab Registry).
 5. Deploy: розгортання із Canary Releases або Blue-Green Deployment за допомогою Argo CD.

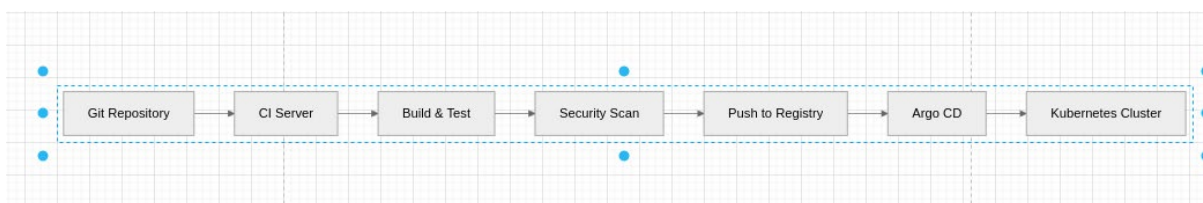


Рисунок 2.15 CI/CD Архітектурна Діаграма

Джерело: сформовано автором на основі виконаного дослідження

Моніторинг та візуалізація метрик HRM-системи здійснюються за допомогою Prometheus і Grafana, що дозволяє оперативно відстежувати продуктивність мікросервісів та реагувати на зміни в часі. Централізоване логування реалізується через ELK Stack (Elasticsearch, Logstash, Kibana) або альтернативно через EFK (Fluentd), що забезпечує гнучкий аналіз та пошук у потоках логів. Для розподіленого трасування запитів застосовується Jaeger, що допомагає візуалізувати шлях запиту через сервіси і діагностувати вузькі місця. Розгортання та обчислювальні ресурси базуються на провайдерах хмарних сервісів: AWS (EKS для Kubernetes, RDS для PostgreSQL, S3 для Data Lake), GCP (GKE, Cloud SQL, Cloud Storage) та Azure (AKS, Azure Database for PostgreSQL, Blob Storage), що забезпечує високу доступність, масштабованість та гнучкість інфраструктури.

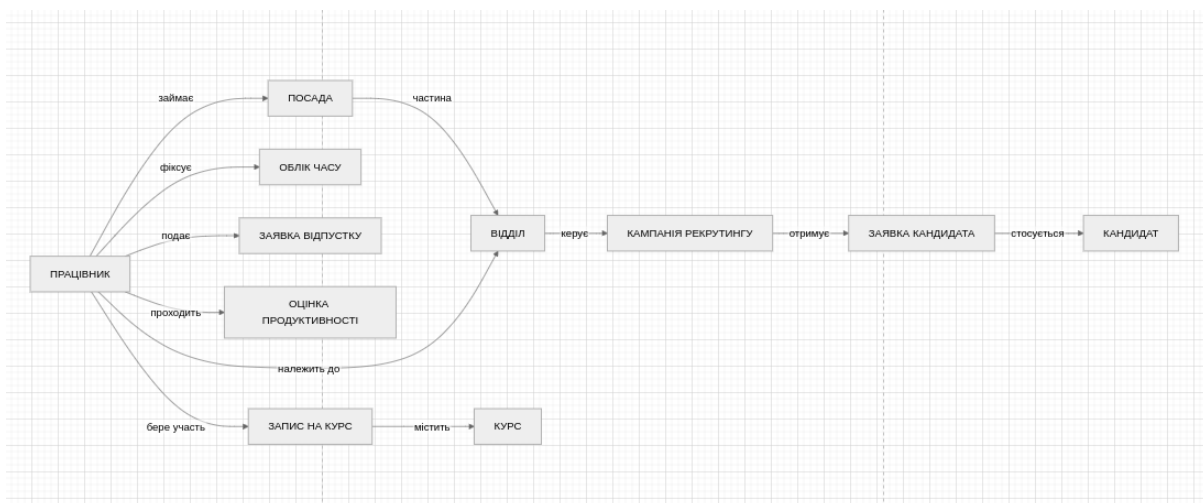


Рисунок 2.16 ER-діаграма ключових сутностей

На Рисунку 2.16 представлена ER-діаграма (діаграма "сутність-зв'язок"), яка ілюструє ключові сутності та взаємозв'язки між ними в межах проекрованої HRM-системи. Ця діаграма відображає концептуальну модель даних системи, показуючи основні об'єкти інформаційної системи та зв'язки між ними.

Основними сутностями, представленими на діаграмі, є:

1. Співробітник: Центральна сутність, що представляє особу, яка працює в організації.
2. Посада: Сутність, що описує посаду, яку обіймає співробітник.
3. Статус/Час: Сутність, ймовірно, пов'язана з обліком робочого часу, статусом зайнятості або іншими часовими параметрами, що стосуються співробітника.
4. Заяви/Відпустки: Сутність, що фіксує заяви співробітників (наприклад, на відпустку, відгул, звільнення) та інформацію про відпустки.
5. Оцінка продуктивності: Сутність для зберігання результатів оцінки продуктивності співробітників.
6. Запис на курс: Сутність, що фіксує факт запису співробітника на певний навчальний курс.
7. Курс: Сутність, що описує навчальний курс.
8. Відділ: Сутність, що представляє структурний підрозділ організації.
9. Клієнтська реакція: (Інтерпретація може варіюватися залежно від контексту системи, але виходячи з зв'язку) Сутність, що фіксує реакцію або зворотний зв'язок від клієнтів, яка, можливо, пов'язана з роботою відділів або кандидатів.
10. Заявки кандидатів: Сутність, що представляє заявки від осіб, які претендують на вакансії.
11. Кандидат: Сутність, що представляє потенційного співробітника, який подав заявку.

Взаємозв'язки між сутностями:

Діаграма ілюструє наступні ключові взаємозв'язки:

- Співробітник "має посаду" (або обіймає) Посаду. Ймовірно, це зв'язок "один-до-одного" або "один-до-багатьох" (історія посад).
- Співробітник пов'язаний зі Статус/Час, що відображає облік робочого часу або статусу.

- Співробітник "надає" або пов'язаний із Заяви/Відпустки. Один співробітник може мати багато заяв/відпусток (зв'язок "один-до-багатьох").
- Співробітник "проходить" Оцінка продуктивності. Може бути зв'язок "один-до-багатьох", якщо оцінки проводяться періодично.
- Співробітник "бере участь" у Запис на курс. Один співробітник може мати багато записів, і один запис стосується одного співробітника.
- Запис на курс "відноситься до" Курс. Один запис стосується одного курсу, але один курс може мати багато записів (зв'язок "один-до-багатьох").
- Відділ "відноситься до" (або має) Клієнтська реакція. Цей зв'язок може вказувати, що реакція клієнтів стосується конкретного відділу або оцінює його роботу.
- Клієнтська реакція пов'язана із Заявки кандидатів. Цей зв'язок менш очевидний без додаткового контексту, але може означати, що клієнтська реакція є джерелом або частиною інформації, пов'язаної із заявками кандидатів (наприклад, відгуки клієнтів про взаємодію з кандидатами на етапі співбесід).
- Заявки кандидатів "зберігається" (або стосується) Кандидат. Один кандидат може мати багато заявок, а одна заявка стосується одного кандидата (зв'язок "один-до-багатьох").

Написи на стрілках, такі як "має", "відноситься до", "надає", "проходить", "бере участь", "часткова", "зберігається", описують природу взаємозв'язку. Написи "часткова" на зв'язку між "Посада" та "Відділ", а також "Запис на курс" та "Відділ" можуть вказувати на неповну участь (partial participation), тобто не всі посади або курси обов'язково пов'язані з відділами в цій моделі.

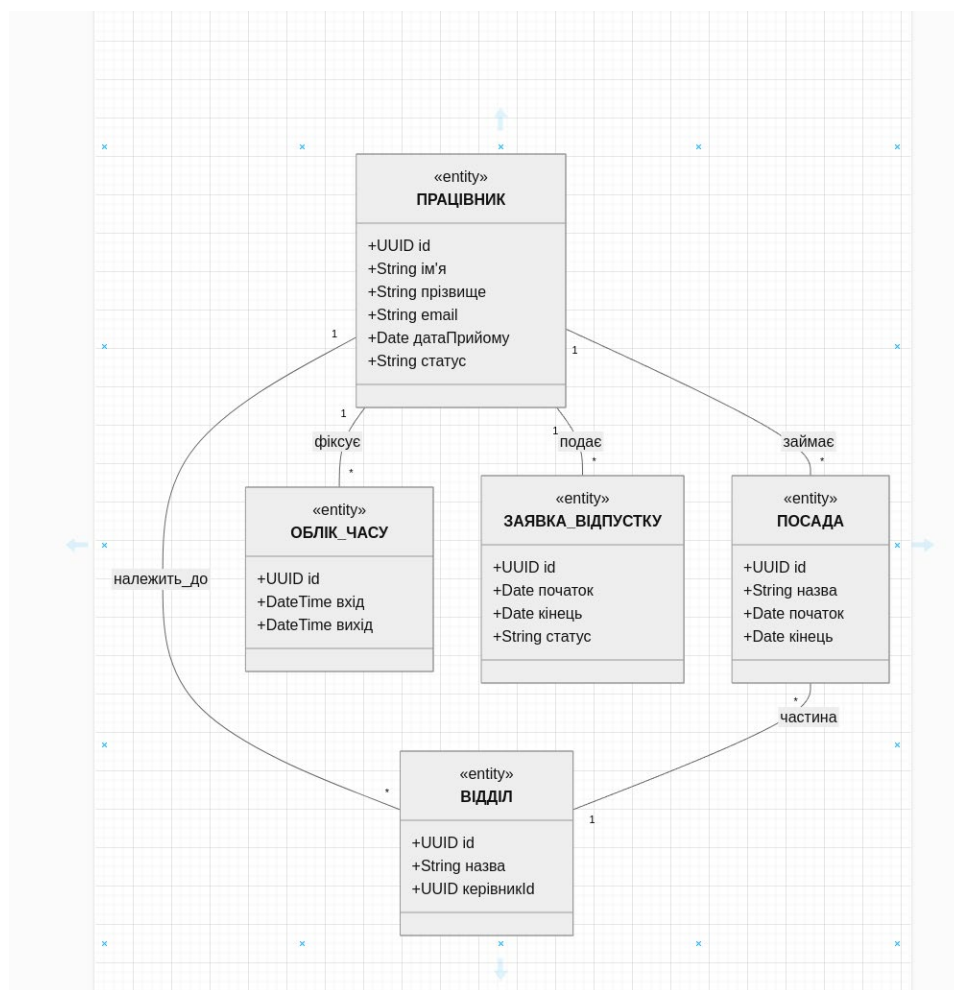


Рисунок 2.17 Діаграма сутностей із атрибутами

На Рисунку 2.17 представлена деталізована діаграма, яка відображає структуру ключових сутностей HRM-системи та їхні основні атрибути, а також взаємозв'язки між ними. На відміну від концептуальної ER-діаграми, ця діаграма надає більш глибоке розуміння складу даних для кожної сутності.

Основними сутностями та їхніми атрибутами є:

1. Працівник:

- **id**: Унікальний ідентифікатор працівника (типу UUID).
- **ім'я**: Ім'я працівника (типу String).
- **прізвище**: Прізвище працівника (типу String).
- **email**: Адреса електронної пошти працівника (типу String).
- **дата прийому**: Дата початку роботи в компанії (типу Date).
- **статус**: Поточний статус працівника (наприклад, активний, у відпустці, звільнений) (типу String).

2. Облік часу:

- id: Унікальний ідентифікатор запису обліку часу (типу UUID).
- дата/час вхід: Дата та час початку робочого дня або активності (типу DateTime).
- дата/час вихід: Дата та час закінчення робочого дня або активності (типу DateTime).

3. Заявка/Відпустка:

- id: Унікальний ідентифікатор заявки або запису про відпустку (типу UUID).
- дата початку: Дата початку періоду (наприклад, відпустки) (типу Date).
- дата кінця: Дата закінчення періоду (наприклад, відпустки) (типу Date).
- статус: Поточний статус заявки або відпустки (наприклад, подано, затверджено, відхилено) (типу String).

4. Посада:

- id: Унікальний ідентифікатор запису про посаду (типу UUID).
- назва: Назва посади (типу String).
- дата початку: Дата початку обіймання посади (типу Date).
- дата кінця: Дата закінчення обіймання посади (типу Date). Наявність дат початку та кінця вказує на можливість відстеження історії посад працівника.

5. Відділ:

- id: Унікальний ідентифікатор відділу (типу UUID).
- назва: Назва відділу (типу String).
- керівник id: Ідентифікатор працівника, який є керівником цього відділу (типу UUID). Це вказує на зв'язок між відділом та працівником-керівником.

Взаємозв'язки між сутностями:

Діаграма відображає такі взаємозв'язки, деталізуючи зв'язки, показані на Рисунку 2.16:

- Зв'язок між Працівник та Облік часу позначено як "фіксує". Це зв'язок "один-до-багатьох" (1:N), оскільки один працівник може мати множину записів про облік часу.
- Зв'язок між Працівник та Заявка/Відпустка позначено як "подає". Це також зв'язок "один-до-багатьох" (1:N), оскільки один працівник може подати множину заявок/записів про відпустку.
- Зв'язок між Працівник та Посада позначено як "займає". З огляду на атрибути дата початку та дата кінця в сутності "Посада", цей зв'язок, ймовірно, є "один-до-багатьох" (1:N), дозволяючи зберігати історію посад працівника. Тобто, працівник може мати кілька записів "Посада" за різні періоди часу.
- Зв'язок між Працівник та Відділ позначено як "належить до". Це зв'язок "багато-до-одного" (N:1), оскільки багато працівників належать до одного відділу.
- Зв'язок між Посада та Відділ позначено як "частина". Це зв'язок "багато-до-одного" (N:1), оскільки багато посад можуть існувати в рамках одного відділу.

Наявність атрибута керівник id у сутності "Відділ" також вказує на додатковий зв'язок між "Відділ" та "Працівник", де працівник виконує роль керівника.

РОЗДІЛ 3. РОЗРОБКА ТА ПРОЄКТУВАННЯ HRM СИСТЕМ

3.1. Архітектура системи

Проектування архітектури інформаційної системи є критично важливим етапом розробки, що визначає структурну організацію системи, її компоненти, їхні взаємозв'язки та принципи функціонування. Обрана архітектура має забезпечувати відповідність системи функціональним та нефункціональним вимогам, таким як масштабованість, надійність, безпека, зручність супроводу та розширення. В контексті розробки HRM-системи було прийнято рішення про застосування мікросервісної архітектури в поєднанні з багатошаровим підходом, що дозволяє досягти високої гнучкості та ефективності. Детальна архітектура серверної частини HRM-системи базується на принципах мікросервісів, де кожен сервіс відповідає за окрему бізнес-функцію та взаємодіє з іншими сервісами через чітко визначені інтерфейси. Такий підхід забезпечує незалежність розробки, розгортання та масштабування окремих компонентів системи. Загальна структура компонентів та їх взаємодія представлена на Рисунку 3.1.

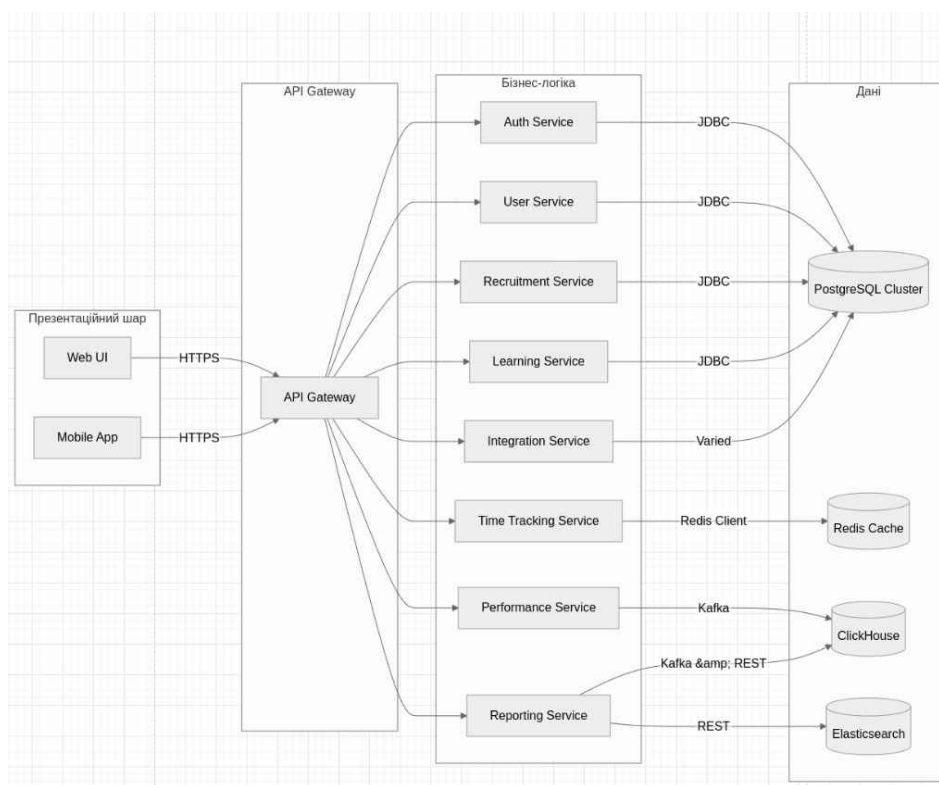


Рисунок 3.1 Детальна схема компонентів і їх взаємодії

Джерело: сформовано автором на основі виконаного дослідження

Як видно з Рисунку 3.1, архітектура серверної частини поділена на кілька логічних шарів:

- **Презентаційний шар:** Включає клієнтські додатки – веб-інтерфейс (Web UI) та мобільний додаток (Mobile App). Ці компоненти відповідають за взаємодію з кінцевим користувачем та надсилання запитів до серверної частини. Комунікація здійснюється за протоколом HTTPS, що забезпечує шифрування та безпеку передачі даних.
- **API Gateway:** Виступає єдиною точкою входу для всіх зовнішніх запитів до мікросервісів бізнес-логіки. API Gateway виконує функції маршрутизації запитів до відповідних сервісів, аутентифікації, авторизації, балансування навантаження та агрегації відповідей. Це дозволяє приховати внутрішню структуру мікросервісів від клієнтів та забезпечує централізоване управління доступом.
- **Бізнес-логіка:** Шар, що містить основні мікросервіси, які реалізують бізнес-функціональність HRM-системи. Кожен сервіс має власну відповідальність: Auth Service (автентифікація та авторизація), User Service (управління користувачами), Recruitment Service (рекрутинг), Learning Service (навчання), Integration Service (інтеграція із зовнішніми системами), Time Tracking Service (облік робочого часу), Performance Service (оцінка продуктивності), Reporting Service (формування звітів). Ці сервіси взаємодіють між собою та з шаром даних.
- **Дані:** Шар, що включає різні сховища даних, оптимізовані для різних типів інформації та завдань. PostgreSQL Cluster використовується для зберігання транзакційних даних, забезпечуючи їх цілісність та надійність. Redis Cache виступає як швидке кеш-сховище для часто використовуваних даних та сесій. ClickHouse застосовується для аналітичних запитів та обробки великих обсягів даних (OLAP). Elasticsearch використовується для повнотекстового пошуку та агрегації. Взаємодія між сервісами бізнес-логіки та сховищами даних відбувається з використанням відповідних протоколів та клієнтів (JDBC для PostgreSQL, Redis Client для Redis, Kafka для потокової обробки даних, REST для Elasticsearch).

Взаємодія між компонентами в мікросервісній архітектурі відбувається переважно через API Gateway, який маршрутизує запити до відповідних сервісів

бізнес-логіки. Сервіси можуть взаємодіяти між собою безпосередньо або опосередковано через брокери повідомлень (наприклад, Kafka), особливо для асинхронних операцій та обміну подіями.

Для ілюстрації типового сценарію взаємодії компонентів розглянемо процес автентифікації користувача. Цей процес детально зображено на діаграмі послідовності на Рисунку 3.2.

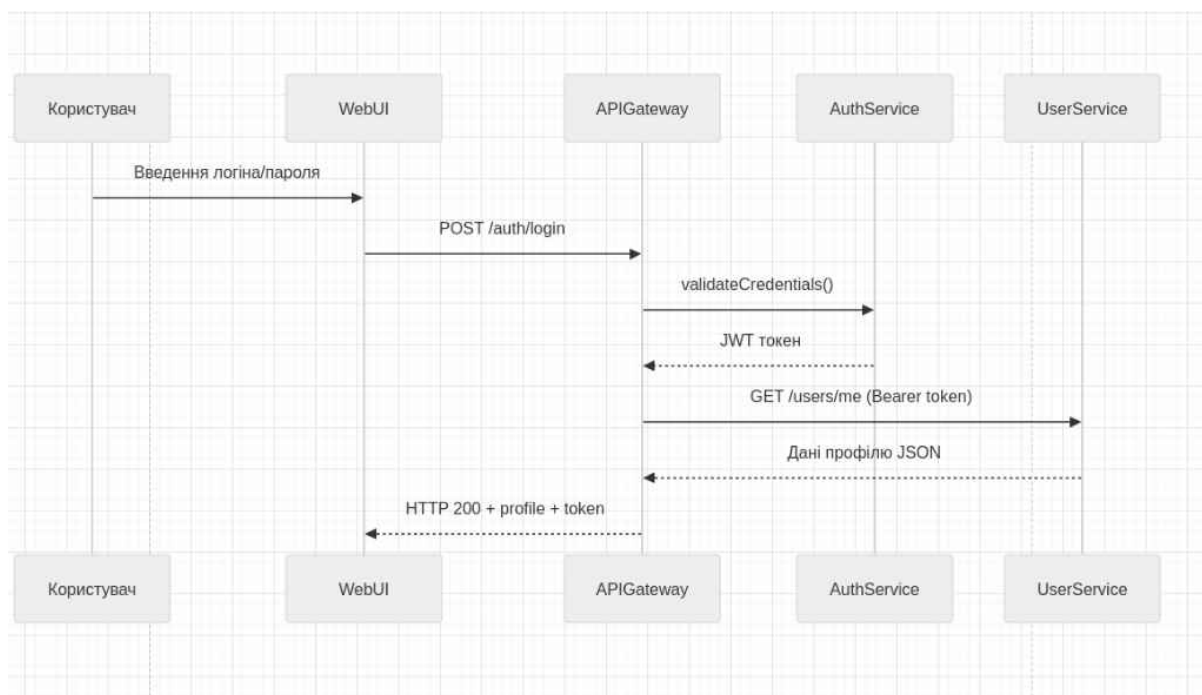


Рисунок 3.2 Діаграма послідовності

Джерело: сформовано автором на основі виконаного дослідження

Згідно з Рисунком 3.2, послідовність дій при автентифікації виглядає наступним чином:

1. Користувач вводить логін та пароль у WebUI.
2. WebUI надсилає HTTP POST запит до `/auth/login` на APIGateway, передаючи облікові дані користувача.
3. APIGateway маршрутизує запит до AuthService.
4. AuthService валідує облікові дані користувача, можливо, звертаючись до сховища даних користувачів (на діаграмі не показано безпосередньо, але мається на увазі взаємодія з User Service або базою даних).
5. У разі успішної валідації, AuthService генерує JWT токен та повертає його до APIGateway.

6. APIGateway, отримавши токен, може виконати додаткові дії, наприклад, отримати дані профілю користувача, надіславши GET запит до `/users/me` (з токеном як Bearer token) на UserService.

7. UserService за токеном ідентифікує користувача та повертає дані профілю у форматі JSON до APIGateway.

8. APIGateway формує кінцеву відповідь, що включає HTTP статус 200 (OK), дані профілю та JWT токен, і повертає її до WebUI.

9. WebUI отримує відповідь та використовує токен для подальших автентифікованих запитів.

Ця діаграма послідовності демонструє, як запит користувача проходить через різні компоненти системи, забезпечуючи безпеку та розподіл відповідальності.

HRM-система не є ізольованою і потребує інтеграції з іншими корпоративними системами для обміну даними та забезпечення цілісності бізнес-процесів. Інтеграційні точки та протоколи обміну відіграють ключову роль у забезпеченні безшовної взаємодії із зовнішнім середовищем. На Рисунку 3.3 представлено схему інтеграції HRM-системи із зовнішніми системами.

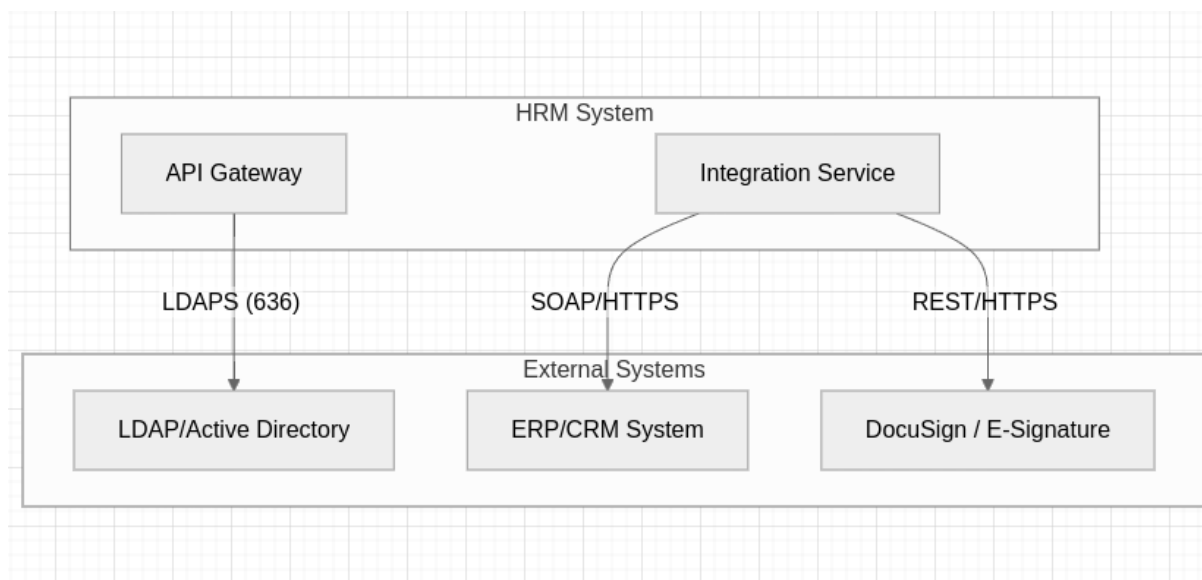


Рисунок 3.3 Схема інтеграції HRM-системи із зовнішніми системами.

Джерело: сформовано автором на основі виконаного дослідження

Згідно з Рисунком 3.3, інтеграція відбувається з такими зовнішніми системами:

- **LDAP/Active Directory:** Система директорій, яка використовується для централізованого управління користувачами та їхніми обліковими записами. Інтеграція з LDAP/AD дозволяє синхронізувати дані про співробітників, виконувати аутентифікацію користувачів та управляти правами доступу. Взаємодія здійснюється за протоколом LDAPS (LDAP Secure), що працює на порту 636 та забезпечує шифровану передачу даних. Зв'язок відбувається через API Gateway, що може виступати як проксі або адаптер для LDAP.

- **ERP/CRM System:** Корпоративні системи планування ресурсів підприємства або управління відносинами з клієнтами. Ці системи можуть містити значну кількість даних про співробітників, їхню діяльність, фінансові показники тощо, які необхідні для повноцінного функціонування HRM-системи. Інтеграція з ERP/CRM системою забезпечує обмін цими даними. На діаграмі показано, що інтеграція може відбуватися за протоколом SOAP/HTTPS, що є поширеним стандартом для обміну структурованими даними в корпоративному середовищі. За взаємодію з боку HRM-системи відповідає Integration Service, який спеціалізується на взаємодії із зовнішніми сервісами.

- **DocuSign / E-Signature:** Система електронного підпису, яка може використовуватися для підписання документів, пов'язаних із трудовими відносинами (наприклад, трудові договори, додаткові угоди, накази). Інтеграція з такою системою спрощує та автоматизує процеси документообігу. Взаємодія може відбуватися за протоколом REST/HTTPS, що є сучасним та гнучким підходом до побудови API. Ця інтеграція також реалізується через Integration Service.

Таким чином, інтеграційні точки представлені взаємодією HRM-системи з LDAP/Active Directory, ERP/CRM System та DocuSign/E-Signature. Використання стандартних протоколів (LDAPS, SOAP/HTTPS, REST/HTTPS) та спеціалізованого сервісу для інтеграції (Integration Service) забезпечує гнучкість, надійність та безпеку обміну даними із зовнішніми системами, що є необхідною умовою для функціонування HRM-системи в складі загальної інформаційної інфраструктури підприємства.

3.2 Проєктування бази знань інформаційної управляючої системи та/або засоби інтелектуального аналізу даних

3.2.1 Структура бази знань

Загальна архітектура та призначення

Представлена модель бази даних розроблена для комплексного управління людськими ресурсами підприємства. Вона базується на реляційному підході з чітко визначеними первинними (PK) та зовнішніми (FK) ключами, а також використанням індексів для оптимізації швидкості доступу до даних. Дотримання третьої нормальної форми (3NF) гарантує мінімізацію дублювання інформації та підтримку цілісності даних, що є критично важливим для надійної HR-системи.

Основні сутності: Співробітники, Відділи та Посади

Центральною сутністю системи є таблиця `employees`, яка зберігає всю ключову інформацію про кожного співробітника, включаючи ідентифікатор, ім'я, прізвище, контактні дані, дату найму та поточний статус. Організаційна структура компанії відображена в таблиці `departments`, де кожен відділ має унікальний ідентифікатор, назву та посилання на керівника (`manager_id`), який також є співробітником. Для зв'язку співробітників з їхніми конкретними ролями в певних відділах використовується таблиця `positions`. Вона деталізує посаду, рівень заробітної плати, а також дати початку та закінчення роботи на цій позиції, ефективно фіксуючи кар'єрний шлях кожного працівника в межах організації.

Операційна діяльність: Облік часу та Відпустки

Для моніторингу робочого часу та присутності персоналу призначена таблиця `time_logs`. Вона фіксує точний час приходу (`timestamp_in`) та уходу (`timestamp_out`) кожного співробітника, а також тип запису (наприклад, робочий час, перерва). Управління відпустками та іншими видами відсутності реалізовано через таблицю `leave_requests`, де зберігаються заявки від співробітників із зазначенням періоду відсутності та поточного статусу заявки (наприклад, "подано", "схвалено", "відхилено").

Розвиток та оцінка персоналу

Процеси оцінки та розвитку співробітників підтримуються таблицями `performance_reviews`, `courses` та `course_enrollments`. Таблиця `performance_reviews` зберігає результати періодичних оцінок продуктивності, включаючи дату проведення, отриманий бал та ідентифікатор оцінювача. Для підвищення кваліфікації персоналу існує каталог навчальних програм у таблиці `courses`, що містить опис курсів та їхню тривалість. Таблиця `course_enrollments` відстежує записи співробітників на конкретні курси, дати зарахування та статус завершення навчання.

Рекрутинг та управління кандидатами

Модуль рекрутингу охоплює процеси залучення нових талантів. Таблиця `recruitment_campaigns` дозволяє планувати та відстежувати кампанії з набору персоналу, вказуючи цільовий відділ, назву кампанії та її терміни. Інформація про потенційних кандидатів зберігається в таблиці `candidates`. Зв'язок між кандидатами та активними рекрутинговими кампаніями, а також статус їхніх заявок та посилання на резюме, фіксується в таблиці `candidate_applications`.

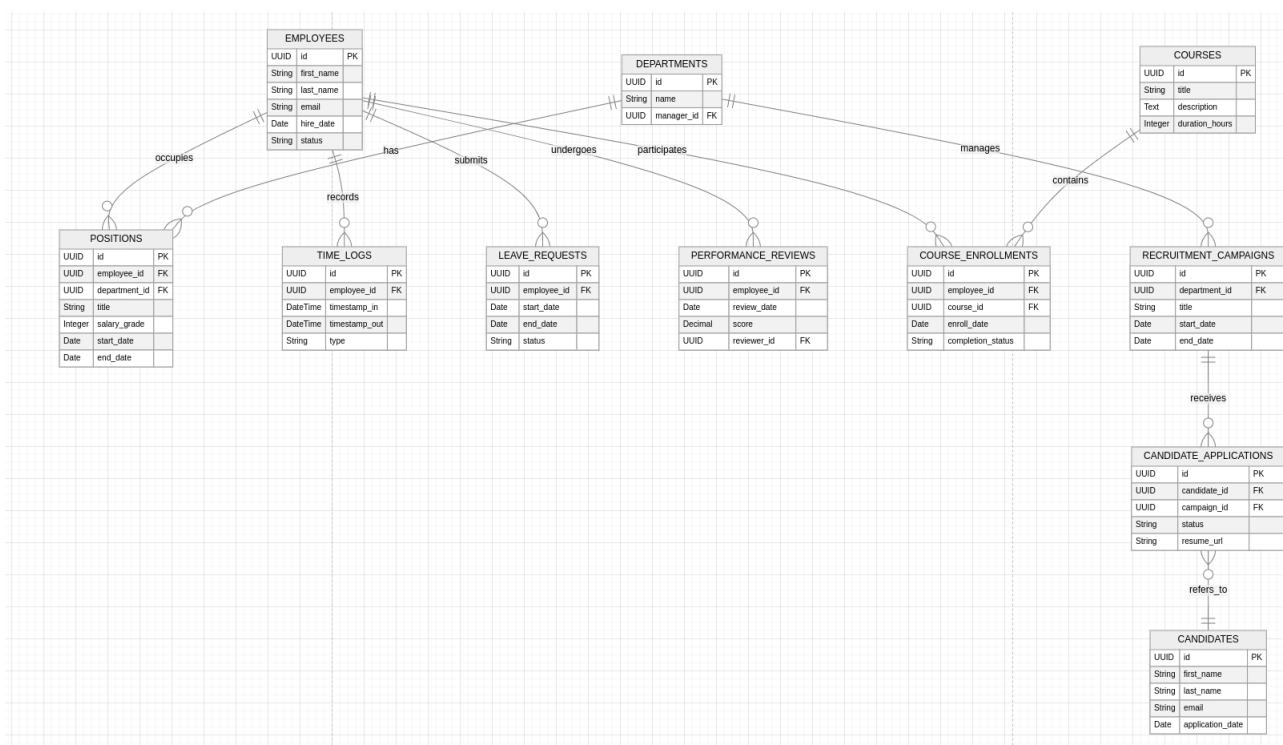


Рисунок 3.4 Діаграма фізичної моделі

Джерело: сформовано автором на основі виконаного дослідження

3.2.2 Функціональне моделювання системи

Функціональне моделювання системи є важливим етапом аналізу та проектування, що дозволяє визначити, які функції має виконувати система, та які процеси вона підтримує. Для візуалізації функціональних вимог та ключових бізнес-процесів HRM-системи було використано діаграми варіантів використання (Use Case діаграми) та діаграми діяльності (Activity діаграми) згідно з нотацією UML (Unified Modeling Language).

Діаграми варіантів використання (Use Case діаграми) є інструментом для моделювання функціональних вимог системи з точки зору зовнішніх суб'єктів (акторів), які взаємодіють із системою. Кожен варіант використання (Use Case) описує послідовність дій, яку система виконує у відповідь на дію актора, для досягнення певної значущої для актора мети.

На Рисунку в додатку А представлена Use Case діаграма ключових варіантів використання HRM-системи.

Діаграма ідентифікує наступних основних акторів, які взаємодіють із системою:

- Кандидат: Особа, яка претендує на вакантну посаду в компанії.
- Керівник: Співробітник, який управляє підлеглими та бере участь у процесах оцінки, рекрутингу тощо.
- HR Менеджер: Співробітник відділу кадрів, який відповідає за управління персоналом, рекрутинг, атестацію та інші HR-процеси.
- Співробітник: Штатний працівник компанії.

Діаграма відображає наступні ключові варіанти використання, доступні для відповідних акторів:

- Подати резюме (Кандидат): Варіант використання, що дозволяє кандидату надіслати своє резюме для розгляду на вакантну посаду.
- Провести співбесіду (Керівник): Варіант використання для керівника, що передбачає проведення співбесід з кандидатами.
- Створити вакансію (HR Менеджер): Варіант використання, що дозволяє HR менеджеру додавати нові вакансії до системи.

- Обробити резюме (HR Менеджер): Варіант використання для HR менеджера, що включає перегляд, оцінку та обробку резюме кандидатів.
- Налаштувати OKR (HR Менеджер, Керівник): Варіант використання, пов'язаний з процесом постановки цілей та ключових результатів (OKR) для співробітників або відділів. Доступний як HR менеджеру, так і керівнику.
- Оцінити продуктивність (Керівник): Варіант використання, що дозволяє керівнику оцінювати продуктивність своїх підлеглих.
- Почати атестацію (HR Менеджер): Варіант використання для ініціювання процесу атестації співробітників.
- Надати фідбек (Співробітник): Варіант використання, що дозволяє співробітнику надавати зворотний зв'язок (наприклад, в рамках процесу атестації або оцінки 360 градусів).

Кожен з цих варіантів використання представляє набір можливих сценаріїв взаємодії між актором та системою. Наприклад, варіант використання "Подати резюме" може включати сценарії успішного подання резюме, подання з помилками, збереження чернетки тощо. Детальний опис сценаріїв для кожного ключового варіанта використання є частиною специфікації функціональних вимог та може бути представлений у вигляді таблиць або текстових описів в окремому підрозділі.

Діаграми діяльності (Activity діаграми) є інструментом для моделювання потоків робіт та діяльності в системі або бізнес-процесах. Вони дозволяють візуалізувати послідовність дій, точки розгалуження, злиття та паралельне виконання операцій. Для кращого розуміння ключових бізнес-процесів, які автоматизує HRM-система, було розроблено Activity-діаграми для процесів рекрутингу та атестації.

На Рисунку 3.5 представлена Activity-діаграма процесу рекрутингу.

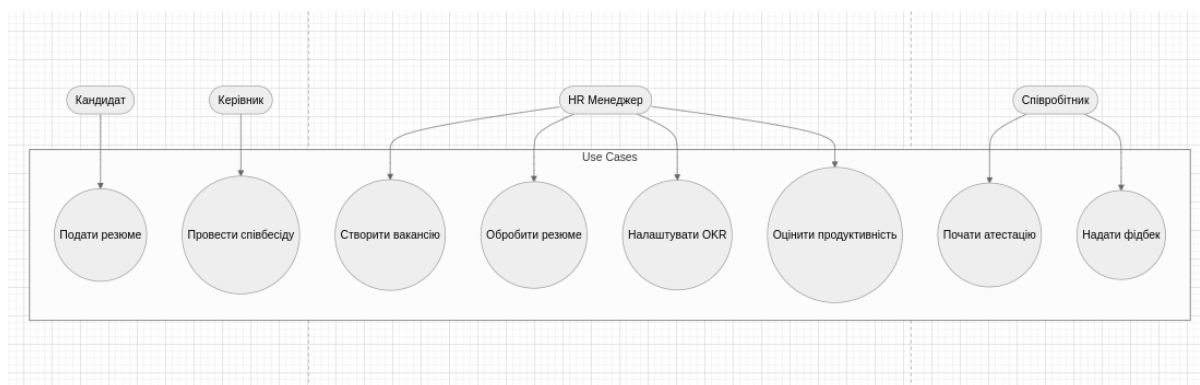


Рисунок 3.5 Діаграма процесу рекрутингу

Джерело: сформовано автором на основі виконаного дослідження

Діаграма ілюструє наступну послідовність дій у процесі рекрутингу:

1. Початок процесу: Стартова точка діаграми.
2. Формування вакансії: Визначення потреб у новому співробітнику та створення опису вакансії.
3. Публікація вакансії: Розміщення інформації про вакансію на внутрішніх та зовнішніх ресурсах.
4. Збір резюме: Етап активного надходження та збору резюме від кандидатів.
5. Завершення збору?: Точка ухвалення рішення. Якщо збір не завершено ("Ні"), процес повертається до етапу "Збір резюме".
6. Відбір кандидатів: Після завершення збору резюме ("Так") відбувається аналіз отриманих резюме та відбір кандидатів, які відповідають вимогам.
7. Планування інтерв'ю: Організація та планування співбесід з відібраними кандидатами.
8. Проведення інтерв'ю: Безпосереднє проведення співбесід.
9. Пропозиція роботи: Точка ухвалення рішення щодо подальших дій після інтерв'ю.
10. Надіслати пропозицію (Так): Якщо прийнято позитивне рішення, кандидату надсилається пропозиція про роботу.

11. Повідомити кандидата (Ні): Якщо прийнято негативне рішення, кандидату надсилається повідомлення про відмову.

12. Завершення: Кінцева точка процесу. Обидва шляхи (успішна пропозиція або відмова) ведуть до завершення процесу для конкретного кандидата/вакансії.

На рисунку в додатку б представлена Activity-діаграма циклу атестації співробітників.

Діаграма відображає таку послідовність дій у процесі атестації:

1. Початок циклу атестації: Стартова точка процесу.
2. Налаштування KPI/OKR: Визначення та узгодження ключових показників ефективності (KPI) або цілей та ключових результатів (OKR) для співробітника на період атестації.
3. Запуск опитування 360°: Ініціювання процесу збору зворотного зв'язку про співробітника від його колег, керівників, підлеглих тощо.
4. Збір фідбеку: Етап надходження та збору відповідей в рамках опитування 360°.
5. Усі відповіді отримані?: Точка ухвалення рішення. Якщо не всі відповіді отримані ("Ні"), процес повертається до етапу "Збір фідбеку".
6. Аналіз результатів: Після отримання всіх відповідей ("Так") проводиться аналіз зібраного фідбеку та результатів KPI/OKR.
7. Підготовка звіту: Формування підсумкового звіту за результатами атестації.
8. Обговорення з співробітником: Проведення зустрічі з співробітником для обговорення результатів атестації та звіту.
9. Встановлення індивідуального плану розвитку: На основі результатів атестації та обговорення формується індивідуальний план розвитку співробітника.
10. Кінець циклу: Кінцева точка процесу атестації для конкретного співробітника.

Ці Activity-діаграми наочно демонструють логіку та послідовність дій у ключових бізнес-процесах HRM-системи, що є основою для реалізації відповідної функціональності в програмному забезпеченні. Вони дозволяють переконатися в повноті покриття процесу системою та виявити потенційні "вузькі місця" або можливості для оптимізації.

3.3. Проектування інтерфейсу користувача

Проектування інтерфейсу користувача (UI) та досвіду користувача (UX) є невід'ємною частиною процесу розробки сучасної інформаційної системи. Ефективний UI/UX дизайн забезпечує зручність використання, інтуїтивно зрозумілу навігацію та приємний досвід взаємодії користувача із системою, що безпосередньо впливає на продуктивність та задоволеність користувачів. На етапі проектування інтерфейсу користувача HRM-системи було враховано потреби різних категорій користувачів (співробітники, керівники, HR-менеджери, кандидати) та специфіку їхніх завдань.

Для візуалізації структури та розташування елементів на ключових екранах системи, а також для узгодження функціональності та логіки взаємодії перед етапом безпосередньої реалізації, були розроблені вайрфрейми (wireframes) та інтерактивні прототипи основних користувацьких інтерфейсів. Вайрфрейми являють собою схематичне зображення екранів, що фокусується на функціональності, контенті та пріоритетності елементів, ігноруючи візуальний дизайн. Прототипи, у свою чергу, додають інтерактивність, дозволяючи імітувати потік користувача та перевіряти сценарії взаємодії.

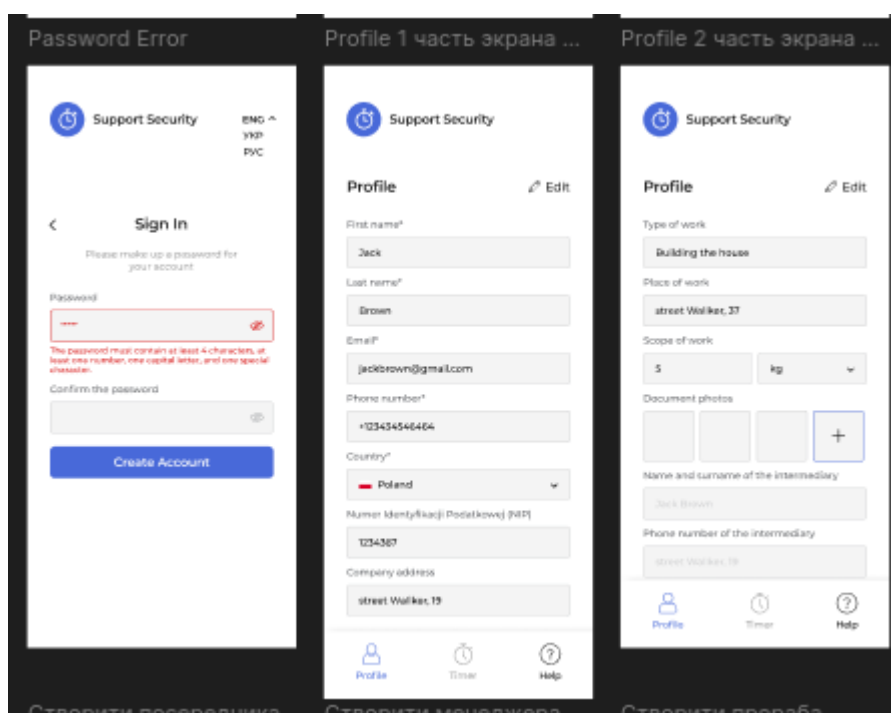


Рисунок 3.6 Вайрфрейми екрану мобільного додатку.

Джерело: сформовано автором на основі виконаного дослідження

На Рисунку 3.6 представлено вайрфрейми або прототипи основних екранів мобільного додатку HRM-системи.

Ці вайрфрейми/прототипи охоплюють ключові розділи системи, такі як:

- Головний екран/Дашборд: Надає швидкий доступ до основної інформації та дій, персоналізованих для ролі користувача.
- Профіль користувача: Дозволяє переглядати та редагувати особисті дані, інформацію про посаду, відділ тощо.
- Розділ "Заяви/Відпустки": Надає можливість подавати заяви на відпустку/відгул, переглядати їхній статус, історію.
- Розділ "Облік часу": Призначений для внесення та перегляду даних обліку робочого часу.
- Розділ "Оцінка продуктивності": Дозволяє переглядати результати оцінки, OKR, KPI.
- Розділ "Навчання": Надає доступ до доступних курсів та записів на них.
- Розділ "Рекрутинг" (для відповідних ролей): Дозволяє управляти вакансіями, переглядати та обробляти резюме кандидатів.

Розробка вайрфреймів та прототипів дозволила на ранніх етапах виявити потенційні проблеми зручність використання, оптимізувати потік користувача та забезпечити відповідність інтерфейсу потребам бізнес-процесів, описаних у розділі 3.2.

При проєктуванні інтерфейсу користувача HRM-системи були застосовані загальноприйняті UX-патерни та принципи, спрямовані на створення інтуїтивно зрозумілого та ефективного досвіду взаємодії. Зокрема, використовувалися такі патерни:

- Dashboard Pattern: Надання персоналізованого оглядового екрана (дашборда), що агрегує ключову інформацію та показники, релевантні для конкретного користувача та його ролі.

- **Form Design Patterns:** Використання зрозумілих та послідовних форм для введення даних (наприклад, при подачі заяви, редагуванні профілю), з чіткими мітками, валідацією введених даних та зворотним зв'язком про помилки.
- **Table/Data Grid Patterns:** Ефективне представлення великих обсягів табличних даних (списки співробітників, вакансій, заяв) з можливістю сортування, фільтрації, пошуку та пагінації.
- **Workflow Patterns:** Візуалізація та підтримка ключових бізнес-процесів (як-от подача заяви на відпустку або процес рекрутингу) через послідовні кроки в інтерфейсі, надаючи користувачеві чітке розуміння поточного етапу та необхідних дій.
- **Feedback Patterns:** Надання користувачеві миттєвого зворотного зв'язку про результат його дій (наприклад, повідомлення про успішне збереження даних, помилку валідації, статус обробки заявки).

Навігаційна структура системи спроектована таким чином, щоб забезпечити легкий та швидкий доступ до всіх функцій та розділів. В веб-інтерфейсі використовується комбінована навігація:

- **Основне меню:** Зазвичай розташоване збоку або зверху, надає доступ до основних розділів системи (наприклад, "Мій профіль", "Відпустки", "Облік часу", "Рекрутинг", "Звіти").
- **Хлібні крихти (Breadcrumbs):** Відображають шлях користувача до поточного екрана, допомагаючи орієнтуватися в ієрархічній структурі розділів.
- **Контекстна навігація:** Посилання або кнопки, що з'являються в контексті конкретного контенту або завдання, спрямовуючи користувача до пов'язаної інформації або дій.

У мобільному додатку навігація адаптована під особливості мобільних платформ, часто використовуючи нижню панель навігації для швидкого доступу до найбільш важливих розділів та бічне меню ("гамбургер-меню") для доступу до менш частих функцій або налаштувань. Використання бібліотеки GoRouter, як

згадано раніше, забезпечує декларативне управління маршрутами та навігацією між екранами.

Загалом, проєктування інтерфейсу користувача та навігації було спрямоване на створення зручного, ефективного та приємного досвіду для всіх категорій користувачів HRM-системи, що є критично важливим для її успішного впровадження та використання.

3.4. Реалізація інформаційної системи

Етап реалізації передбачає безпосереднє написання коду програмного забезпечення відповідно до розробленої архітектури та функціональних вимог. Реалізація HRM-системи здійснювалася з використанням сучасних технологій та підходів, що забезпечують високу якість коду, продуктивність та можливість подальшого розвитку системи.

Ефективна розробка програмного забезпечення вимагає правильно налаштованого середовища та автоматизованих процесів інтеграції та розгортання. Для реалізації HRM-системи було налаштовано стандартизоване середовище розробки та конвеєр CI/CD (Continuous Integration/Continuous Deployment).

Для розробки серверної частини (мікросервісів на Java та Python) та клієнтських частин (веб на React, мобільний додаток на Flutter) використовувалися відповідні інтегровані середовища розробки (IDE), такі як IntelliJ IDEA (для Java), PyCharm (для Python), VS Code (для React та Flutter). Всі розробники використовували спільні налаштування форматування коду, лінери та статичні аналізатори для забезпечення його якості та єдиного стилю. Управління залежностями здійснювалося за допомогою стандартних для обраних технологій інструментів (Maven/Gradle для Java, pipenv/Poetry для Python, npm/yarn для React, pub для Flutter).

CI/CD конвеєр:

Було налаштовано конвеєр безперервної інтеграції та безперервного розгортання з використанням інструментів автоматизації. Типова конфігурація CI/CD включає:

1. Система контролю версій: Використовується Git з віддаленим репозиторієм (наприклад, GitLab, GitHub, Bitbucket). Всі зміни коду фіксуються та зберігаються централізовано.

2. Сервер CI (наприклад, Jenkins, GitLab CI, GitHub Actions): Автоматично запускає процеси інтеграції при кожному коміті або запиті на злиття (merge request).

3. Етапи CI:

- Збірка (Build): Компіляція коду (для Java, Flutter), збірка артефактів (JAR-файли для Java, Docker-образи, веб-збірки, мобільні APK/IPA).
- Тестування (Testing): Автоматичний запуск модульних, інтеграційних та інших видів тестів для перевірки функціональності та відсутності регресій.
- Аналіз коду (Code Analysis): Використання інструментів статичного аналізу (наприклад, SonarQube) для перевірки якості коду, пошуку вразливостей та відповідності стандартам кодування.
- Створення образів/пакетів: У разі успішного проходження попередніх етапів створюються виконувані артефакти (наприклад, Docker-образи мікросервісів).

4. Реєстр артефактів/образів: Зібрані артефакти або Docker-образи зберігаються в централізованому сховищі.

5. Етапи CD (для автоматизованого розгортання):

- Розгортання в тестовому середовищі: Автоматизоване розгортання нової версії системи в тестовому середовищі для подальшого ручного або автоматизованого тестування.
- Розгортання в production-середовищі: Після успішного тестування в тестовому середовищі, нова версія автоматично або напівавтоматично розгортається в production-середовищі.

Налаштування CI/CD конвеєра дозволило автоматизувати процеси збірки, тестування та розгортання, значно прискоривши цикл розробки, підвищивши якість програмного забезпечення та зменшивши ризик помилок при розгортанні.

Відповідно до мікросервісної архітектури, реалізація системи розподілена між незалежними сервісами, кожен з яких є окремим модулем коду з чітко визначеними функціями та інтерфейсами. Кожен мікросервіс, як правило, має внутрішню структуру, що включає контролери (для обробки вхідних запитів API), сервіси (для реалізації бізнес-логіки) та репозиторії (для взаємодії зі сховищем даних).

- **Auth Service:** Містить класи для обробки запитів на аутентифікацію та авторизацію (AuthController), сервіси для валідації облікових даних та генерації токенів (AuthService, TokenService), а також може взаємодіяти з репозиторіями для доступу до даних користувачів або інтегруватися з LDAP/AD. API цього сервісу включає кінцеві точки для входу користувача (POST /login), реєстрації, оновлення токенів тощо.

- **User Service:** Містить класи для управління даними користувачів (UserController), сервіси для виконання операцій над користувачами (створення, читання, оновлення, видалення - CRUD) (UserService), та репозиторії для взаємодії з базою даних користувачів (UserRepository). API включає кінцеві точки для управління користувачами (GET /users/{id}, POST /users, PUT /users/{id} тощо).

- **Recruitment Service:** Включає класи для управління вакансіями та заявками кандидатів (VacancyController, CandidateController), сервіси для реалізації бізнес-логіки рекрутингу (VacancyService, CandidateService), та репозиторії для доступу до відповідних даних (VacancyRepository, CandidateRepository). API надає кінцеві точки для створення, перегляду, оновлення вакансій та обробки заявок кандидатів.

- **Time Tracking Service:** Містить класи для управління записами обліку часу (TimeEntryController), сервіси для обробки даних обліку часу (TimeEntryService), та репозиторії (TimeEntryRepository). API надає функціональність для додавання, редагування та перегляду записів робочого часу.

Аналогічним чином структуровані й інші сервіси бізнес-логіки, кожен з яких фокусується на своїй предметній області та надає відповідний API для взаємодії. Взаємодія між сервісами відбувається через API Gateway або внутрішні канали комунікації (наприклад, брокери повідомлень).

Клієнтські частини (веб та мобільний додаток) організовані відповідно до обраних фреймворків (React, Flutter).

- **Веб-інтерфейс:** Розділений на компоненти (React Components), які відповідають за відображення окремих частин інтерфейсу користувача. Логіка

управління станом зосереджена в Redux Store, а взаємодія з API реалізується через RTK Query Client. Код організовано за фічами або розділами системи.

- Мобільний додаток: Також побудований на основі компонентів (віджети у Flutter). Логіка розподілена між шарами Presentation, Logic та Storage, як описано раніше. Riverpod Providers використовуються для управління станом та залежностями.

Надання конкретних фрагментів коду дозволяє ілюструвати практичну реалізацію спроектованих рішень, продемонструвати застосування обраних технологій та патернів програмування, а також підтвердити рівень володіння автором матеріалом. Обсяг та складність прикладів коду залежать від вимог до магістерської роботи та специфіки проєкту.

ВИСНОВКИ

У ході виконання магістерської роботи було розглянуто актуальні виклики та вимоги до систем управління персоналом у контексті цифрової трансформації бізнес-процесів. На підставі проведеного аналізу існуючих рішень, вимог та потреб компанії було спроектовано та обґрунтовано архітектуру мікросервісної HRM-системи, що поєднує модульність, масштабованість та високу надійність.

Перший розділ дослідження включав аналіз ринку HRM-систем, порівняння комерційних і open-source рішень та виявлення ключових функціональних і нефункціональних вимог. Це дало змогу чітко сформулювати цілі — від автоматизації кадрового обліку до підтримки комплексних HR-процесів: рекрутингу, атестації, навчання та аналітики.

У другому розділі обґрунтовано вибір технологічних компонентів та описано архітектурну топологію системи. Застосування Kubernetes, Docker, мікросервісного підходу, Redis, PostgreSQL, ClickHouse та Elasticsearch забезпечує високу відмовостійкість, продуктивність і можливість гнучкого масштабування. Визначені нефункціональні характеристики — безпека, продуктивність, зручність використання, інтегрованість та підтримка — відповідають світовим стандартам і кращим практикам.

У третьому розділі було представлено докладне проектування компонентів: UML-діаграми, Use Case та Activity-діаграми спростили розуміння взаємодії модулів системи. Опис бізнес-логіки та інтеграційних точок гарантує, що система безперешкодно взаємодіятиме з корпоративними ERP, LDAP та іншими зовнішніми сервісами.

Розроблена модель даних із суворою нормалізацією і підтримкою історичних подій забезпечує аналітичні можливості та збереження цілісності інформації. Технологічний стек і інструменти розгортання (CI/CD, Argo CD, Helm) дозволяють швидко впроваджувати зміни та підтримувати стабільність у виробничому середовищі.

Практична реалізація проєктованих компонентів дає підстави стверджувати, що запропонована HRM-система відповідає поставленим вимогам

та може бути впроваджена в організації для оптимізації управління персоналом. Перспективи подальшого розвитку включають впровадження інтелектуальних алгоритмів аналізу даних, розширення можливостей штучного інтелекту для автоматизації рутинних завдань та інтеграцію з системами корпоративного управління знаннями.

Отже, виконана магістерська робота доводить ефективність дизайн-рішень та технологічного підходу для створення сучасної HRM-платформи, здатної підвищити прозорість, оперативність та гнучкість управління людськими ресурсами в компанії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Christopher S. (2018), “Total automation. How computer algorithms change life”. Kyiv: Our format, vol. 280.
2. Лук’янихін В.О. (2004) Потенціал системи управління та персонал сучасної організації, с. 3.
3. Літорович О.В. (2020) Особливості менеджменту персоналу підприємств в умовах діджиталізації.
4. Lachapelle, Andersen, County & Wedum (2011) Аналітичні звіти та інтеграція з ERP у HRM-системах.
5. Лук’янихін В.О. (2004) Потенціал системи управління та персонал сучасної організації, с. 6.
6. Tatiana Kolmykova, Vadym Lukianychin, Alexander Telizenko (2013) Освітня компонента формування людського капіталу.
7. Козак К.Б., Седіков Д.В., Седікова І.О. (2022) Управління персоналом в умовах глобальних інформаційних процесів: Економіка харчової промисловості, Том 14, с. 52–53.
8. Байстрюченко Н.О., Лук’янихін В.О., Теліженко О.М. (2015) Освітня компонента формування людського капіталу на основі компетентнісного підходу, с. 75.
9. [Електронний ресурс] Digital transformation and the evolution of HR trends. <http://infosystems.mu/digital-transformation-and-the-evolution-of-hr-trends>
10. Бабала Л., Данилюк І., Хома Н. (2022) HRM-системи управління персоналом, с. 241.
11. Євтушенко Г.І., Шевченко О.С. (2019) Інноваційні методи управління персоналом на підприємствах у сучасних умовах господарювання.
12. [Електронний ресурс] Deloitte. “Human Capital Trends Deutschland”. <https://www2.deloitte.com/de/de/pages/human-capital/articles/human-capital-trends-deutschland.html>

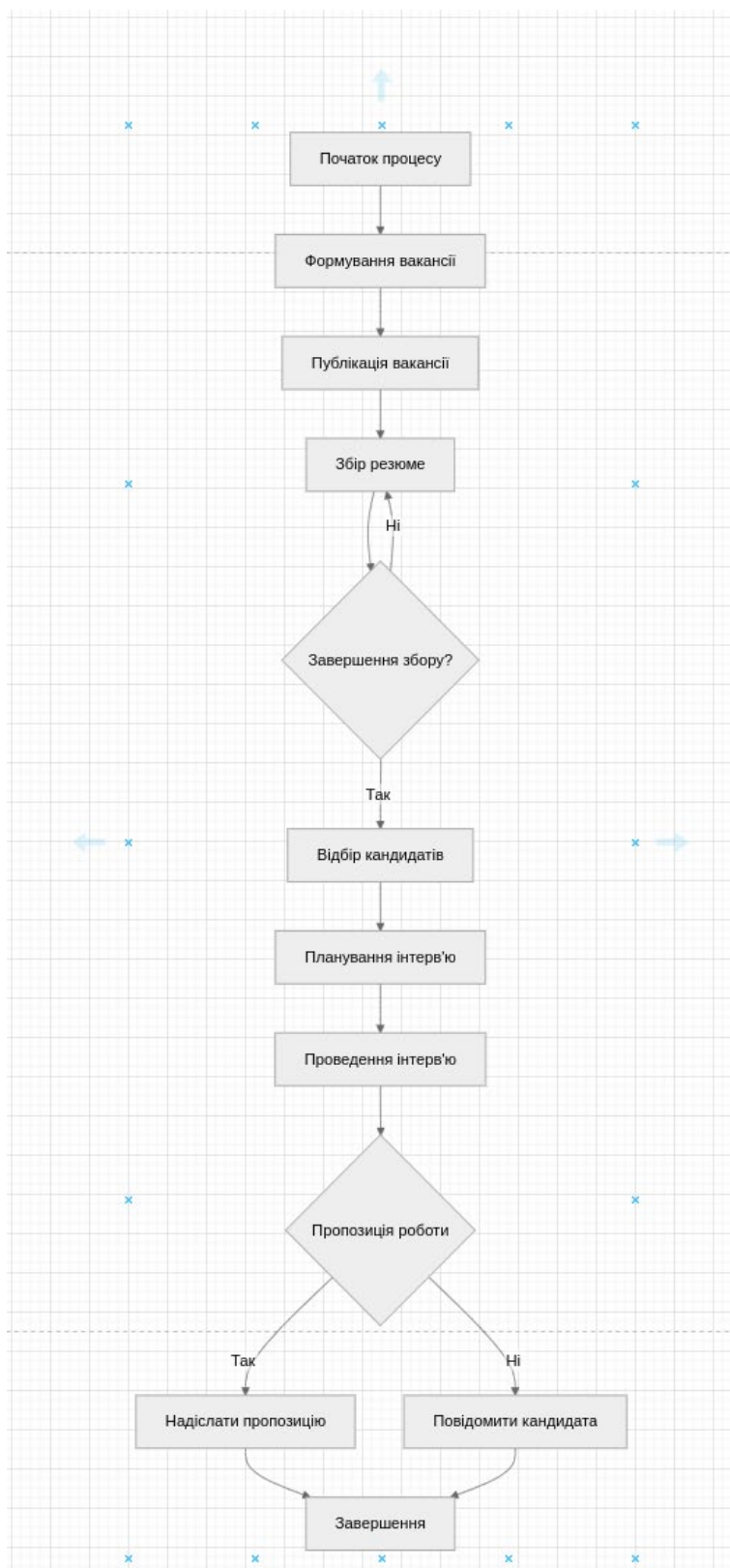
13. Capgemini Consulting, MIT Sloan School of Management. “The Digital Advantage: How digital leaders outperform their peers in every industry. Transform to the power of digital”.
14. Wood M. “The best HR software for small business: 7 industry-leading options.”
15. International Institute of Business Analysis. (2015). *A Guide to the Business Analysis Body of Knowledge® (BABOK® Guide)*, Version 3.0.
16. Fowler F.J. (2014). *Survey Research Methods*. SAGE Publications.
17. Davenport T.H. (1993). *Process Innovation: Reengineering Work through Information Technology*. Harvard Business School Press.
18. Purcell J., Kinnie N., Hutchinson S., Rayton B., Swart J. (2007). *People and Performance: How people management impacts on organisational performance*. CIPD.
19. Babala L., Danilyuk I., Khoma N. (2022). *HRM-системи управління персоналом*. Київ: KHEУ.
20. Ulrich D. (1997). *Human Resource Champions: The Next Agenda for Adding Value and Delivering Results*. Harvard Business School Press.
21. Becker B.E., Huselid M.A. (2001). *The HR Scorecard: Linking People, Strategy, and Performance*. Harvard Business School Press.
22. Kotter J.P. (1996). *Leading Change*. Harvard Business Review Press.
23. Jackson S.E., Schuler R.S., Jiang K. (2014). An Aspirational Framework for Strategic Human Resource Management. *Academy of Management Annals*, 8(1), 1–56.
24. Boxall P., Purcell J. (2003). *Strategy and Human Resource Management*. Palgrave Macmillan.
25. Society for Human Resource Management (SHRM). (2020). *2020 HR Technology Report*. SHRM.org.
26. ISO. (2018). *ISO 30409:2018 – Human resource management — Workforce planning*. ISO.org.
27. Chartered Institute of Personnel and Development (CIPD). (2018). *CIPD Profession Map*. CIPD.co.uk.

28. Hardt D. (2012). *The OAuth 2.0 Authorization Framework*. RFC 6749. <https://tools.ietf.org/html/rfc6749>
29. Advanced Distributed Learning (ADL). (2001). *Sharable Content Object Reference Model (SCORM) 1.2*. <https://adl.gov/>
30. Rustici Software. (2013). *Experience API (xAPI) 1.0.3*. <https://xapi.com/>
31. Hightower K., Burns B., Beda J. (2017). *Kubernetes: Up and Running*. O'Reilly Media.
32. Merkel D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, 239, 2.
33. Marler J.H., Boudreau J.W. (2017). An evidence-based review of HR Analytics. *International Journal of Human Resource Management*, 28(1), 3–26.
34. Rafin S. (2022). *Final 16 October, 22-final correction – M.Engg. Thesis – Sanjida Rafin*. Приклад системної ієрархії та BFF-підходу.
35. Apache Kafka Documentation. (2025). *Kafka: A Distributed Streaming Platform*. <https://kafka.apache.org/>
36. Istio Documentation. (2025). *Istio Service Mesh*. <https://istio.io/>
37. Prometheus Authors. (2025). *Prometheus Monitoring System*. <https://prometheus.io/>
38. Elastic. (2025). *Elasticsearch: The Search Engine*. <https://elastic.co/elasticsearch/>
39. Redis Labs. (2025). *Redis: In-memory Data Structure Store*. <https://redis.io/>
40. Zimmermann, O. (2020). *Event Sourcing in Domain-Driven Design*. Addison-Wesley.
41. Richardson, C. (2018). *Microservices Patterns: With examples in Java*. Manning.
42. ISO. (2019). *ISO 639-1: Codes for the representation of names of languages — Part 1: Alpha-2 code*. ISO.org.
43. Young, G. (2017). *CQRS and Event Sourcing: Architecture Patterns*. O'Reilly Media.

44. Nygard, M. (2017). *Release It!: Design and Deploy Production-Ready Software*. Pragmatic Bookshelf.
45. Doerr, J. (2018). *Measure What Matters: How Google, Bono, and the Gates Foundation Rock the World with OKRs*. Portfolio.
46. Kaplan, R. S., & Norton, D. P. (1992). *The Balanced Scorecard: Measures that Drive Performance*. Harvard Business Review.
47. Levenson, A. (2011). *Using Targeted Analytics to Improve Organizational Performance*. *Journal of Organizational Effectiveness*, 1(1), 55–60.
48. Bracken, D. W., Rose, D. S., & Church, A. H. (2016). *The Evolution and Future of 360° Feedback*. *Industrial and Organizational Psychology*, 9(4), 761–794.
49. Goodhue, D. L., Wixom, B. H., & Watson, H. J. (2002). *Realizing Business Benefits Through CRM: Hitting the Right Target in the Right Way*. *MIS Quarterly Executive*, 1(2), 79–94.
50. Twigg, C. A. (2004). *Quality Indicators for Online Course Development*. EduCause.
51. Pappas, C. (2015). *The Ultimate eLearning Course Development Checklist*. eLearning Industry.
52. Kelly, B. (2019). *Gamification in Enterprise Training*. *Journal of Business Learning*, 12(2), 45–59.
53. Clark, R. C., & Mayer, R. E. (2016). *E-Learning and the Science of Instruction*. Wiley.
54. Siemens, G. (2005). *Connectivism: A Learning Theory for the Digital Age*. *International Journal of Instructional Technology & Distance Learning*
55. Fowler, M. (2021). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley.
56. Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
57. Turnbull, J. (2014). *The Art of Monitoring*. Turnbull Press.
58. Redux Toolkit Documentation. (2022). <https://redux-toolkit.js.org/>
59. Flutter Documentation. (2023). <https://flutter.dev/docs/>

ДОДАТКИ

ДОДАТОК А



ДОДАТОК Б



ДОДАТОК В

```
import 'dart:convert';
import 'package:http/http.dart' as http;

class ApiService {
  final String baseUrl;

  ApiService(this.baseUrl);

  Future<http.Response> post(String endpoint, Map<String, dynamic> body) async {
    final url = Uri.parse('$baseUrl$endpoint');
    final response = await http.post(
      url,
      headers: {'Content-Type': 'application/json'},
      body: jsonEncode(body),
    );
    return response;
  }
}

// ----- File: lib/api/auth_service.dart -----
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:shared_preferences/shared_preferences.dart';

class AuthService {
  final String _tokenKey = 'auth_token';

  /// Верифікація облікового запису
  Future<bool> verifyAccount(String url, Map<String, String> headers, dynamic body) async {
    final response = await http.post(Uri.parse(url), headers: headers, body: body);

    if (response.statusCode == 202) {
      final Map<String, dynamic> responseData = json.decode(response.body);
      if (responseData.containsKey('data')) {
        final String token = responseData['data'];

        // Збереження токєну
        final prefs = await SharedPreferences.getInstance();
        await prefs.setString(_tokenKey, token);

        return true;
      } else {
        // Обробка випадку, коли відповідь не містить токєну
        throw Exception('Token not found in the response');
      }
    } else {
      // Обробка помилок при верифікації
      throw Exception('Failed to verify account: ${response.statusCode}');
    }
  }
}
```

```

    }

    /// Отримання токена зі сховища
    Future<String?> getToken() async {
      final prefs = await SharedPreferences.getInstance();
      return prefs.getString(_tokenKey);
    }

    /// Вихід з аккаунту (видалення токена)
    Future<void> logout() async {
      final prefs = await SharedPreferences.getInstance();
      await prefs.remove(_tokenKey);
    }
  }

  // ----- File: lib/api/verification_service.dart -----
  import 'dart:convert';
  import 'package:http/http.dart' as http;

  class VerificationService {
    final String baseUrl;

    VerificationService(this.baseUrl);

    /// Метод для перевірки електронної пошти та номера телефону
    Future<http.Response> checkEmailAndPhone(String email, String phoneNumber) async {
      final url = Uri.parse('$baseUrl/verification/send_otp');

      // Тіло запиту має містити обидва значення: email і phoneNumber
      final response = await http.post(
        url,
        headers: {'Content-Type': 'application/json'},
        body: jsonEncode({
          'email': email,
          'phone_number': phoneNumber, // Важливо використовувати правильний ключ, як
очікується сервером
        })),
      );

      return response;
    }

    /// Метод для верифікації OTP
    Future<http.Response> verifyOtp(String phoneNumber, String code) async {
      final url = Uri.parse('$baseUrl/verification/verify_otp');
      final response = await http.post(
        url,
        headers: {'Content-Type': 'application/json'},
        body: jsonEncode({

```

```

        'user': {
          'phoneNumber': phoneNumber,
        },
        'code': code,
      })),
    );
    return response;
  }
}

```

// ----- File: lib/geo/full_screen_map.dart -----

```

import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:geolocator/geolocator.dart';

```

```

class FullScreenMap extends StatefulWidget {
  const FullScreenMap({super.key});

```

```

  @override
  _FullScreenMapState createState() => _FullScreenMapState();
}

```

```

class _FullScreenMapState extends State<FullScreenMap> {
  GoogleMapController? mapController; // Зроблено nullable
  LatLng? currentLocation;

```

```

  @override
  void initState() {
    super.initState();
    _determinePosition();
  }

```

```

  void _onMapCreated(GoogleMapController controller) {
    mapController = controller;
    if (currentLocation != null) { // Якщо локація вже доступна, центруємо камеру
      _animateToCurrentLocation();
    }
  }

```

```

  void _animateToCurrentLocation() {
    mapController?.animateCamera(
      CameraUpdate.newCameraPosition(
        CameraPosition(target: currentLocation!, zoom: 15),
      ),
    );
  }

```

```

  Future<void> _determinePosition() async {
    bool serviceEnabled;
    LocationPermission permission;

```

```

serviceEnabled = await Geolocator.isLocationServiceEnabled();
if (!serviceEnabled) {
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(content: Text('Location services are disabled.'))
  );
  return;
}

permission = await Geolocator.checkPermission();
if (permission == LocationPermission.denied) {
  permission = await Geolocator.requestPermission();
  if (permission == LocationPermission.denied) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Location permissions are denied'))
    );
    return;
  }
}

if (permission == LocationPermission.deniedForever) {
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(content: Text('Location permissions are permanently denied, we cannot
request permissions.'))
  );
  return;
}

Position position = await Geolocator.getCurrentPosition();
setState(() {
  currentLocation = LatLng(position.latitude, position.longitude);
  if (mapController != null) {
    _animateToCurrentLocation();
  }
});
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Full Screen Map'),
    ),
    body: currentLocation == null
      ? const Center(child: CircularProgressIndicator())
      : GoogleMap(
        onMapCreated: _onMapCreated,
        initialCameraPosition: CameraPosition(
          target: currentLocation ?? const LatLng(0, 0), // Додаємо fallback локацію
          zoom: 11.0,
        ),
        myLocationEnabled: true,
        myLocationButtonEnabled: true,
      ),
  );
}

```

```

    ),
  );
}
}

```

// ----- File: lib/load_screen.dart -----

```

import 'package:app_mobile1/api/auth_service.dart';
import 'package:app_mobile1/pages/registration_pages/registration_page.dart';
import 'package:app_mobile1/pages/worker/bottom_nav_worker_bar.dart';
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'dart:math' as math;

```

```

class LoadingScreen extends StatefulWidget {
  const LoadingScreen({super.key});

  @override
  _LoadingScreenState createState() => _LoadingScreenState();
}

```

```

class _LoadingScreenState extends State<LoadingScreen> with
SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _opacityAnimation;

```

```

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(seconds: 3),
      vsync: this,
    )..repeat();

    _opacityAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(
      CurvedAnimation(parent: _controller, curve: Curves.easeIn),
    );

```

```

    // Додаємо затримку перед завантаженням даних
    Future.delayed(const Duration(milliseconds: 500), () {
      _loadData();
    });
  }

```

```

Future<void> _loadData() async {
  try {
    final authService = AuthService();
    final token = await authService.getToken(); // Отримання токєну

    if (mounted) {
      if (token != null) { // Перевірка наявності токєну
        await _loadAdditionalData();
      }
    }
  } catch (e) {
    // Обробка помилок
  }
}

```

```

        Navigator.of(context).pushReplacement(
          MaterialPageRoute(
            builder: (context) => const BottomNavWorkerBar(),
          ),
        );
      } else {
        Navigator.of(context).pushReplacement(
          MaterialPageRoute(
            builder: (context) => const RegistrationPage(),
          ),
        );
      }
    }
  } catch (e) {
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Failed to load data: $e')),
      );
    }
  }
}

```

```

Future<void> _loadAdditionalData() async {
  // Виконуємо додаткові завантаження
  await Future.wait([
    _loadUserData(),
    _loadSettings(),
    _loadAppHeader(),
    _loadCountryDropdown(),
    _loadCustomTextField(),
    _loadEmailWidget(),
    _loadSettingsPage(),
  ]);
}

```

```

Future<void> _loadUserData() async {
  await Future.delayed(const Duration(seconds: 2));
}

```

```

Future<void> _loadSettings() async {
  await Future.delayed(const Duration(seconds: 1));
}

```

```

Future<void> _loadAppHeader() async {
  await Future.delayed(const Duration(seconds: 1));
}

```

```

Future<void> _loadCountryDropdown() async {
  await Future.delayed(const Duration(seconds: 2));
}

```

```

Future<void> _loadCustomTextField() async {

```

```
    await Future.delayed(const Duration(seconds: 1));  
  }
```

```
Future<void> _loadEmailWidget() async {  
  await Future.delayed(const Duration(seconds: 1));  
}
```

```
Future<void> _loadSettingsPage() async {  
  await Future.delayed(const Duration(seconds: 1));  
}
```

```
@override  
void dispose() {  
  _controller.dispose();  
  super.dispose();  
}
```

```
@override  
Widget build(BuildContext context) {  
  return Scaffold(  
    body: Center(  
      child: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
          Stack(  
            alignment: Alignment.center,  
            children: [  
              SvgPicture.asset(  
                'assets/icons/logo_Test.svg',  
                width: 200,  
                height: 200,  
              ),  
              Positioned(  
                child: CustomPaint(  
                  size: const Size(200, 200),  
                  painter: LinePainter(),  
                ),  
              ),  
              Positioned(  
                child: AnimatedBuilder(  
                  animation: _controller,  
                  builder: (context, child) {  
                    return Transform.rotate(  
                      angle: _controller.value * 2.0 * math.pi,  
                      child: CustomPaint(  
                        size: const Size(200, 200),  
                        painter: RotatingLinePainter(),  
                      ),  
                    );  
                  },  
                ),  
              ),  
            ],  
          ),  
        ],  
      ),  
    ),  
  );  
}
```

```

        ],
    ),
    const SizedBox(height: 20),
    FadeTransition(
      opacity: _opacityAnimation,
      child: const Text(
        'Support Security',
        style: TextStyle(
          fontSize: 24,
          fontWeight: FontWeight.bold,
          color: Colors.black,
        ),
      ),
    ),
  ],
),
],
),
);
}
}

```

```

class LinePainter extends CustomPainter {
  @override
  void paint(Canvas canvas, Size size) {
    final paint = Paint()
      ..color = Colors.white
      ..strokeWidth = 6.0
      ..style = PaintingStyle.stroke;

    final center = Offset(size.width / 2, size.height / 2);

    canvas.drawLine(
      center,
      Offset(center.dx, center.dy - size.height / 2),
      paint,
    );

    final circlePaint = Paint()
      ..color = Colors.white
      ..style = PaintingStyle.fill;

    canvas.drawCircle(center, 4.0, circlePaint);
  }

  @override
  bool shouldRepaint(CustomPainter oldDelegate) {
    return false;
  }
}

```

```

class RotatingLinePainter extends CustomPainter {
  @override

```

```

void paint(Canvas canvas, Size size) {
  final paint = Paint()
    ..color = Colors.white
    ..strokeWidth = 6.0
    ..style = PaintingStyle.stroke;

  final center = Offset(size.width / 2, size.height / 2);

  canvas.drawLine(
    center,
    Offset(center.dx, center.dy - size.height / 2 + 20),
    paint,
  );

  canvas.drawLine(
    center,
    Offset(center.dx, center.dy - size.height / 2 + 10),
    paint..color = Colors.white,
  );

  final circlePaint = Paint()
    ..color = Colors.white
    ..style = PaintingStyle.fill;

  canvas.drawCircle(center, 4.0, circlePaint);
}

@override
bool shouldRepaint(CustomPainter oldDelegate) {
  return false;
}
}

// ----- File: lib/main.dart -----
import 'package:app_mobile1/api/auth_servise.dart';
import 'package:app_mobile1/load_screen.dart';
import 'package:app_mobile1/models/user_profile_provider.dart';
import 'package:app_mobile1/pages/notification_pages/notification_screen.dart';
import 'package:app_mobile1/pages/partner/bottom_navigation_bar.dart';
import 'package:app_mobile1/pages/worker/bottom_nav_worker_bar.dart';
import 'package:app_mobile1/pages/registration_pages/registration_page.dart';
import 'package:flutter_dotenv/flutter_dotenv.dart';
import 'package:flutter_stripe/flutter_stripe.dart';
import 'package:provider/provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_secure_storage/flutter_secure_storage.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await dotenv.load(fileName: ".env");
  Stripe.publishableKey = dotenv.env["STRIPE_PUBLISH_KEY"]!;

```

```

await Stripe.instance.applySettings();

runApp(
  ChangeNotifierProvider(
    create: (context) => UserProfileProvider(),
    child: const MyApp(),
  ),
);
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: FutureBuilder<String?>(
        future: _getToken(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const LoadingScreen(); // Показуємо екран завантаження поки перевіряється
            токен
          } else if (snapshot.hasData && snapshot.data != null) {
            return const BottomNavigationBarScreen(); // Якщо токен є, переходимо на головний
            екран
          } else {
            return const RegistrationPage(); // Якщо токена немає, переходимо на екран
            реєстрації
          }
        },
      ),
      routes: {
        '/notification_screen': (context) => const NotificationsScreen(),
      },
    );
  }

  Future<String?> _getToken() async {
    const FlutterSecureStorage storage = FlutterSecureStorage();
    final token = await storage.read(key: 'jwt_token');
    print('Token retrieved: $token');
    return token;
  }
}

// ----- File: lib/models/confirmation.dart -----
class Confirmation {
  final String title;
  final String details;
  final String date;
}

```

```

final String state; // Додавання поля стану

Confirmation({required this.title, required this.details, required this.date, required this.state});
}

final List<Confirmation> confirmationList = [
  Confirmation(
    title: "Confirmed",
    details: "23.03.23",
    date: "Confirmed",
    state: "Confirmed",
  ),

  Confirmation(
    title: "Confirmed",
    details: "Details for Confirmation 2",
    date: "24.03.23",
    state: "Confirmed",
  ),
  Confirmation(
    title: "Confirmed",
    details: "Details for Confirmation 2",
    date: "24.03.23",
    state: "Confirmed",
  ),
  Confirmation(
    title: "Checking in progress",
    details: "Details for Confirmation 2",
    date: "24.03.23",
    state: "Checking in progress",
  ),
  Confirmation(
    title: "Ignored",
    details: "Details for Confirmation 2",
    date: "24.03.23",
    state: "Ignored",
  ),
  Confirmation(
    title: "Checking in progress",
    details: "Details for Confirmation 2",
    date: "24.03.23",
    state: "Checking in progress",
  ),
  Confirmation(
    title: "Not confirmed",
    details: "Details for Confirmation 2",
    date: "24.03.23",
    state: "Not confirmed",
  ),
  Confirmation(
    title: "Not confirmed",

```

```

        details: "Details for Confirmation 2",
        date: "24.03.23",
        state: "Not confirmed",
    ),
    Confirmation(
        title: "Ignored",
        details: "Details for Confirmation 2",
        date: "24.03.23",
        state: "Ignored",
    ),
    Confirmation(
        title: "Ignored",
        details: "Details for Confirmation 2",
        date: "24.03.23",
        state: "Ignored",
    ),
];

// ----- File: lib/models/employee.dart -----
// employee.dart
class Employee {
    String name;
    String email;
    String type;

    Employee({required this.name, required this.email, required this.type});
}

```

```

// ----- File: lib/models/user_data.dart -----
// lib/models/user_data.dart

```

```

class UserData {
    String? firstName;
    String? lastName;
    String? phoneNumber;
    String? email;
    String? password;
    String? country;
    String? companyName;
    String role = 'partner';

    Map<String, dynamic> toJson() {
        return {
            'first_name': firstName ?? "",
            'last_name': lastName ?? "",
            'phone_number': phoneNumber ?? "",
            'email': email ?? "",
            'password': password ?? "",
            'country': country ?? "",
            'company_name': companyName ?? "",
            'role': role,

```

```
};  
}  
}
```

```
// ----- File: lib/models/user_profile_provider.dart -----  
import 'package:flutter/material.dart';
```

```
class UserProfile {  
  String firstName = 'Dmytro';  
  String lastName = 'Santarskyi';  
  String country = 'PL Poland';  
  String companyAddress = "";  
  String nip = "";  
  String typeOfWork = "";  
  String placeOfWork = "";  
  String phoneNumber = '+48432324432';  
  String phoneNumberCode = '432324432';  
  String email = 'dim4ksab@gmail.com';  
  String numIntermediaries = "";  
  String numManagers = "";  
  String numForemans = "";  
  String numWorkers = "";  
  String companyName = "";  
  String ownManager = 'Vasya1';  
  String confirmedHours = '8.0';  
  String nonConfirmedHours = '8.0';  
  String requiredWorkedHours = '5';  
  String confirmedIncome = '1200.0';  
  String nonConfirmedIncome = '400.0';  
  String hourlyRate = '12';  
  String speciality = "Building the house1";  
  String role = "Manager";  
  List<String> imagePaths = ["", "", "", ""]; // Додано для зберігання шляхів до зображень  
}
```

```
class UserProfileProvider with ChangeNotifier {  
  UserProfile _userProfile = UserProfile();  
  
  UserProfile get userProfile => _userProfile;  
  
  void updateUserProfile(UserProfile updatedProfile) {  
    _userProfile = updatedProfile;  
    notifyListeners();  
  }  
  
  void updateImage(int index, String path) {  
    if (index >= 0 && index < _userProfile.imagePaths.length) {  
      _userProfile.imagePaths[index] = path;  
      notifyListeners();  
    }  
  }  
}
```

```
}
```

```
// ----- File: lib/pages/forgot_pages/forgot_page.dart -----  
import 'package:app_mobile1/widgets/header/app_header.dart';  
import 'package:flutter/material.dart';  
import 'forgot_page_email.dart';  
import 'forgot_page_phone.dart';
```

```
class ForgotPage extends StatelessWidget {  
  const ForgotPage({super.key});
```

```
  @override
```

```
  Widget build(BuildContext context) {  
    return Scaffold(  
      body: SafeArea(  
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.center,  
          children: [  
            const AppHeader(),  
            const SizedBox(height: 127),  
            _buildForgotTitle(),  
            const SizedBox(height: 47),  
            _buildButtons(context),  
            const Spacer(),  
          ],  
        ),  
      ),  
    );  
  }  
}
```

```
Widget _buildForgotTitle() {  
  return const Column(  
    children: [  
      Text(  
        "Forgot your password?",  
        style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),  
      ),  
      SizedBox(height: 10),  
      Text(  
        "Which password reset method do you prefer?",  
        style: TextStyle(fontSize: 14, color: Colors.grey),  
      ),  
    ],  
  );  
}
```

```
Widget _buildButtons(BuildContext context) {  
  return Column(  
    children: [  
      SizedBox(  
        width: double.infinity,
```

```

child: Padding(
  padding: const EdgeInsets.symmetric(horizontal: 33),
  child: ElevatedButton(
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => const ForgotPagePhone()),
        );
      // Дії при натисканні на "Email Verification"
    },
    style: ElevatedButton.styleFrom(
      backgroundColor: const Color(0xFF4869D9),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(7),
      ),
    ),
    child: const Text(
      "By phone number",
      style: TextStyle(
        color: Colors.white,
        fontSize: 18,
      ),
    ),
  ),
),
const SizedBox(height: 10),
SizedBox(
  width: double.infinity,
  child: Padding(
    padding: const EdgeInsets.symmetric(horizontal: 33),
    child: ElevatedButton(
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => const ForgotPageEmail()),
          );
        // Дії при натисканні на "SMS Verification"
      },
      style: ElevatedButton.styleFrom(
        backgroundColor: const Color(0xFF4869D9),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(7),
        ),
      ),
      child: const Text(
        "By email",
        style: TextStyle(
          color: Colors.white,
          fontSize: 18,
        ),
      ),
    ),
  ),
),

```

```

    ),
  ),
),
),
),
],
);
}
}

```

```

// ----- File: lib/pages/forgot_pages/forgot_page_email.dart -----
import 'package:app_mobile1/widgets/verification_UI.dart.dart';
import 'package:flutter/material.dart';
import '../registration_pages/new_account_page_pw.dart';

```

```

class ForgotPageEmail extends StatefulWidget {
  const ForgotPageEmail({super.key});

  @override
  State<ForgotPageEmail> createState() => _ForgotPageEmailState();
}

```

```

class _ForgotPageEmailState extends State<ForgotPageEmail> {
  String enteredPin = "";

  @override
  Widget build(BuildContext context) {
    return VerificationPagesConstructor(
      asset: "assets/icons/email.svg",
      textTitle: "Password Reset",
      text:
        "Enter the six code we send to your\ne-mail address to reset your password",
      onCompleted: (pin) {
        // Дії після введення пін-коду
      },
      onChanged: (value) {
        setState() {
          enteredPin = value;
        });
      },
      buttonFunction: enteredPin.length == 6
        ? () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => const NewAccountPagePw(),
              );
          }
        : null,
    );
  }
}

```

```

}

// ----- File: lib/pages/forgot_pages/forgot_page_phone.dart -----
import 'package:app_mobile1/widgets/verification_UI.dart.dart';
import 'package:flutter/material.dart';
import '../registration_pages/new_account_page_pw.dart';

class ForgotPagePhone extends StatefulWidget {
  const ForgotPagePhone({super.key});

  @override
  State<ForgotPagePhone> createState() => _ForgotPagePhoneState();
}

class _ForgotPagePhoneState extends State<ForgotPagePhone> {
  String enteredPin = "";

  @override
  Widget build(BuildContext context) {
    return VerificationPagesConstructor(
      asset: "assets/icons/phone.svg",
      textTitle: "Password Reset",
      text:
        "Enter the six code we send to your\phone number to reset your password",
      onCompleted: (pin) {
        // Дії після введення пін-коду
      },
      onChanged: (value) {
        setState() {
          enteredPin = value;
        });
      },
      buttonFunction: enteredPin.length == 6
        ? () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => const NewAccountPagePw(),
              );
          }
        : null,
    );
  }
}

```

```

// ----- File: lib/pages/notification_pages/notification_screen.dart -----
import 'package:app_mobile1/widgets/header/icons/clock_icon_button.dart';
import 'package:flutter/material.dart';
import 'package:flutter_svg/svg.dart';

```

```

ValueNotifier<int> unreadNotificationsCount = ValueNotifier<int>(0);

```

```

class NotificationItem {
  final String message;
  bool isRead;

  NotificationItem({required this.message, this.isRead = false});
}

class NotificationsScreen extends StatefulWidget {
  const NotificationsScreen({super.key});

  @override
  State<NotificationsScreen> createState() => _NotificationsScreenState();
}

class _NotificationsScreenState extends State<NotificationsScreen> {
  final List<NotificationItem> notifications = [
    NotificationItem(message: "Нове повідомлення", isRead: false),
    NotificationItem(message: "Ще одне повідомлення", isRead: true),
    NotificationItem(message: "Нове повідомлення", isRead: false),
    NotificationItem(message: "Ще одне повідомлення", isRead: true),
    NotificationItem(message: "Нове повідомлення", isRead: false),
    NotificationItem(message: "Ще одне повідомлення", isRead: true),
  ];

  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addPostFrameCallback((_) {
      unreadNotificationsCount.value =
        notifications.where((n) => !n.isRead).length;
    });
  }

  void toggleReadStatus(int index) {
    setState() {
      notifications[index].isRead = !notifications[index].isRead;
    };
    WidgetsBinding.instance.addPostFrameCallback((_) {
      unreadNotificationsCount.value =
        notifications.where((n) => !n.isRead).length;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Padding(
        padding: const EdgeInsets.all(8.0),
        child: SafeArea(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,

```

```

children: [
  Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      const SizedBox(width: 15, height: 93),
      SvgPicture.asset('assets/icons/L_D_26.svg'),
      const SizedBox(width: 12),
      const Text(
        "Support Security",
        style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
      ),
      const Spacer(),
      const ClockIconButton(),
    ],
  ),
  const SizedBox(height: 20),
  _buildBackButton(context),
  const SizedBox(height: 36),
  _buildNotificationsList(),
],
),
),
),
);
}

```

```

Widget _buildBackButton(BuildContext context) {
  return Row(
    children: [
      IconButton(
        icon: const Icon(Icons.arrow_back),
        onPressed: () {
          Navigator.pop(context);
        },
      ),
      const SizedBox(width: 12),
      const Text(
        "Notifications",
        style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
      ),
    ],
  );
}

```

```

Widget _buildNotificationsList() {
  return Expanded(
    child: ListView.builder(
      padding: const EdgeInsets.symmetric(horizontal: 8.0),
      itemCount: notifications.length,
      itemBuilder: (context, index) {
        final notification = notifications[index];
        return Padding(

```

```

padding: const EdgeInsets.all(4.0),
child: Card(
  elevation: 0,
  shape: RoundedRectangleBorder(
    side: const BorderSide(color: Color(0xFFD0D1D2)),
    borderRadius: BorderRadius.circular(3),
  ),
  child: ListTile(
    leading: notification.isRead
      ? null
      : const Icon(Icons.circle, color: Colors.blue, size: 10.0),
    title: Text(
      notification.message,
      style: TextStyle(
        color: notification.isRead
          ? const Color(0xFF8D8D8D)
          : Colors.black),
    ),
    trailing: IconButton(
      icon: Icon(notification.isRead
        ? Icons.mark_email_read
        : Icons.mark_email_unread),
      onPressed: () => toggleReadStatus(index),
    ),
    onTap: () {
      if (!notification.isRead) {
        toggleReadStatus(index);
      }
    },
  ),
),
);
},
),
);
}
}
}

```

// ----- File: lib/pages/partner/bottom_navigation_bar.dart -----

```

import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'profile_page_partner.dart';
import 'employees/list_worker_page_partner.dart';
import 'geo_check_page_partner.dart';

```

```

class BottomNavigationBarScreen extends StatefulWidget {
  const BottomNavigationBarScreen({super.key});

```

```

  @override
  State<BottomNavigationBarScreen> createState() => _BottomNavigationBarScreenState();
}

```

```

class _BottomNavigationBarScreenState extends State<BottomNavigationBarScreen> {
  int _currentIndex = 0;

  final List<Widget> _screens = [
    const ProfilePagePartner(),
    const ListWorkerPagePartner(),
    const GeoCheckPagePartner(),
  ];

  BottomNavigationBarItem createNavItem(String assetName, String label) {
    return BottomNavigationBarItem(
      icon: SvgPicture.asset(
        'assets/icons/$assetName.svg',
        color: Colors.grey,
      ),
      activeIcon: SvgPicture.asset(
        'assets/icons/$assetName.svg',
        color: Colors.blue,
      ),
      label: label,
    );
  }

  List<BottomNavigationBarItem> getNavBarItems() {
    return [
      createNavItem('profile1', 'Profile'),
      createNavItem('Group', 'Group'),
      createNavItem('check', 'Check'),
    ];
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: IndexedStack(
        index: _currentIndex,
        children: _screens,
      ),
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _currentIndex,
        onTap: (index) {
          setState() {
            _currentIndex = index;
          });
        },
        items: getNavBarItems(),
        selectedItemColor: Colors.blue,
        unselectedItemColor: Colors.grey,
      ),
    );
  }
}

```

```
}
```

```
// ----- File: lib/pages/partner/check_page_partner.dart -----  
import 'package:app_mobile1/pages/partner/confirmation_history_screen.dart';  
import 'package:app_mobile1/pages/payment_page.dart';  
import 'package:app_mobile1/widgets/header/icons/clock_icon_button.dart';  
import 'package:flutter/material.dart';  
import 'package:app_mobile1/models/employee.dart';  
import 'package:google_maps_flutter/google_maps_flutter.dart';  
import 'package:app_mobile1/widgets/header/app_header.dart';  
import 'package:app_mobile1/widgets/forms/button_next.dart';
```

```
class CheckPagePartner extends StatelessWidget {  
  final Employee employee;  
  
  const CheckPagePartner({super.key, required this.employee});
```

```
  final bool payed = true;
```

```
  @override
```

```
  Widget build(BuildContext context) {  
    return Scaffold(  
      body: SafeArea(  
        child: Padding(  
          padding: const EdgeInsets.only(right: 25),  
          child: ListView(  
            children: [  
              const AppHeader(  
                clockIconButton: ClockIconButton(),  
              ),  
              const SizedBox(height: 45),  
              Padding(  
                padding: const EdgeInsets.only(right: 8, left: 33),  
                child: Column(  
                  children: [  
                    _buildTitleAndBackButton(context),  
                    const SizedBox(height: 20),  
                    if (payed) ...[  
                      _buildEmployeeInformation(),  
                      const SizedBox(height: 10),  
                      _buildCenteredIcon(context),  
                      const SizedBox(height: 10),  
                      _buildConfirmationSection(context),  
                      _buildViewHistoryButton(context),  
                    ] else  
                    Stack(  
                      children: [  
                        IgnorePointer(  
                          ignoring: true,  
                          child: Opacity(  
                            opacity: 0.2,
```

```

        child: Column(
          children: [
            _buildEmployeeInformation(),
            const SizedBox(height: 10),
            _buildCenteredIcon(context),
            const SizedBox(height: 10),
            _buildConfirmationSection(context),
            _buildViewHistoryButton(context),
          ],
        ),
      ),
    ),
    Positioned(
      top: 135,
      left: 128,
      child: Image.asset(
        'assets/icons/block.png',
        width: 70,
        height: 70,
        fit: BoxFit.contain,
      ),
    ),
  ],
),
if (!payed)
  ButtonNext(
    buttonFunction: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => const PaymentPage(),
        );
    },
    name: "Unlock by paying",
  ),
],
),
),
],
),
),
);
}

```

```

Widget _buildTitleAndBackButton(BuildContext context) {
  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      const Expanded(
        child: Text(
          "Availability at workplace",

```

```

        style: TextStyle(
          fontWeight: FontWeight.bold,
          fontSize: 22,
          color: Colors.black,
        ),
        textAlign: TextAlign.center,
      ),
    ),
    Container(
      width: 46,
      height: 46,
      decoration: BoxDecoration(
        color: const Color.fromRGBO(223, 223, 223, 0.5),
        borderRadius: BorderRadius.circular(7),
      ),
      child: IconButton(
        onPressed: () => Navigator.pop(context),
        icon: const Icon(Icons.close, color: Colors.black),
        tooltip: "Close",
      ),
    ),
  ],
);
}

```

```

Widget _buildCenteredIcon(context) {
  return Center(
    child: Column(
      children: [
        const SizedBox(height: 18),
        ButtonNext(
          buttonFunction: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const PaymentPage()),
            );
          },
          name: 'Confirm availability',
        ),
      ],
    ),
  );
}

```

```

Widget _buildEmployeeInformation() {
  return Align(
    alignment: Alignment.topLeft,
    child: Text.rich(
      TextSpan(
        style: const TextStyle(fontSize: 16),
        children: [
          TextSpan(text: "${employee.name}\n${employee.email}"),

```

```

    ],
  ),
),
);
}

```

```

Widget _buildConfirmationSection(BuildContext context) {
  return Container(
    padding: const EdgeInsets.all(12),
    margin: const EdgeInsets.all(12),
    decoration: BoxDecoration(
      border: Border.all(color: Colors.grey),
      borderRadius: BorderRadius.circular(5),
    ),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        const Text(
          "Confirmed",
          style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
        ),
        const SizedBox(height: 5),
        const Text(
          "23.03.23 - 00493204230423482",
          style: TextStyle(fontSize: 14),
        ),
        const SizedBox(height: 5),
        _buildMapSection(context),
      ],
    ),
  );
}

```

```

Widget _buildSquarePhotoContainer() {
  return Container(
    width: 93,
    height: 93,
    decoration: BoxDecoration(
      color: const Color.fromRGBO(223, 223, 223, 0.5),
      borderRadius: BorderRadius.circular(5),
      border: Border.all(color: Colors.grey),
    ),
  );
}

```

```

Widget _buildMapSection(BuildContext context) {
  return Container(
    margin: const EdgeInsets.symmetric(vertical: 10),
    child: Row(
      children: [
        Expanded(
          flex: 1,

```

```

child: SizedBox(
  height: 93,
  child: GoogleMap(
    onMapCreated: (GoogleMapController controller) {},
    onTap: (LatLng position) {},
    initialCameraPosition: const CameraPosition(
      target: LatLng(40.7128, -74.0060),
      zoom: 14,
    ),
    myLocationEnabled: true,
  ),
),
),
const SizedBox(width: 10),
Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    _buildSquarePhotoContainer(),
  ],
),
],
),
);
}

```

```

Widget _buildViewHistoryButton(BuildContext context) {
  // Додано параметр context
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 10),
    child: Align(
      alignment: Alignment.center,
      child: TextButton(
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => const ConfirmationsHistoryScreen(),
            ),
          );
        },
        child: const Text(
          "View confirmations history",
          style: TextStyle(
            fontSize: 16,
            color: Colors.blue,
            decoration: TextDecoration.underline,
          ),
        ),
      ),
    ),
  );
}

```

```
// ----- File: lib/pages/partner/confirmation_history_screen.dart -----
import 'package:app_mobile1/widgets/header/app_header.dart';
import 'package:app_mobile1/widgets/header/icons/clock_icon_button.dart';
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:table_calendar/table_calendar.dart';
import 'package:flutter_svg/flutter_svg.dart';

class Confirmation {
  final String status;
  final String date;
  final String phoneNumber;
  final String checkedBy;
  final String email;
  final LatLng location;

  Confirmation({
    required this.status,
    required this.date,
    required this.phoneNumber,
    required this.checkedBy,
    required this.email,
    required this.location,
  });
}

class ConfirmationsHistoryScreen extends StatefulWidget {
  const ConfirmationsHistoryScreen({super.key});

  @override
  _ConfirmationsHistoryScreenState createState() => _ConfirmationsHistoryScreenState();
}

class _ConfirmationsHistoryScreenState extends State<ConfirmationsHistoryScreen> {
  final List<Confirmation> confirmations = [
    Confirmation(
      status: 'Confirmed',
      date: '23.03.23',
      phoneNumber: '00493204230423482',
      checkedBy: 'Jack Brown',
      email: 'jackbrown@gmail.com',
      location: const LatLng(53.7128, -45.0060),
    ),
    Confirmation(
      status: 'Confirmed',
      date: '23.03.23',
      phoneNumber: '00493204230423482',
      checkedBy: 'Jane Doe',
      email: 'janedoe@gmail.com',
      location: const LatLng(40.7128, -74.0060),
    ),
  ];
}
```

```

    ),
    Confirmation(
      status: 'Not confirmed',
      date: '23.03.23',
      phoneNumber: '00493204230423482',
      checkedBy: 'John Smith',
      email: 'johnsmith@gmail.com',
      location: const LatLng(45.7128, -73.0060),
    ),
  ];

```

```

DateTime? _selectedDate;

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SafeArea(
      child: Column(
        children: [
          const AppHeader(clockIconButton: ClockIconButton()),
          const SizedBox(height: 60),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 10),
            child: Row(
              children: [
                IconButton(
                  icon: const Icon(Icons.arrow_back),
                  onPressed: () {
                    Navigator.pop(context);
                  },
                ),

```



Звіт подібності

метадані

Назва організації

Kyiv National Economic University named after Vadym Hetman KNEU

Заголовок

Розробка HRM-системи для оптимізації управління персоналом і внутрішніх процесів компанії

Автор

Науковий керівник / Експерт

Селіверстов Петро Сергійович Устенко С.В.

Підрозділ

кафедра інформаційних систем в економіці

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



25

Довжина фрази для коефіцієнта подібності 2.



14605

Кількість слів



114593

Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		31
Інтервали		1
Мікропробіли		28
Білі знаки		0
Парафрази (SmartMarks)		13

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз		Колір тексту
ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/bitstreams/4767ad56-71cd-48b8-9505-65f5f83e7c59/download	43 0.29 %
2	https://ir.kneu.edu.ua/bitstreams/4767ad56-71cd-48b8-9505-65f5f83e7c59/download	30 0.21 %
3	https://ir.kneu.edu.ua/bitstreams/4767ad56-71cd-48b8-9505-65f5f83e7c59/download	16 0.11 %
4	https://ir.kneu.edu.ua/bitstreams/4767ad56-71cd-48b8-9505-65f5f83e7c59/download	13 0.09 %
5	https://ir.kneu.edu.ua/bitstreams/4767ad56-71cd-48b8-9505-65f5f83e7c59/download	13 0.09 %

6	https://ir.kneu.edu.ua/bitstreams/4767ad56-71cd-48b8-9505-65f5f83e7c59/download	11 0.08 %
7	РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ З СИСТЕМОЮ ЗАХИСТУ ТРАНЗАКЦІЙ З ПРОДАЖУ ЦИФРОВИХ ІНТЕГРАЛЬНИХ МІКРОСХЕМ 5/30/2024 International University of Economics and Humanities named after Academician Stepan Demianchuk (International University of Economics and Humanities named after Academician Stepan Demianchuk)	11 0.08 %
8	https://www.journalsofznu.zp.ua/index.php/comp-science/article/download/4153/3961/	10 0.07 %
9	https://ir.kneu.edu.ua/bitstreams/4767ad56-71cd-48b8-9505-65f5f83e7c59/download	9 0.06 %
10	https://ir.kneu.edu.ua/bitstreams/4767ad56-71cd-48b8-9505-65f5f83e7c59/download	7 0.05 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
------------------	-----------	--

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
------------------	-----------	--

з програми обміну базами даних (0.08 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ З СИСТЕМОЮ ЗАХИСТУ ТРАНЗАКЦІЙ З ПРОДАЖУ ЦИФРОВИХ ІНТЕГРАЛЬНИХ МІКРОСХЕМ 5/30/2024 International University of Economics and Humanities named after Academician Stepan Demianchuk (International University of Economics and Humanities named after Academician Stepan Demianchuk)	11 (1) 0.08 %

з Інтернету (1.49 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/bitstreams/4767ad56-71cd-48b8-9505-65f5f83e7c59/download	189 (15) 1.29 %
2	http://www.economy.nayka.com.ua/pdf/2_2019/63.pdf	13 (2) 0.09 %
3	https://www.journalsofznu.zp.ua/index.php/comp-science/article/download/4153/3961/	10 (1) 0.07 %
4	https://ir.kneu.edu.ua/server/api/core/bitstreams/075b0a45-dbeb-4496-89bc-4e6df1093925/content	5 (1) 0.03 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)
------------------	-------	---------------------------------------

АНАЛІЗ ПЕРЕВАГ ВИКОРИСТАННЯ HRM-СИСТЕМ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ УПРАВЛІННЯ ПЕРСОНАЛОМ

Селіверстов Петро Сергійович, Вознюк Я. Ю. аспірант кафедри іс

здобувач вищої освіти інституту інформаційних технологій в економіці

*Київський національний економічний
університет імені Вадима Гетьмана*

seliverstov.vv1401s@gmail.com

У сучасних умовах стрімких змін у бізнес-середовищі управління персоналом набуває все більшої ваги як стратегічний елемент успіху організацій. Важливим інструментом для підвищення ефективності управління людськими ресурсами є HRM-системи (Human Resource Management Systems). Їх використання дозволяє автоматизувати процеси управління персоналом, зменшити адміністративні витрати та підвищити якість прийняття управлінських рішень.

Історично система управління персоналом еволюціонувала від простого кадрового обліку до комплексного стратегічного управління людськими ресурсами. Зі зростанням складності бізнес-процесів та кількості даних про співробітників виникла потреба у спеціалізованих ІТ-рішеннях. З появою HRM-систем у 90-х роках XX століття організації отримали можливість ефективніше координувати всі етапи роботи з персоналом — від рекрутингу до звільнення.

HRM-системи інтегрують широкий спектр функцій, включаючи:

- рекрутинг і адаптацію працівників;
- управління продуктивністю;
- планування кар'єри та розвитку персоналу;
- оцінку компетенцій;
- управління компенсаціями та пільгами;
- облік часу і присутності на робочому місці;
- аналіз та прогнозування кадрових потреб.

Переваги використання HRM-систем для підприємств:

1. **Оптимізація кадрових процесів.** HRM-системи дозволяють автоматизувати рутинні операції, такі як облік робочого часу, складання графіків, обробка заявок на відпустку тощо. Це суттєво зменшує навантаження на HR-відділ і дозволяє фокусуватися на стратегічних задачах.
2. **Підвищення прозорості процесів.** Завдяки централізованому зберігання даних про співробітників керівництво має можливість швидко отримувати актуальну інформацію для прийняття обґрунтованих рішень.

3. **Аналітика та прогнозування.** Сучасні HRM-системи містять аналітичні модулі, які допомагають виявляти тренди, аналізувати текучість кадрів, визначати рівень залученості персоналу та прогнозувати кадрові потреби.
4. **Підвищення залученості співробітників.** Через портали самообслуговування працівники отримують доступ до своїх даних, можуть подавати заявки, реєструватися на навчання або оцінювати своє самопочуття на роботі, що підвищує їхню мотивацію і відчуття залученості.
5. **Покращення взаємодії між підрозділами.** HRM-системи часто мають інтеграцію з іншими корпоративними системами (ERP, CRM), що дозволяє забезпечити узгодженість дій різних служб підприємства.

Водночас впровадження HRM-систем супроводжується певними викликами:

- потреба у значних початкових інвестиціях;
- необхідність навчання персоналу;
- ризики, пов'язані із захистом персональних даних;
- потреба в адаптації HR-процесів під стандарти нової системи.

Тим не менш, довгострокові переваги значно переважають потенційні труднощі. Правильний вибір і налаштування HRM-системи сприяє створенню гнучкого, прозорого й орієнтованого на розвиток середовища управління персоналом.

Майбутнє HRM-систем визначатиметься подальшим впровадженням штучного інтелекту, машинного навчання та аналітики великих даних. Вони дозволять ще краще прогнозувати потреби в персоналі, індивідуалізувати плани розвитку співробітників і підвищувати ефективність процесів залучення та утримання талантів.

Отже, використання HRM-систем є важливою умовою підвищення конкурентоспроможності підприємств у сучасному бізнес-середовищі.

Список використаних джерел:

1. «40 HR technology statistics every HR team should know in 2024» (2024), Shortlister, [Online], available at: www.myshortlister.com/insights/hr-technology-statistics
2. «Essential Ai In Talent Acquisition Statistics In 2024» (2024), ZipDo, [Online], available at: www.zipdo.co/statistics/ai-in-talent-acquisition/
3. «Gartner Identifies Top Four HR Investment Trends for 2023» (2023), Gartner, [Online], available at: www.gartner.com/en/newsroom/press-releases/03-06-2023-gartner-identifies-top-four-hr-investment-trends-for-2023

4. Marr, B. (2024), «The Eight Biggest HR Trends In 2024», Forbes, [Online], available at: www.forbes.com/sites/bernardmarr/2023/12/18/the-eight-biggest-hr-trends-in-2024/?sh=24eada29197c
5. McKinsey & Company (2022), «The state of AI in 2022 – and a half decade in review», [Online], available at: www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2022-and-a-half-decade-in-review
6. Stanciu, Th. (2024), «HR Trends to Leverage in Your Strategy», Mirro, [Online], available at: www.mirro.io/blog/hr-trends-2024Третяк О.П. Сучасні персонал-технології у системі управління персоналом на підприємстві / О.П. Третяк // Науковий вісник НЛТУ України. – 2014 – № 24 (4). – С. 389–396 [Електронний ресурс]. – Режим доступу : http://nbuv.gov.ua/UJRN/nvnltu_2014_24.4_66
7. Федик М.В. Креативна зайнятість в умовах цифрових трансформацій / М.В. Федик // Інвестиції: практика та досвід. – 2022. – № 1. – С. 48–54 [Електронний ресурс]. – Режим доступу : http://www.investplan.com.ua/pdf/1_2022/9.pdf.

Науковий керівник: Устенко С. В., док. екон. наук