МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Навчально-науковий інститут інформаційних технологій та бізнесу
Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

на тему: **«Розробка CRM системи для малого та середнього бізнесу у сфері підбору та роботи з персоналом»**

Виконав: студент 4 курсу, групи КН-42
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Паша Олексій Дмитрович
Керівник: *викладач кафедри інформаційних технологій та аналітики даних*
Ляховчук Сергій Васильович
Рецензент: *ФОП в галузі інформаційних технологій, викладач кафедри менеджменту та маркетингу НаУОА*
Шпак Андрій Григорович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

Завідувач кафедри інформаційних технологій та аналітики даних _____
(проф., д.е.н. Кривицька О.Р.)
Протокол № 10 від 28 травня 2025 р.

Острог, 2025

АНОТАЦІЯ

кваліфікаційної роботи

на здобуття освітнього ступеня бакалавра

Тема: *Розробка CRM системи для малого та середнього бізнесу у сфері підбору та роботи з персоналом*

Автор: *Паша Олексій Дмитрович*

Науковий керівник: *Ляховчук Сергій Васильович*

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: *54 с., 12 рис., 0 табл., 0 додатків, 23 джерела.*

Ключові слова: *веб-розробка, CRM, databases, RAG, front-end, back-end, API, рекрутинг, підбір персоналу, середній та малий бізнес*

Короткий зміст праці:

Кваліфікаційна робота присвячена розробці веб-орієнтованої CRM-системи для малого та середнього бізнесу, що працює в секторі рекрутингу, з акцентом на найм робітничих кадрів на польському ринку праці. У звіті проаналізовано предметну область програмного забезпечення для рекрутингу, оцінено глобальні та локальні ринкові тенденції, а також представлено порівняльний аналіз існуючих рішень.

CRM-система реалізована з використанням сучасних технологій, включаючи Django для бекенду, PostgreSQL для управління даними та React для фронтенду. Для підтримки різних типів користувачів, таких як адміністратори, рекрутери, фрілансери та менеджери, було впроваджено рольову модель управління доступом (RBAC).

Основний функціонал включає централізоване управління кандидатами, відстеження проектів і вакансій, аналітику рекрутингу. Особливу увагу було

приділено масштабованості та нормалізації бази даних, що забезпечує ефективність та узгодженість системи.

Крім того, система впроваджує концепцію штучного інтелекту в процеси рекрутингу завдяки реалізації механізмів семантичного пошуку та Retrieval-Augmented Generation (RAG).

Розроблена CRM-платформа має на меті заповнити критичну прогалину на ринку програмного забезпечення для підбору персоналу для невеликих агентств, пропонуючи зручне, автоматизоване та інтелектуальне рішення, пристосоване до динамічних потреб масового найму в Центральній та Східній Європі.

The qualification work is dedicated to the development of a web-oriented CRM system for small and medium-sized businesses operating in the recruitment sector, with a focus on hiring blue-collar personnel in the Polish labor market. The report analyzes the subject domain of recruitment software, evaluates the global and local market trends, and presents a comparative overview of existing solutions.

The CRM system architecture is implemented using modern technologies, including Django for the backend, PostgreSQL for data management, and React for the frontend. A role-based access control model was incorporated to support different user types such as administrators, recruiters, freelance agents, and managers.

The core functionality includes centralized candidate management, project and vacancy tracking, recruitment analytics. Special attention was given to scalability and database normalization, ensuring system efficiency and consistency.

In addition, the system introduces the concept of artificial intelligence in recruitment processes through the implementation of semantic search and Retrieval-Augmented Generation (RAG) mechanisms.

The developed CRM platform aims to fill a critical gap in the recruitment software market for small agencies, offering a user-friendly, automated, and intelligent solution tailored to the dynamic needs of mass hiring in Central and Eastern Europe.

ЗМІСТ

ВСТУП

У сучасному світі, який характеризується динамічним розвитком технологій та активною інтеграцією інтелектуальних інформаційних систем у всі сфери діяльності, особливої актуальності для бізнесів набуває питання автоматизації бізнес-процесів, особливо у канонічних сферах, як облік, бухгалтерія та управління персоналом. Зростаюча конкуренція на ринку праці, необхідність швидкого пошуку кваліфікованих працівників, а також тенденції до глобалізації робочої сили потребують впровадження ефективних інструментів для обліку, аналітики та взаємодії з кандидатами. Одним із ключових таких інструментів виступають системи управління взаємовідносинами з кандидатами / клієнтами — Recruitment CRM (Candidate/Client Relationship Management).

Особливої важливості такі системи набувають у сегменті низькокваліфікованого персоналу, де процеси набору відзначаються високою динамікою, великим обсягом заявок, частими змінами у статусах кандидатів та високими вимогами до швидкості роботи рекрутерів. Водночас, існуючі на ринку CRM-рішення здебільшого орієнтовані на корпоративний сектор або висококваліфіковані кадри, залишаючи сегмент рекрутингу робітничих спеціальностей недостатньо охопленим та відносно відкритим.

Мета даної роботи полягає у розробці інформаційної системи Recruitment CRM, яка б враховувала специфіку ринку Польщі, орієнтувалася на мультикомпанійне використання, дозволяла ефективно управляти даними про кандидатів, вакансії, проєкти, рекрутерів, а також реалізовувала функціональність на базі сучасного технологічного стеку (Django, PostgreSQL, React) із можливістю інтеграції елементів штучного інтелекту (зокрема, семантичного пошуку та Retrieval-Augmented Generation).

Актуальність теми зумовлена різними факторами, зокрема потребою в цифровізації рекрутингових процесів, зменшенні впливу людського фактора на якість прийняття рішень, необхідністю швидкого доступу до релевантної інформації та використанням сучасних AI-технологій для пошуку, фільтрації та

аналізу кандидатських даних.

Станом на 2023 рік, кадрові агенції працевлаштовували більше мільйона осіб тільки у польщі за рік. Більше 8799 агенцій провадило свою діяльність у цьому регіоні, при цьому щороку відкривається близько 2000 нових компаній у цій сфері, результуючи у оборот капіталу на ринку більш ніж 5.8 мільярдів злотих (реальний об’єм ринку імовірно більший)[15].

Мета дослідження полягає у розробці веборієнтованої CRM-системи з підтримкою можливості користування багатьма компаніями, багаторольової моделі доступу (RBAC), включенням аналітичних та AI-компонентів для підвищення ефективності роботи HR-відділів тощо.

Об’єктом дослідження є процеси цифрового управління рекрутинговими даними, зокрема в межах аутсорсингових компаній, які здійснюють підбір персоналу на позиції низької або середньої кваліфікації.

Предметом дослідження виступає інформаційна система CRM-класу, її архітектура, моделі даних, алгоритми обробки інформації, а також технічні, програмні та математичні засоби реалізації системи з урахуванням можливостей подальшої AI-інтеграції.

РОЗДІЛ 1. ЗАГАЛЬНІ ПОЛОЖЕННЯ. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметного середовища

Світовий ринок програмного забезпечення для рекрутингу (ATS/CRM) демонструє стабільне зростання. Станом на 2023 рік його обсяг оцінюється приблизно в $2,3 млрд (для порівняння, у 2017 році було ~$1,78 млрд)[1]. Прогнозується подальше розширення цього ринку: за деякими оцінками, він сягне близько $3,6 млрд до 2032 року (середньорічні темпи зростання близько 5,1%)[1]. Інші дослідження дають подібну картину: наприклад, глобальний сегмент онлайн-рекрутменту (електронного найму) оцінено в $1,8 млрд у 2022 році з прогнозом зростання до $4 млрд у 2032 році (CAGR ~8,6% у 2023–2032 рр.)[2].

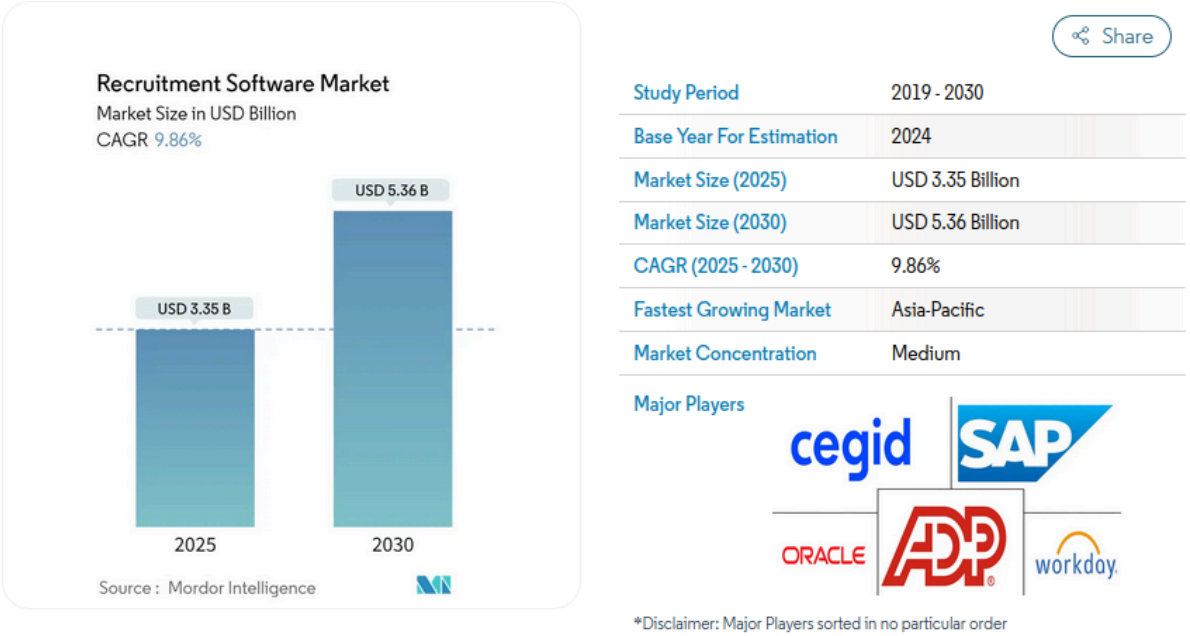

Рис 1.1.1 Статистика ринку програмного забезпечення для сфери підбору персоналу [9]

Джерело: https://www.mordorintelligence.com/industry-reports/recruitment-software-market

Основними драйверами зростання є впровадження хмарних платформ, використання штучного інтелекту та машинного навчання у підборі персоналу, а також активніше залучення соцмереж і мобільних технологій для пошуку кандидатів [1]. Важливим фактором стала й цифрова трансформація HR-процесів у відповідь на пандемію COVID-19, що прискорила перехід до дистанційного найму та автоматизації рекрутингу [1].

Світовий ринок рекрутингових CRM/ATS значною мірою орієнтований на потреби великого бізнесу. За статистикою, близько 99% корпорацій з рейтингу Fortune 500 використовують системи ATS у своєму процесі найму[3]. Загалом ~70% великих компаній застосовують ATS, тоді як серед малого та середнього бізнесу проникнення цих технологій поки що близько 20% [3]. Приблизно 75% рекрутерів у світі користуються ATS або подібними інструментами для обробки заявок кандидатів і покращення *candidate experience* (досвіду кандидатів) [3]. Більшість фахівців з підбору персоналу відзначають позитивний вплив таких систем: 94% рекрутерів стверджують, що їхня ATS підвищила ефективність процесів найму в організації [3].

Глобальний ринок рекрутингових систем представлений як великими міжнародними вендорами, так і спеціалізованими рішеннями для окремих сегментів. Помітну частку займають корпоративні HRM-платформи: зокрема, за оцінками аналітиків, компанії Oracle, IBM та SAP разом контролюють суттєву частку ринку ATS [1]. До найбільших постачальників також належать такі системи, як iCIMS, Workday, ADP, Cornerstone OnDemand, Bullhorn, Jobvite, Greenhouse тощо, кожна з яких має від ~3% до 10% світового ринку[1]. Ці рішення здебільшого орієнтовані на великий бізнес і часто інтегруються з комплексними HRM-системами підприємств.

Поряд із ними існує широкий спектр систем, націлених на середній та малий бізнес, а також на рекрутингові агенції. До відомих глобальних продуктів для рекрутингу належать, наприклад, **Teamtailor**, **Zoho Recruit**, **Recruitee**, **Workable**, **Lever** та інші. Вони пропонують зручні *SaaS*-рішення для

автоматизації підбору персоналу у різних країнах. Натомість у окремих країнах сформувалися і свої сильні гравці. *На польському ринку*, зокрема, популярними є локальні CRM/ATS-системи такі як **eRecruiter**, **HRappka**, **Traffit**, **Elevato**, **Element** тощо [4]. Багато великих польських роботодавців користуються саме такими системами, а дехто збудував власні внутрішні ATS під свої потреби.

Польський ринок CRM у рекрутингу. У Польщі найбільшою рекрутинговою системою є eRecruiter – рішення, що з'явилося 2009 року і належить до групи компаній **Pracuj.pl** (лідера онлайн-ринку праці) [5]. Станом на сьогодні eRecruiter обслуговує понад 2 000 корпоративних клієнтів і щороку допомагає опрацьовувати понад 15 млн кандидатів[6]. Ця система фактично стала стандартом для великих польських роботодавців, пропонуючи широкий набір функцій: мульти-постинг вакансій (тісна інтеграція з порталом Pracuj.pl), гнучке налаштування етапів відбору, інструменти командної роботи з *hiring manager*-ами, аналіз резюме та інше. В eRecruiter нещодавно навіть з'явилися елементи AI – наприклад, модуль на базі ChatGPT для автоматичного розподілу кандидатів по відповідних вакансіях. Водночас, це рішення позиціонується у верхньому ціновому сегменті: модель ціноутворення непрозора (вартість підключення визначається індивідуально), а сама система містить настільки багатий функціонал, що новим користувачам може бути складно одразу ним опанувати. Незважаючи на ці недоліки, eRecruiter залишається провідним інструментом HR на польському ринку, регулярно оновлюється та підтримує високі стандарти безпеки (відповідність GDPR тощо)[5].

Іншим помітним локальним продуктом є **HRappka** – комплексна HRM/CRM-система, спеціально розроблена для агентств з працевлаштування та внутрішніх HR-відділів. На відміну від eRecruiter, HRappka охоплює не лише рекрутинг, а й кадровий облік, розрахунок заробітної плати, управління навчанням та інші HR-процеси в межах єдиної платформи. Сильними сторонами HRappka користувачі відзначають інтуїтивність інтерфейсу та високу автоматизацію рутинних завдань, зокрема: мультиканальне розміщення вакансій,

ведення бази кандидатів з тегами і фільтрацією, генерація документів (договорів), планування співбесід, мобільний додаток тощо. Система підтримує роботу з іноземними кандидатами (інтеграція з зарубіжними джоб-порталами, багатомовний інтерфейс)[7], що актуально для польського ринку робочої сили. Водночас HRappka критикують за не надто зрозумілу цінову політику та додаткову плату за навчання користувачів, що може бути бар'єром для дрібних компаній. Також відзначається, що рекрутинг-модуль хоча і досить функціональний, але історично є доповненням до основного HR-блоку системи.

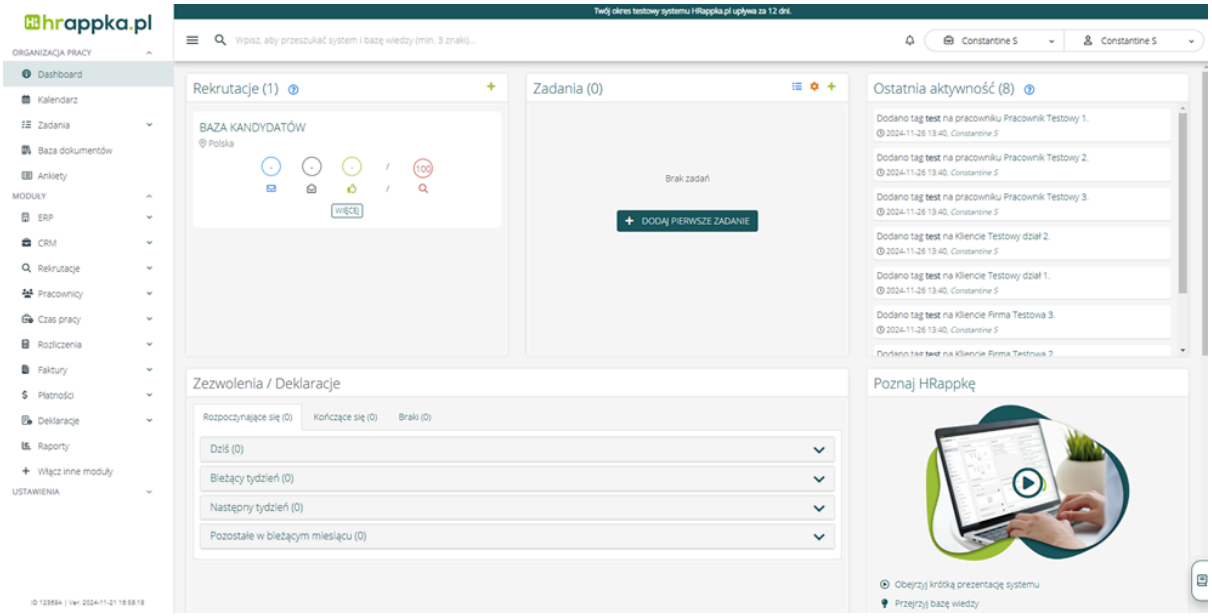

Рис. 1.1.2 Головне вікно "HRAppka"

Джерело: https://hrappka.pl

Загалом, рекрутингові CRM/ATS набувають все більшого поширення як у світі, так і в Польщі, хоча рівень впровадження залежить від розміру компаній. У корпоративному сегменті ці інструменти вже стали невід'ємною частиною процесу найму, тоді як серед менших роботодавців їх використання лише набирає обертів. Польський ринок рекрутмент-ПЗ зростає слідом за глобальними тенденціями: компанії усвідомлюють потребу скорочувати час закриття вакансій та конкурувати за кадри, впроваджуючи сучасні системи обліку кандидатів. За даними опитувань, понад половина роботодавців планує збільшити інвестиції в

ATS-рішення, усвідомлюючи їхню віддачу. Важливим аспектом є дотримання законодавства та конфіденційності даних – впровадження CRM у рекрутингу полегшує виконання вимог GDPR, автоматизуючи збір згод кандидатів та управління їхніми персональними даними. В цілому роль CRM-систем у сфері підбору персоналу стає визначальною: ці платформи підвищують продуктивність рекрутерів, дають аналітику для ухвалення рішень та покращують досвід як рекрутерів, так і кандидатів.

Впровадження CRM/ATS-системи приносить ряд конкретних переваг для процесу підбору персоналу:

Підвищення ефективності та швидкості найму. Автоматизація рутинних задач (розміщення вакансій, відбір резюме, планування інтерв'ю) суттєво скорочує трудовитрати і час закриття вакансій. За рахунок цифровізації процесів компанії економлять кошти: скорочується потреба в паперовій роботі та ручному адмініструванні. Наприклад, система може автоматично відсівати нерелевантні резюме за заданими критеріями – до 70% невідповідних заявок можуть одразу відхилятися ATS без перегляду рекрутером [3]. Це дозволяє рекрутинговим командам зосередитись на перспективних кандидатах. В результаті найм відбувається швидше, що особливо важливо, адже найкращі кандидати залишаються на ринку лише ~10 днів.

Формування централізованої бази кандидатів. CRM для рекрутингу виконує роль єдиного сховища даних про претендентів. Усі анкети, резюме, результати інтерв'ю та комунікація зберігаються в системі, структуровані за вакансіями та статусами. Це створює власний *talent pool* компанії – базу потенційних кандидатів на майбутні потреби. Рекрутери можуть легко знаходити попередніх кандидатів по тегах, навичках, місцезнаходженню тощо, не починаючи кожен пошук «з нуля». Така база даних підвищує якість підбору, адже дозволяє будувати довгострокові відносини з кандидатами (концепція *Candidate Relationship Management*). За рахунок CRM компанія підтримує контакт з перспективними спеціалістами, навіть якщо ті не були найняті одразу, і може

залучити їх у майбутньому.

Покращення комунікації та досвіду кандидатів. Вчасний і персоналізований зворотний зв’язок кандидатам – ключовий чинник привабливості роботодавця. CRM-система спрощує спілкування: шаблони і автоматичні розсилки дозволяють інформувати претендентів про статус їхньої заявки, надсилати запрошення на співбесіду, відмови тощо. Наприклад, система може автоматично надсилати *autoresponder*-листи кожному новому кандидату, дякуючи за заявку. Це створює позитивний досвід для шукачів роботи і підвищує імідж роботодавця. За даними досліджень, використання ATS безпосередньо впливає на *candidate experience*: більшість рекрутерів відзначають, що технології допомогають підтримувати регулярний контакт з кандидатами та покращувати їхню залученість. Окрім того, CRM дозволяє збирати зворотній зв’язок від кандидатів (наприклад, через опитування NPS після завершення процесу найму), щоб HR-відділ міг вдосконалювати свої процеси.

Підтримка командної співпраці та аналітики. Сучасні рекрутингові системи забезпечують зручну співпрацю рекрутера з керівниками найму та іншими членами команди. Через загальну платформу всі задіяні особи можуть залишати оцінки кандидатів, коментарі, бачити прогрес по вакансії в реальному часі. Наприклад, eRecruiter пропонує окремий кабінет для *hiring manager*-ів, де вони можуть переглядати кандидатів і давати зворотний зв’язок по кожному. Це спрощує прийняття колективних рішень щодо найму. Крім того, CRM-система генерує статистичні звіти: скільки кандидатів на кожному етапі, джерела найму, середній час закриття вакансій, ефективність роботи рекрутерів тощо. Така аналітика дозволяє керівництву HR оцінювати вузькі місця процесу і підвищувати KPI найму. За потреби, через **REST API** можна інтегрувати CRM з іншими системами (наприклад, HRIS або календарями) для ще більшої узгодженості даних.

1.2. Аналіз аналогічних розробок

Для визначення місця майбутньої системи на ринку варто аналіз існуючих рішень – як глобальних, так і польських CRM/ATS для рекрутингу.

Основними конкурентами майбутнього продукту є:

HRappka. Це польська комплексна система управління HR-процесами, орієнтована на рекрутингові агенції та компанії, що наймають масовий персонал. HRappka поєднує функції ATS і класичної HRM-системи: у ній є модулі обліку кадрів та зарплат, управління навчанням і відвідуваністю, CRM для роботи з клієнтами-замовниками, а також потужний рекрутинговий модуль. В частині рекрутингу HRappka забезпечує автоматизацію на кожному кроці: мультипостинг вакансій на різні сайти, ведення бази кандидатів (з можливістю пошуку по резюме, тегах, фільтрам), онлайн-анкети та тести для кандидатів, планувальник співбесід, розсилку повідомлень кандидатам і навіть мобільний додаток для рекрутерів. Важливо, що система підтримує багатокомпанійність – тобто агенція може вести окремо проєкти різних клієнтів і бази кандидатів для них. Також реалізована відповідність GDPR: кандидатам надсилаються форми-згоди, дані автоматично анонімізуються після завершення терміну зберігання. Користувачі відзначають, що HRappka є достатньо інтуїтивною і регулярно розширює свій функціонал. Водночас, слабким місцем називають високу початкову вартість впровадження та додаткові платежі за тренінги для персоналу, що може відлякати малі фірми. Крім того, оскільки HRappka історично розвивалася як «комбайн» для HR, її модуль рекрутингу не настільки глибоко спеціалізований, як у вузькопрофільних ATS (фактично рекрутинг є одним із багатьох модулів). Ця система найкраще підходить для агентств, які хочуть мати єдину платформу і для найму, і для подальшого управління працевлаштованими співробітниками.

eRecruiter. Найпоширеніша ATS-система в Польщі, що належить до групи **Grupa Pracuj** і тісно інтегрована з порталом Pracuj.pl [5]. eRecruiter позиціонується як універсальне рішення для корпоративного найму в середніх і

великих компаніях. Система працює на ринку понад 15 років і за цей час стала фактичним стандартом для багатьох роботодавців. Функціонально eRecruiter покриває всі базові потреби: розміщення вакансій, електронний збір відгуків, гнучке налаштування етапів відбору, шаблони листів кандидатам, інтеграція з календарями для планування інтерв'ю, управління базою резюме. Ще одна сильна сторона – аналітика та звітність: eRecruiter дозволяє будувати звіти по ефективності джерел найму, тривалості вакансій, завантаженості рекрутерів тощо, що важливо для HR-стратегії. Недоліками, за відгуками користувачів, є порівняно висока складність інтерфейсу (через велику кількість функцій новачкам важко швидко зорієнтуватися) та відсутність прозорих тарифів – ціну підключення можна дізнатися лише після індивідуальної презентації і оцінки потреб клієнта. Цільова аудиторія eRecruiter – переважно середні та великі роботодавці в Польщі, яким потрібен інструмент для обробки значних обсягів заявок.

Teamtailor. Популярна глобальна SaaS-платформа родом зі Швеції, яка позиціонується як «ATS нового покоління» з фокусом на імідж роботодавця та зручність для кандидатів. Teamtailor широко використовується у Європі й відома своїм сучасним інтерфейсом та гнучкими налаштуваннями. Ключова особливість – вбудований конструктор кар'єрного сайту: компанії можуть створювати власні сторінки вакансій з брендовим дизайном і контентом без залучення розробників. Це допомагає привертати кандидатів привабливою подачею вакансій. Система також підтримує інтеграцію з численними джоб-порталами для публікації оголошень, має модулі CRM (для роботи з *пасивними кандидатами*) та реферальну програму. Інтерфейс Teamtailor вирізняється **візуальним канбан-представленням** воронки найму: рекрутер бачить дошку з колонками етапів (нові, розгляд, інтерв'ю, оффер тощо) і може перетягувати картки кандидатів між етапами мишкою. Такий підхід інтуїтивно зрозумілий і надає швидкий огляд процесу. Teamtailor пропонує розвинені можливості колаборації – внутрішній чат для обговорення кандидатів, призначення відповідальних,

нагадування. Крім того, останні оновлення включають *AI*-функції: розшифровку та підсумовування відеоінтерв'ю, автоматичні тригери для відправки листів чи завдань при зміні статусу кандидата тощо [8]. Користувачі високо оцінюють Teamtailor за *user-friendly* дизайн та ефективне відстеження кандидатів на кожному етапі. У Польщі його використовують переважно міжнародні та *IT*-компанії; на ринку масового підбору Teamtailor менш відомий, хоча потенційно може застосовуватись і для найму лінійного персоналу завдяки простоті використання.

Zoho Recruit. Продукт індійської компанії Zoho Corporation, що входить до її екосистеми бізнес-додатків. Zoho Recruit – це хмарна ATS з елементами CRM, яка пропонує широку функціональність за відносно невисоку ціну, що приваблює малий та середній бізнес. Система буває у двох основних варіантах: для корпоративних HR-відділів і для рекрутингових агенцій (де є додатковий модуль управління клієнтами та замовленнями). До ключових можливостей належать: публікація вакансій на безліч платформ, вбудований парсинг резюме, фільтрація кандидатів за налаштованими тестами і питаннями, планування інтерв'ю (інтеграція з календарями), автоматизація листування (наприклад, відправка SMS та email при зміні статусу кандидата). Zoho Recruit дуже гнучко налаштовується під процеси замовника – користувач може створювати власні поля, налаштовувати робочі процеси, права доступу тощо. Серед плюсів системи відзначають розгорнуту аналітику і звітність щодо найму, а також тісну інтеграцію з іншими продуктами Zoho (CRM, Projects, People тощо). Для польського ринку Zoho Recruit цікавий тим, що має україномовну та польськомовну версії інтерфейсу, підтримує багато валюті форматів дат, а отже може використовуватись локальними компаніями. Проте значного поширення в Польщі він поки не набув, програючи місцевим гравцям (тим же eRecruiter або Traffit) в знаності бренду.

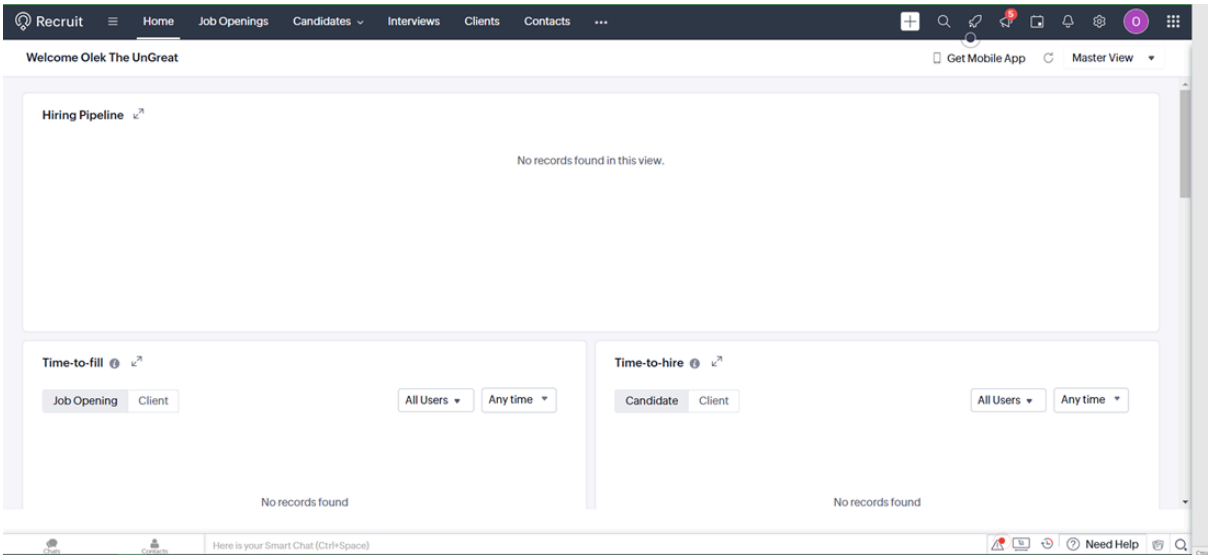

Рис. 1.2.1 Головне вікно “Zoho Recruit”

Джерело: https://recruit.zoho.eu/

Recruitee. Нідерландська SaaS-система, яка завоювала популярність у Європі завдяки *user-friendly* підходу та можливостям спільної роботи. Recruitee позиціює себе як «collaborative hiring software», тобто платформа для колективного найму, де рекрутери, менеджери і члени команди легко взаємодіють під час відбору кандидатів. Система має сучасний інтерфейс з настроюваними конвеєрами найму (drag-and-drop дошка вакансій, подібна до Teamtailor) та пропонує широкі можливості інтеграцій. Однією з особливостей Recruitee є CareerHub – вбудований модуль для створення сторінки кар’єри компанії з мінімальними зусиллями. Також Recruitee надає браузерне *sourcing*-розширення, що дозволяє імпортувати кандидатів прямо з LinkedIn чи інших сайтів у пару кліків. На польському ринку Recruitee присутня і конкурує з місцевими ATS: її обирають переважно *tech*-компанії та стартапи, які цінують міжнародний досвід цього продукту. Для сегменту масового найму Recruitee теж застосовна, оскільки дозволяє ефективно вести одразу багато вакансій і кандидатів, але важливо врахувати відсутність суто польської специфіки.

1.3. Постановка задачі

Для того, щоб адекватно поставити задачі для проекту спочатку необхідно оцінити хто буде його використовувати, яка цільова аудиторія, і робити продукт

клієнтоорієнтованим. Розроблювана CRM-система буде орієнтована на невеликі рекрутингові агентства та підприємства, які займаються підбором некваліфікованого/робітничого персоналу в Польщі (agencji pracy). Це сегмент ринку, який наразі недостатньо охоплений існуючими рішеннями: більшість популярних ATS спрямовані або на великий бізнес, або на *white-collar* найм, тобто висококваліфікований персонал. В той же час, аутсорсингові агенції, що масово закривають вакансії робітничих спеціальностей (склади, виробництво, логістика тощо), потребують спеціалізованого інструменту.

Цільова аудиторія системи – це рекрутери і менеджери малого агентства або HR-відділу (5–15 користувачів), які паралельно ведуть десятки вакансій для різних клієнтів і опрацьовують значний потік кандидатів щомісяця. Ринок географічно обмежується Польщею, з можливим масштабуванням на сусідні країни Центральної та Східної Європи (де подібна ситуація на ринку праці). Сегмент – переважно **blue-collar recruitment**, тобто найм працівників без вищої освіти / кваліфікації, часто іноземців, на типові вакансії з високою плинністю. Для цього сегменту важлива максимальна автоматизація та простота, адже обсяги резюме великі, а ресурси HR-відділу обмежені.

Вимоги до системи. Виходячи з потреб цільових користувачів та спілкування з компаніями та працівниками у даній сфері, включаючи проведення інтерв’ю, сформовано такі основні вимоги до функціональності, дизайну майбутньої CRM:

Функціональні вимоги:

Централізоване управління даними: Єдина платформа для зберігання, організації та доступу до всієї інформації, пов'язаної з кандидатами, працівниками, проектами та пропозиціями роботи.

Оптимізоване відстеження кандидатів: Інструменти для відстеження кандидатів протягом усього процесу рекрутингу, від подачі заявки до працевлаштування.

Управління вакансіями: Повний життєвий цикл вакансії – від створення та публікації оголошення до закриття. Можливість налаштовувати етапи відбору під конкретну вакансію (наприклад, телефонна співбесіда, тестове завдання, інтерв'ю на місці тощо) і переміщувати кандидатів між етапами із фіксацією результатів.

Аналітика і звітність: Вбудовані звіти щодо ключових показників рекрутингу – кількість відкритих вакансій, кількість кандидатів на етапах, середній час закриття вакансії, коефіцієнт відгуку з різних джерел, ефективність роботи окремих рекрутерів. Для агентства важливі також звіти по клієнтах (скільки кандидатів найдено для кожного замовника, статуси вакансій клієнта тощо).

Мультикомпанійність: Система повинна підтримувати ведення декількох компаній/клієнтів одночасно в одному обліковому записі агенції. Тобто рекрутер бачить вакансії розподіленими за клієнтами, база кандидатів може фільтруватися за компаніями, і доступ клієнтів до своїх вакансій із власним кабінетом (якщо планується) теж ізольований.

Інтеграції: Бажано, хоча не обов'язково, забезпечити інтеграцію з популярними в Польщі джоб-порталами для автоматичного розміщення вакансій. Варто зазначити, що система фокусується саме на роботі з даними кандидатів та працівників у середині компаній і грає значно меншу роль у пошуку кандидатів.

Додатково: Підтримка зберігання файлів (резюме, скани документів) в прив'язці до профілю кандидата; модуль управління базою клієнтів-роботодавців (простий CRM для відстеження контактів замовників і їх вакансій); журнал подій (логування дій користувачів для безпеки і аудиту).

Висновки

Підсумовуючи, ринок ПЗ для сфери підбору персоналу та рекрутингу дуже активно розвивається як глобально, так і в Польщі. Найчастіше системи CRM/ATS використовують великі компанії, сегмент малого та середнього

бізнесу, особливо у сфері низькорівневого персоналу залишається відносно відкритим.

На локальному ринку вже присутні сильні гравці як eRecruiter та HRappka, проте навіть у них є ряд недоліків. Іноземні рішення, як Teamtailor, Zoho Recruit чи Recruitee теж мають переваги, проте не адаптовані до місцевого ринку та вимог. Саме тому створення нової CRM-системи, орієнтованої на малі агентства з підбору blue-collar персоналу в Польщі, є актуальним і затребуваним.

Подібна система має бути простою у користуванні, автоматизовувати та покращувати процеси роботи рекрутерів і бути пристосованою до специфіки ринку.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз предметної області

Предметною областю проєкту є CRM-система для рекрутингу, що підтримує процеси підбору персоналу та управління взаєминами з кандидатами.

Основним об’єктом дослідження є програмне середовище, в якому автоматизуються типові рекрутингові процеси: збір та обробка інформації про кандидатів, керування вакансіями і проєктами найму, відстеження результатів та статистики тощо.

Щоб краще описати інформаційну модель та роботу з вхідними і вихідними даними, спочатку опишемо основні процеси системи:

Додавання кандидатів рекрутерами: Рекрутер створює в системі новий запис кандидата, вносячи його особисті дані, резюме, навички, досвід тощо. Кандидат фіксується у базі даних та закріплюється за відповідальним рекрутером.

Переведення кандидата в працівники: Якщо кандидат успішно проходить всі етапи відбору та приймає пропозицію, рекрутер змінює його статус. Система переводить запис кандидата до списку працівників, зберігаючи інформацію про зайняту посаду, проєкт або компанію, куди його прийнято.

Створення проєктів і вакансій: Система дозволяє адміністраторам або рекрутерам створювати **проєкти** – умовні групи вакансій, що об’єднані спільною темою чи замовником (наприклад, проєкт найму для певного клієнта або департаменту). У межах проєкту створюються конкретні **вакансії** (позиції), які потрібно закрити. Вакансія містить опис посади, вимоги, та пов’язана з відповідальним рекрутером і проєктом.

Формування статистики та аналітика: CRM-система збирає дані та аналітику з усіх процесів. На основі цих даних генеруються звіти і метрики. Користувачі з відповідними правами (наприклад, адміністратор компанії) можуть

переглядати ці статистичні показники у вигляді діаграм та звітів для прийняття управлінських рішень.

Інформаційна модель системи включає основні сутності та зв’язки між ними. До головних сутностей належать: Кандидат, Працівник, Рекрутер, Проєкт, Вакансія, Компанія і Клієнт.

Кандидат – потенційний співробітник, інформація про якого зберігається в системі (ПІБ, контактні дані, резюме, навички, етапи проходження відбору тощо). Кандидат може бути прив’язаний до однієї або кількох вакансій. Якщо кандидат успішно наймається, його статус змінюється на працівника, але історія кандидата зберігається для аналітики.

Працівник – сутність, що репрезентує кандидата, який пройшов успішно відбір та прийнятий на роботу. Працівник може бути пов’язаний із певною компанією (якщо система використовується рекрутинговою агенцією, працівник може належати до компанії-клієнта). Дані працівника розширюють і доповнюють дані кандидата, оскільки на етапі переходу у роль працівника компаніям потрібно зберігати значно більше інформації про людей (документи, інформаці про звільнення / припинення роботи тощо).

Рекрутер – користувач системи, який займається підбором персоналу. За кожним рекрутером закріплюється група кандидатів, яких саме він знайшов і веде. Рекрутер взаємодіє з вакансіями (створює їх або отримує в роботу від адміністратора/менеджера), проводить оцінку кандидатів та відмічає зміни їх статусу.

Проєкт – об’єкт/компанія де будуть працювати люди. Проєкт може відповідати окремому клієнту рекрутингової агенції або внутрішньому напрямку найму в компанії (наприклад склад, що потребує 50 працівників на різні позиції). Кожен проєкт містить вакансії та інформацію про затребування. Проєкт дозволяє згрупувати вакансії для спільного контролю та аналітики (на рівні проєкту можна відстежувати сумарну кількість закритих позицій, ефективність тощо).

Вакансія – відкрита позиція, на яку ведеться найм. Вакансія належить до певного проєкту і містить опис посади, вимоги до кандидата, статус (відкрита, на паузі, закрита) та іншу інформацію необхідну для ефективного пошуку кандидатів. З вакансією пов’язані кандидати, що подалися або були розглянуті на цю позицію. Кількість кандидатів на одну вакансію необмежена; одночасно один кандидат може розглядатися на декілька вакансій, хоча це і не частий випадок на практиці.

Компанія – юридична особа або організація, в рамках якої ведеться рекрутинг. Поняття компанії є важливим для мультитенантної (багатокомпанійної) архітектури: система може обслуговувати декілька компаній одночасно, ізольовано одна від одної. Кожна компанія має власних користувачів (рекрутерів, адміністраторів, клієнтів) та свій набір даних (кандидати, вакансії, проєкти). Таким чином, мультикомпанійність забезпечує ізоляцію даних – користувачі однієї компанії не мають доступу до даних іншої.

Зв’язки між вказаними об’єктами формують структуру даних системи. Ключові зв’язки такі:

Компанія – Штатний Рекрутер: *1 до N*. Компанія може мати кілька рекрутерів, тоді як рекрутер прив’язаний тільки до однієї компанії.

Компанія – Фріланс Рекрутер: *N до N*. Будь яка кількість рекрутерів фрілансерів може працювати з будь якою кількістю компаній. Тобто рекрутер може шукати на позиції різних компаній, якщо вони зробили ці вакансії публічними.

Компанія – Проєкт: *1 до N*. В межах однієї компанії може вестися кілька проєктів найму. Проєкт належить саме тій компанії, в якій він створений; між різними компаніями проєкти не перетинаються.

Проєкт – Вакансія: *1 до N*. Кожен проєкт містить одну або більше вакансій.

Вакансія – Кандидат: *N до N*. Один кандидат може бути пов’язаний із кількома вакансіями. Так само, на одну вакансію може претендувати багато кандидатів.

Рекрутер – Кандидат: *1 до N*. Рекрутер додає кандидатів і працює з ними, тому кожен кандидат прив’язаний до певного рекрутера-куратора. Таким чином, один рекрутер може вести багато кандидатів, але кожен кандидат закріплений лише за одним відповідальним рекрутером. І хоча можуть бути ситуації коли кандидат звертається до іншого рекрутера, наприклад через певний час або з приводу іншої вакансії, тим не менш тоді цей кандидат просто буде закріплений за новим рекрутером і не може бути ведений двома одночасно. Більше того це відповідає бізнес-правилу, згідно з яким рекрутер відповідає за «своїх» кандидатів і, як правило, має доступ тільки до них. Наприклад, якщо два рекрутери працюють у одній компанії, кожен бачить у системі лише профілі своїх кандидатів і не має доступу до бази кандидатів колеги – така ізоляція забезпечує конфіденційність кандидатських баз і внутрішню конкуренцію.

Приклади бізнес-обмежень: Окрім вже згаданої ізоляції кандидатів за рекрутерами, інформаційна модель накладає й інші логічні обмеження. Наприклад, рекрутер (не адміністратор) не може бачити проєкти або вакансії, що не належать його компанії. В системі також буде реалізовано правило, що одна вакансія прив’язана до одного проєкту і не може існувати поза проєктом – для підтримки чистоти структури.

Таким чином виглядають обмеження на вхідні та вихідні дані системи та правила їх роботи.

2.2 Проектування системи

Для проєктування системи використано UML нотації та інші діаграми, які дозволили формально описати вимоги, структуру даних та архітектуру застосунку у зрозумілому для навіть не технічних людей форматі.

Діаграма варіантів використання (Use Case diagram): Це графічне представлення, що ілюструє взаємодію зовнішніх акторів (користувачів або систем) із програмною системою. Діаграма варіантів використання відображає функціональні вимоги до системи у вигляді сценаріїв використання: вона показує, які дії (use cases) можуть виконувати різні категорії користувачів. Use Case-діаграми є зручним інструментом на етапі аналізу вимог, адже наочно демонструють, які саме можливості надає система кожній ролі користувачів і як ці ролі взаємодіють із системою.

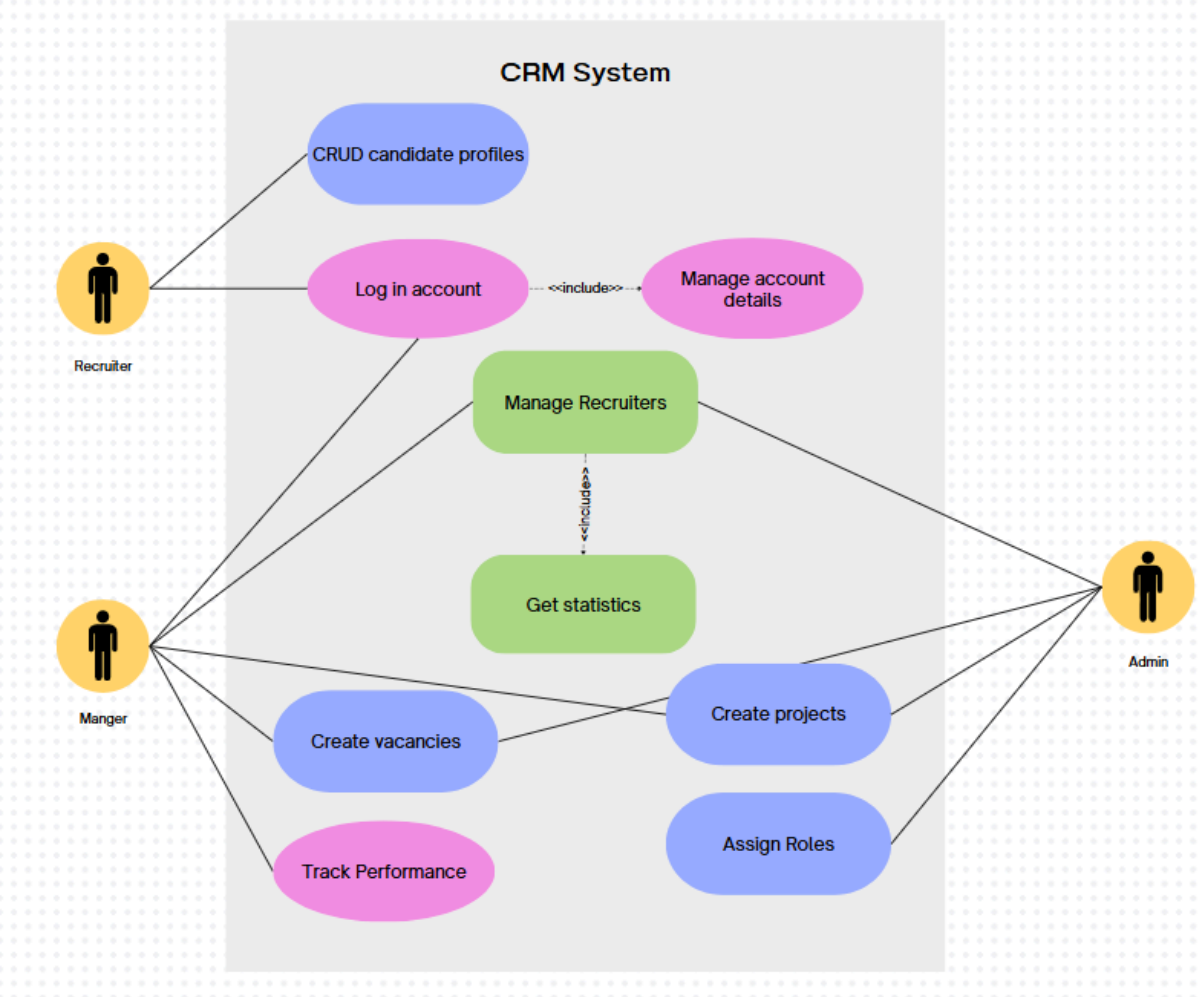

Рис. 2.2.1 Приклад Use-case діаграми

Джерело: розроблено автором

ER-діаграма (Entity-Relationship діаграма) — це інструмент проєктування структури бази даних. Вона відображає сутності (entity) – тобто основні таблиці чи об’єкти у системі, їх атрибути – та зв’язки (relationship) між ними. Ця діаграма

є інструментом для побудови схем даних таким чином, щоб вони точно відображали вимоги системи та структуру майбутньої БД.

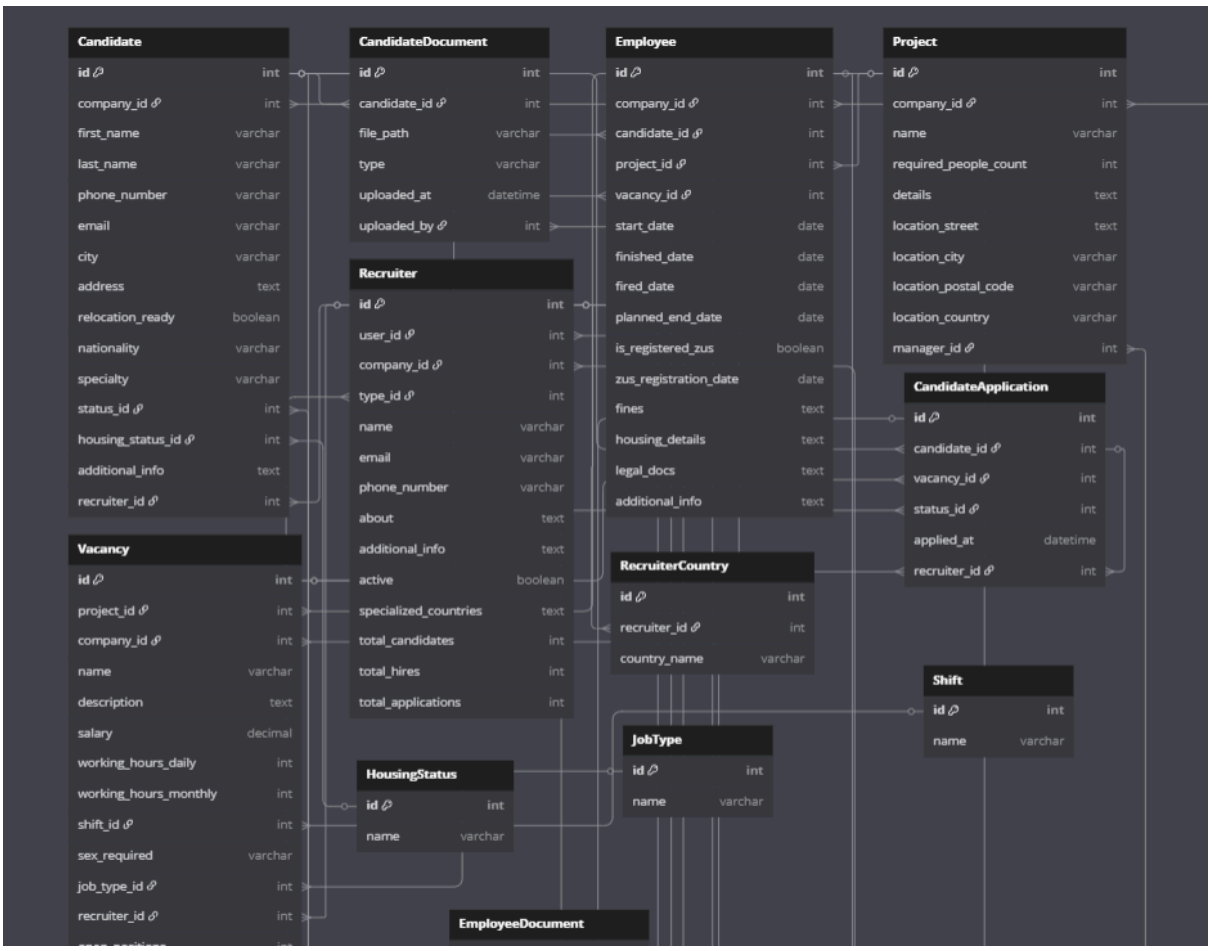

Рис. 2.2.2 Приклад частини схеми бази даних (ER діаграми)

Джерело: розроблено автором

У загальному даний проект створюється та проектується зразу з дуже дальноглядними планами. Відповідно на етапі планування закладється те, що платформа буде призначена для обслуговування декількох компаній. Відповідно одним з перших архітектурних рішень є відокремлення даних кожної компанії. У цьому випадку схема бази даних використовує практичний та ефективний підхід: кожна основна таблиця (наприклад, Candidate, Employee, Project, Vacancy та Recruiter) прив'язана до конкретної компанії за допомогою поля company_id. Ця модель спільної схеми зберігає всі дані в одній базі даних, але при цьому з чітким зазначенням, до якої компанії вони належать.

Така схема є економічно вигідною та простою в обслуговуванні. Оскільки всі компанії використовують однакові таблиці, існує лише одна база даних для резервного копіювання, оновлення та масштабування. Це звичайний вибір для SaaS-продуктів, особливо на ранніх і середніх стадіях зростання. Звісно тут є свої нюанси, кожен запит повинен перевіряти company_id, щоб переконатися, що користувачі можуть бачити тільки свої власні дані. Django допомагає з цим, або вручну додаючи фільтр до кожного запиту, або використовуючи бібліотеки, які підставляють фільтр автоматично. Це не тільки запобігає помилкам, але й зменшує ризик витоку даних.

Ця модель також є масштабованою, вона може підтримувати багато компаній клієнтів з мільйонами рядків даних, якщо є відповідні індекси - особливо для стовпця company_id та інших полів, за якими часто здійснюють пошук. При цьому не всі дані можуть бути специфічними для клієнта. Деякі довідкові дані, як-от статуси чи категорії посад, можуть бути однаковими для всіх компаній [16].

Окрім цього важливо також передбачити можливість масштабування бази даних та нормалізувати її для уникнення аномалій та проблем при подальшому користуванні. Досягнувши третьої нормальної форми (3НФ), можна уникнути аномалій при оновленні та неузгодженості даних.

Кожна важлива сутність — Candidate, Employee, Vacancy, Project — має свою таблицю. А такі речі як JobType, Status, Shift, RecruiterType виступають у ролі lookup-таблиць. Це дозволяє не повторювати текстові значення всюди, а зберігати їх в одному місці.

Lookup-таблиці пов’язані з основними таблицями через зовнішні ключі — наприклад, vacancy.job_type_id має посилається на таблицю **JobType**, а статус кандидата — на таблицю **Status**. Такі зв’язки не тільки захищають від помилок (наприклад, від неіснуючих статусів), а й документують структуру системи.

Такий підхід мінімізує дублювання й підвищує консистентність. Наприклад, інформація про компанію зберігається тільки в таблиці **Company**, а в

інших таблицях є лише company_id. Це зручно: змінити назву компанії потрібно в одному місці, а не по всій базі.

Із потенційних недоліків такої системи роботи може бути те, що для виконання запитів інколи доводиться робити join'и таблиць, які при зростанні обсягу даних можуть уповільнювати систему. Проте для PostgreSQL це не проблема — з правильними індексами і на сучасному обладнанні навіть мільйони рядків обробляються швидко.

У випадку коли система виросте до сотень компаній і мільйонів записів, можна починати задумуватись про **partitioning** — розбиття великих таблиць за компанією або датою. Це дозволяє ефективніше видаляти дані певного клієнта або оптимізувати зберігання активних і неактивних записів.

У більшості рекрутингових систем кандидат після найму перетворюється на працівника. Якщо залишити їх у різних таблицях, частина інформації дублюватиметься: ім'я, контакти тощо. Це не критично, але значно краще просто зробити зв'язок між ними та не повторювати ту ж інформацію двічі.

Звісно можливо було б зберігати і **Candidate**, і **Employee** в одній таблиці з полем status, але тоді з'являється плутанина: адже в працівника та кандидата часто зовсім різні поля (зарплата у одного та резюме в іншого) [17].

Загальна архітектура системи спроектована за принципом багаторівневої клієнт-серверної архітектури. Можна виділити такі рівні: frontend (клієнтська частина), backend (сервер застосунку та API) та база даних. На frontend реалізовано веб-додаток на основі React, що забезпечує динамічний інтерфейс користувача. Backend побудовано на фреймворку Django (мовою Python) з використанням надбудови Django REST Framework, проте більш детально про це у розділі 3.1.

Багаторівнева архітектура забезпечує чіткий розподіл обов'язківміж клієнтською частиною, що відповідає за презентацію та зручність роботи користувача, сервером – бізнес-логіка, правила та безпека, **база даних** – надійне зберігання та вибірка інформації. Подібна архітектура є стандартною для

сучасних веб-додатків і полегшує масштабування та підтримку системи – кожен рівень можна модифікувати або масштабувати незалежно (наприклад, при зростанні навантаження можна окремо масштабувати сервер додатка і базу даних).

Не менш важливим аспектом архітектури та планування розробки даної CRM є приділення уваги ролям та різним користувачам системи — адмінам компаній, рекрутерам, фріланс-рекрутерам, менеджерам тощо. Щоб керувати доступом та забезпечити безпеку даних в залежності від ролі користувача (фріланс рекрутер не повинен бачити внутрішню статистику компаній з якими працює наприклад) найкраще підходить **RBAC** (role based access control, виділення доступів на основі ролей). Кожна роль має чітко визначені дозволи та доступ лише до своїх даних.

Основні ролі можуть бути наступними:

- Адміністратор компанії має повний контроль у межах своєї організації: керує користувачами, створює проєкти й вакансії, бачить усі дані компанії, включно з аналітикою та кандидатами всіх рекрутерів. Він не має доступу до інших компаній.

- Рекрутер працює з підбором персоналу: додає кандидатів, оновлює їх статуси, пов’язує з вакансіями. Але бачить лише «своїх» кандидатів, не має доступу до глобальних налаштувань чи аналітики.

- Фріланс-рекрутер — це зовнішній підрядник. Йому відкриваються лише конкретні вакансії, над якими він працює. Доступ до решти даних обмежений, що дозволяє безпечно залучати зовнішніх фахівців, не розкриваючи всю базу.

Менеджер компанії - отримує доступ до частин програми, як аналітика рекрутерів, статистичні дані, вакансії та кандидати, доступи схожі до адміністратора.

- Відділ легалізації - персонал що працює з даними працівників та кандидатів і має доступ до всіх документів у системі, проте не створює кандидатів або працівників.

2.3 Математичне та алгоритмічне забезпечення

У перспективі проєкту планується підтримка функціоналу **Retrieval-Augmented Generation (RAG)** – генерації відповідей за допомогою штучного інтелекту сформульованими на основі інформації в системі або джерелі даних. Відповідно алгоритми машинного навчання будуть застосовані для впровадження даного функціоналу.

2.3.1 Retrieval-Augmented Generation (генерація з доповненням пошуком)

RAG – це підхід, що поєднує можливості великих мовних моделей (LLM) з пошуком у зовнішній базі знань для підвищення якості та актуальності згенерованих відповідей. Ідея полягає в тому, що перед тим, як LLM сформує відповідь на запит користувача, система виконує **етап пошуку** релевантної інформації поза моделлю, і надає цю інформацію моделі як контекст [10]. За визначенням, Retrieval-Augmented Generation – це процес оптимізації відповіді LLM шляхом залучення авторитетної зовнішньої бази знань перед генерацією відповіді. Іншими словами, модель доповнюється актуальними даними ззовні, не обмежуючись тільки тим, що було в її тренувальних даних. Такий метод дозволяє подолати два істотні недоліки LLM: *галюцинації* (вигадування відповіді за відсутності знань) та *обмеженість знань* (статичність тренувальної вибірки, на яку накладається часовий «зріз»). RAG, по суті, «нагадує» моделі про свіжі факти або специфічні дані з корпоративної бази знань, не потребуючи перевчання моделі. Це значно підвищує релевантність, точність і корисність відповіді у вузьких доменах [10].

І звісно, це дозволяє використовувати не треновані моделі значно ефективніше, оскільки процес тренування моделей на власних даних може бути достатньо складним, це рішення є значно ефективніше.

Принцип роботи RAG можна розбити на кілька етапів, кожен з яких має своє математичне та алгоритмічне підґрунтя:

1. **Векторизація даних (Embedding):** Спершу готується база знань у зручному для пошуку вигляді. Документи, записи або інша текстова інформація, яка може стати в пригоді для відповідей, обробляється спеціальною моделлю-векторизатором (embedding model). Модель перетворює кожен текстовий фрагмент (наприклад, дані кандидата) на високорозмірний числовий вектор – цей процес називається **embedding**. Вектори відображають семантичний зміст тексту: схожі за змістом тексти перетворюються на близькі (майже паралельні) вектори у просторі. Отримані вектори зберігаються у спеціалізованій векторній базі даних або в сховищі, що підтримує швидкий пошук по векторах[10]. Фактично формується бібліотека знань, зрозуміла для генеративної моделі у математичному сенсі (через простір ознак). Ця стадія є підготовчою і може виконуватися офлайн або асинхронно (оновлюватися періодично).

2. **Обчислення вектору запиту:** Коли користувач ставить запитання до системи (наприклад: *"Скільки у нас зараз відкритих вакансій?"*), цей запит також пропускається через ту саму або подібну embedding-модель. На виході отримуємо вектор запиту – точку у тому ж семантичному просторі, що й вектори документів. Даний вектор відображає зміст запиту в числовій формі.

3. **Пошук релевантної інформації (Retrieval):** Далі система виконує **пошук найближчих сусідів**: порівнює вектор запиту з векторами бази знань, щоб знайти найбільш схожі. Для цього застосовуються метрики схожості/відстані між векторами – найчастіше **косинусна схожість** (cosine similarity) або евклідова відстань. Косинусна міра розраховується як косинус кута між двома векторами, і значення близькі до 1 означають велику схожість за змістом запиту і документах[11]. Завдяки векторному пошуку система може знайти, наприклад, ті документи, в яких міститься відповідь на запитання, навіть якщо формулювання відрізняється від запиту користувача. У нашому прикладі система знайде записи або поля бази, де зазначена кількість відкритих вакансій (наприклад, знайде останній звіт або підсумковий показник по вакансіях). Цей процес називається **semantic search** – пошук за змістом, а не за ключовими словами. Він значно

розширює можливості: як зазначено в джерелах, для задач з великими обсягами неструктурованих даних семантичний пошук знаходить потрібну інформацію точніше, ніж звичайний пошук за ключовими словами. В результаті цього кроку ми отримуємо невелику вибірку знайдених фрагментів інформації (наприклад, топ-5 документів або витягів, що найкраще підходять до запиту).

4. **Формування підказки моделі (Augmentation):** На цьому етапі знайдена інформація використовується для формування *розширеного запиту* до мовної моделі. Простіше кажучи, до початкового питання користувача додається контекст у вигляді обраних текстових фрагментів з бази знань. Це може бути реалізовано як: *"<Питання користувача> + Додаткова інформація: <уривки документів>"*. Така комбінована підказка (prompt) подається на вхід LLM. Таким чином, LLM вже володіє не лише власними параметрами, але й актуальними фактами, які були знайдені. Цей крок називається augmenting the prompt – доповнення промпту зовнішніми даними. Застосовуються техніки **prompt engineering**, щоб модель ефективно використала ці дані: наприклад, можна інструктувати модель на кшталт: "Враховуючи наведені дані, відповідай на запит користувача…" і далі передавати цитати з бази знань.

5. **Генерація відповіді (Generation):** Отримавши збагачений контекстом prompt, мовна модель генерує вихід – текст відповіді, що вже містить потрібну інформацію. Завдяки попередньому етапу, відповідь повинна бути більш точною та конкретною, адже модель "підглянула" у актуальні дані. У випадку CRM-системи це означає, що бот може дати відповідь, спираючись на поточну статистику чи профілі кандидатів, навіть якщо ці деталі не входили до тренувального корпусу моделі. Наприклад, на запит *"Скільки кандидатів було найнято минулого місяця?"* система витягне відповідний аналітичний показник з БД (скажімо, 5 кандидатів) і модель згенерує відповідь: *"Минулого місяця було найнято 5 кандидатів."* – замість узагальненої або некоректної відповіді, яку дала б модель без контексту.

6. **Оновлення бази знань:** Щоб забезпечити довгострокову актуальність, база знань повинна регулярно оновлюватися. Нові дані (наприклад, додано нових кандидатів, змінено статус вакансій) мають індексуватися – тобто проходити через embedding-модель – і додаватися до векторного індексу. Періодично варто вилучати застарілу інформацію. Цей процес може бути автоматизований. Таким чином, RAG-підсистема підтримує **актуальність знань**, синхронізуючись з основною БД CRM.

Архітектурно RAG можна уявити як поєднання двох компонент: **пошукового (retriever)** – векторна база + механізм similarity search, і **генеративного (generator)** – LLM моделі. Важливо, що між ними відбувається мінімальний інтерфейс – передача текстових фрагментів. Це робить рішення модульним: можна замінити базу знань або модель незалежно. Застосування RAG нині дуже широке: від чатботів підтримки користувачів, що черпають інформацію з документації, до аналітичних систем, які на природній мові відповідають на запити до корпоративних даних. В нашому випадку RAG підвищить **інтелектуальність CRM** – рекрутери чи клієнти зможуть у чат-формі запитувати систему про різні аспекти (наприклад, “Покажи мені всіх кандидатів з навичками Python, що пройшли технічну співбесіду”) і отримувати відповідь у зрозумілій формі, зібрану з різних таблиць системи. Це спрощує доступ до інформації, адже не потрібно вручну формувати запити чи будувати звіти – AI-асистент сам знаходить і подає дані.

2.3.2 Принципи роботи векторної бази даних

Як згадано, серцем RAG-системи є **векторна база даних** – сховище, оптимізоване для операцій над векторами. Вектори в нашому контексті – це результати роботи embedding-моделі, які перетворюють тексти (або інші об’єкти, наприклад, зображення) на точки в просторі високої вимірності. Наприклад, сучасна модель OpenAI *text-embedding-3-small* перетворює текст у вектори розмірності 1536 [12], тобто кожен документ стає набором з 1536 чисел. Ці числові «відбитки» мають ту властивість, що схожий за значенням масив чисел

відповідає схожому за змістом тексту. Як зазначають дослідники, embedding-вектори містять семантичну інформацію про текст, розподілену по багатьох вимірах простору ознак [13].

Звичайні реляційні бази даних не пристосовані для ефективного пошуку подібності векторів, оскільки для цього треба обчислювати відстані у багатовимірному просторі для великої кількості точок. Якщо зберігати вектори в SQL-таблиці та здійснювати повний перебір значень – це було б надто повільно у реальних масштабах. Тому виник клас спеціалізованих систем – **векторні бази даних**. За визначенням, векторна БД – це система, призначена для ефективного зберігання та пошуку високовимірних векторних даних [14]. Вона індексує та структурує вектори таким чином, щоб можна було дуже швидко знаходити найближчі сусіди (nearest neighbors) до заданого вектора запиту. Типова векторна база підтримує операції CRUD над векторами, фільтрацію за метаданими (для складних запитів, коли крім схожості треба, наприклад, врахувати ще певний атрибут документа), а також горизонтальне масштабування для роботи з мільйонами і мільярдами векторів.

Формування векторів. Векторна база не самостійно «розуміє» тексти – вона покладається на модель embeddings, яка працює на рівні машинного навчання. Для нашого проекту як векторизатор планується використати хмарну модель OpenAI (наприклад, *text-embedding-ada-002* або оновлену серію *text-embedding-3*). OpenAI пропонує декілька моделей для отримання embeddings з різною розмірністю та точністю; зокрема, модель *text-embedding-ada-002* генерує вектори розмірності 1536, що стало певним стандартом для багатьох рішень семантичного пошуку. Ці вектори характеризуються тим, що захоплюють контекст і значення слів: наприклад, два різних резюме програмістів, у яких описано схожі навички, дадуть embeddings, близькі один до одного, хоча в самому тексті можуть бути різні слова. Важливо відзначити, що якість embeddings безпосередньо впливає на ефективність RAG: якщо модель добре

“розуміє” тексти, то потрібна інформація майже напевно опиниться серед топ-N найближчих сусідів для запиту.

Пошук за схожістю (semantic similarity search). Коли вектори готові і збережені, база даних повинна забезпечити їх швидке порівняння. Наївний підхід – порахувати відстань від запиту до кожного вектора – не масштабований, тому використовуються алгоритми пошуку приблизних найближчих сусідів (Approximate Nearest Neighbors, ANN). Такі алгоритми будують спеціальні індекси (наприклад, **HNSW** – Hierarchical Navigable Small World, **IVF** – Inverted File Index, **PQ** – Product Quantization тощо) для векторних даних. Ці структури дозволяють значно звузити простір пошуку, жертвуючи мінімальною точністю заради великого виграшу в продуктивності. Векторні бази даних часто пропонують на вибір різні індекси та метрики. Наприклад, в Qdrant або Pinecone можна обрати метрику: **косинусна схожість, евклідова відстань чи скалярний добуток** (dot product) – залежно від характеру embeddings та задачі. Косинусна схожість є популярним вибором для текстових embedding, адже вона нормальнозує довжину векторів і зосереджується на куті між ними, фактично вимірюючи **близькість значення**. Як вже згадувалося, косинус ~1 означає майже паралельні вектори, тобто дуже схожий зміст двох текстів. Інші метрики можуть бути корисні, якщо embeddings не нормалізовані або мають інший сенс. Важливо, що усі ці метрики – математично обґрунтовані функції від компонентів векторів, тому реалізація пошуку часто використовує лінійну алгебру (бібліотеки BLAS) та спеціальні структури даних у пам’яті для швидкодії.

Принцип роботи векторного пошуку можна пояснити на простому рівні: уявімо, що всі вектори – це точки у просторі, і ми хочемо знайти ті точки, що найближче до точки запиту. Якщо вимірність мала (наприклад, 2D чи 3D), задачу можна вирішити побудовою k-d дерева або розбиттям простору на сітку. Але у випадку 1536-вимірного простору інтуїція людини вже не працює, тому використовують евристичні алгоритми. HNSW, наприклад, будує граф, де точки з’єднані з кількома найближчими сусідами на різних «шарах» масштабів; пошук

відбувається шляхом переходів по цьому графу, швидко збігаючись до групи найближчих точок. Результатом є список знайдених ID документів і, можливо, відстаней до них.

Отримавши ID потрібних документів, система RAG завантажує їх текстовий вміст (або заздалегідь зберігає короткі «пасажи» в самому векторному індексі, що теж практикується) і передає на генерацію відповіді, як описано вище.

Висновки

Підсумовуючи, математично-алгоритмічне забезпечення проєкту включає класичні методи (реляційна база даних та її модель) та також новітні технології штучного інтелекту (векторні бази даних, генеративні моделі тощо). Поєднання Django + React є дуже хорошою комбінацією для розробки клієнтської та серверної частини, а RAG-модуль дає потенціал системі бути на крок попереду конкурентів та відповідати всім сучасним тенденціям, при цьому автоматизуючи та покращуючи процеси роботи з даними.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

Основними технологіями розробки стали кілька мов та фреймворків — Django (Python), React (JS), Postgresql, pgvector і звісно html та css, оскільки це веб додаток.

Django Бекенд — високорівневий веб-фреймворк на Python, обраний за можливості швидкої розробки та надійні принципи проектування. Він має багато вбудованих функцій (ORM, панель адміністратора, аутентифікація тощо), які сприяють чистій архітектурі та безпеці, що робить його чудовим вибором для швидкої побудови надійного бекенду. Готова підтримка Django поширених функцій (таких як форми, управління користувачами та запобігання SQL-ін'єкціям або XSS) зменшує необхідність вигадувати велосипед. Читабельність Python і велика екосистема Django (плагіни, бібліотеки) також означають, що майбутні вдосконалення (наприклад, інтеграція чат-бота або нових функцій) можуть бути реалізовані з меншими зусиллями.

React для фронтенду був обраний щоб забезпечити динамічний, адаптивний користувальницький інтерфейс, який буде виглядати сучасно і надасть плавну приємну взаємодію з оновленнями у реальному часі, тобто додаток може швидко відображати зміни без повного перезавантаження сторінки. Крім того, відокремлення фронтенду (React) від бекенду (Django) за допомогою REST API забезпечує гнучкість: інтерфейс можна розробляти і масштабувати незалежно, а майбутні мобільні додатки або інші інтерфейси можуть повторно використовувати той самий API.

Як базу даних було обрано PostgreSQL завдяки її надійності, можливостям та простоті інтеграції з Django. PostgreSQL - це потужна реляційна база даних що при цьому є опенсорс, відома своєю високою надійністю та високою продуктивністю. Розширення pgvector у свою чергу перетворює PostgreSQL на векторну базу даних, готову до використання ШІ, вводячи новий векторний тип даних разом з операторами та індексами для пошуку схожості[18].

Відповідно це створює можливість зберігати дані безпосередньо в Postgres і виконувати запити в межах однієї бази даних. Зберігання як стандартних реляційних даних, так і векторних в одному місці спрощує архітектуру і процес розробки. pgvectorscale - рішення, що базується на pgvector і забезпечує ще вищу продуктивність для пошуку та більш економічне зберігання векторів, що дозволяє PostgreSQL ефективно обробляти великі робочі навантаження [20].

Насправді, бенчмарки показали, що Postgres з pgvector/pgvectorscale може досягти значно менших затримок запитів і вищої пропускної здатності порівняно зі спеціалізованою хмарною векторною БД, причому за меншу частину вартості при самостійному розміщенні [20].

Для створення RESTful API було використано Django REST Framework (DRF), що є розширенням Django, яке спрощує створення Web API, надаючи інструменти для серіалізації, аутентифікації та обробки запитів.

Обраний стек є оптимальним для розробки як невеликих рішень, так і масштабування на рівень корпоративних проектів з величезним об’ємом даних. Дизайн та ORM Django можуть впоратися з більшим навантаженням, додавши кешування або більше робочих процесів за потреби, а збірки React можна оптимізувати через CDN. PostgreSQL може вертикально масштабуватися для досить великих наборів даних, а його розширюваність (за допомогою pgvector/pgvectorscale) означає, що навіть функції штучного інтелекту не зможуть легко перерости базу даних. Якщо використання значно зросте, архітектура підтримує і горизонтальне масштабування - наприклад, фронтенд React може обслуговуватися статично, а Django може бути розгорнутий на декількох серверах додатків за балансувальником навантаження, з керованим Postgres або репліками для читання. Саме такий вибір стеку дозволяє уникнути неймовірної складності та непотрібного коду і функціональності (оверінженірінгу як то кажуть) при початковій розробці, в той же час залишаючи місце для розширення, щоб CRM не зайшла в глухий кут, якщо пізніше буде додано більше користувачів або функцій.

3.2 Вимоги до технічного та програмного забезпечення

Для отримання максимально приємного користувацького досвіду роботи веб застосунку Recrutio необхідне мінімальне технічне та програмне забезпечення. Далі буде визначено вимоги відповідно до архітектури відкритих систем стандарту OSI.

Системне програмне забезпечення (операційні системи):

- Сервер, на якому працює стек Django/PostgreSQL, повинен використовувати сучасну операційну систему, яка підтримує необхідні середовища виконання. Рекомендується використовувати Unix-подібну ОС, таку як Ubuntu Linux LTS, завдяки її стабільності та сумісності з Python і PostgreSQL. Однак стек є кросплатформенним - розробники також можуть запускати його на Windows 10/11 або macOS під час розробки, і навіть Windows Server OS може розмістити його, якщо це необхідно.
- На стороні клієнта не потрібно встановлювати ніякого спеціального програмного забезпечення, окрім стандартного веб-браузера, а це означає, що достатньо будь-якого персонального комп’ютера, на якому працює ОС (Windows, macOS або дистрибутив Linux).
- Єдині вимоги до серверного обладнання - це вимоги для комфортної роботи бази даних та веб-сервера (наприклад, багатоядерний процесор, 8+ ГБ оперативної пам'яті та достатній обсяг дискового простору, залежно від обсягу даних кандидатів та завантажених файлів).

Інструментальне програмне забезпечення (фреймворки, середовища виконання, драйвери):

- Основним середовищем виконання є Python 3 (для Django). Сам фреймворк Django (і пов'язані з ним бібліотеки, такі як Django REST Framework) повинні бути встановлені в середовищі.
- Потрібен сервер баз даних PostgreSQL (версії 13+ або новішої), а також розширення pgvector (і, за бажанням, pgvectorscale), увімкнене в базі даних - ці розширення, можливо, потрібно буде встановити як модулі

PostgreSQL перед використанням. Додаток використовує драйвер бази даних Psycopg (бібліотека Python) для зв'язку Django з PostgreSQL.

- Для фронтенд-збірки під час розробки використовується середовище виконання Node.js (з npm) для компіляції та компонування React-додатку; виробнича збірка React-додатку потім стає статичними файлами (HTML, CSS, JS), які надаються клієнтам.
- Додаткове інструментальне програмне забезпечення включає в себе веб-сервер або сервер додатків: під час розробки використовується вбудований в Django сервер виконання.
- Мережеві драйвери операційної системи та драйвери пристроїв очевидно також повинні функціонувати, щоб забезпечити підключення.
- На стороні клієнта потрібен також веб-браузер (Chrome, Firefox, Edge тощо), через який кінцеві користувачі можуть взаємодіяти з системою.
- Інтернет-з'єднання також обов'язково потрібно

3.3 Опис програмної реалізації

Програмна реалізація є однією з найважливіших задач і подребує дуже серйозного підходу. Розробка власне була поділена на кілька ключових частин. Почалась вона з вибору найкращих технологій, таких як мови програмування та фреймворки або бібліотеки. Спочатку було обрано ключові технології для розробки клієнської, ними стали:

React - це розповсюджена бібліотека JavaScript для створення користувацьких інтерфейсів, зокрема односторінкових додатків (SPA - single page applications). React було обрано через велику кількість факторів, детальніше описаних у попередніх розділах, проте у цьому розділі важливим буде його компонентна архітектура, яка сприяє багаторазовому використанню коду, підтримці та масштабуванню, що буде дуже корисним особливо з ростом системи.

Material UI - популярний фреймворк React UI, який реалізує Material Design від Google. Використаний для швидшого створення приємного візуального стилю та верстки, MUI дозволяє розробляти адаптивні інтерфейси в одному стилі у всій системі, прискорюючи етапи створення прототипів та розробки.

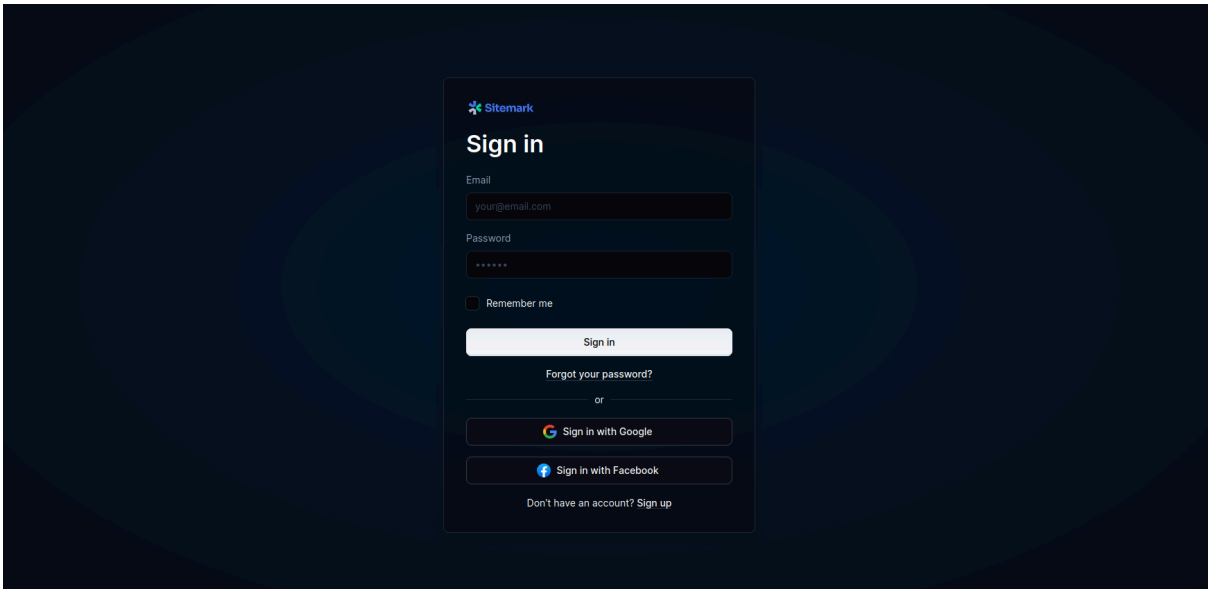

Рис. 3.3.1 Приклад шаблону Material UI

Джерело: https://mui.com/material-ui/getting-started/templates/

Nivo - використовується для візуалізації даних створення інтерактивних діаграм та графіків. Nivo відомий своїми надійними і настроюваними компонентами діаграм, що робить його підходящим вибором для інформаційних панелей або аналітичних переглядів в CRM-додатках.

HTML + CSS - звісно не обійшлось без звичних елементів для веб розробки, а саме мови розмітки та таблиці каскадних стилів.

У загальному фронтенд організовано відповідно до стандартних конвенцій та практик React, що сприяє чіткому розподілу завдань:

- Файли у кореневій папці:
  - index.js: основна точка входу в додаток, що відповідає за рендеринг додатку в DOM.
  - App.js відповідає за маршутизацію
  - theme.js - визначає глобальні стилі, особливо для MUI, щоб на виході отримати дизайн в узгоджених кольорах та темну і світлу теми.
- Data - Містить файли макетів даних (mockData.js, dashboardCustomData.js, mockGeoFeatures.js), які використовуювались під час розробки для імітації реакції бекенда. Це дозволило протестувати фронтент навіть до розробки серверної частини.
- components - містить елементи/компоненти для повторного використання (хедер, графіки тощо)

Компоненти, такі як header.jsx, barchart.jsx, piechart.jsx тощо, слугують атомарними будівельними блоками інтерфейсу. Наприклад, компонент заголовка, навігаційну роль у на різних сторінках додатку де це потрібно, забезпечуючи узгодженість дизайну та стилю, а також візуальну ієрархію у всьому додатку. Абстрагуючись від таких громістких загальних елементів інтерфейсу, така розробка дозволяє досягти більшої ефективності та забезпечує узгоджений дизайн у всій системі.

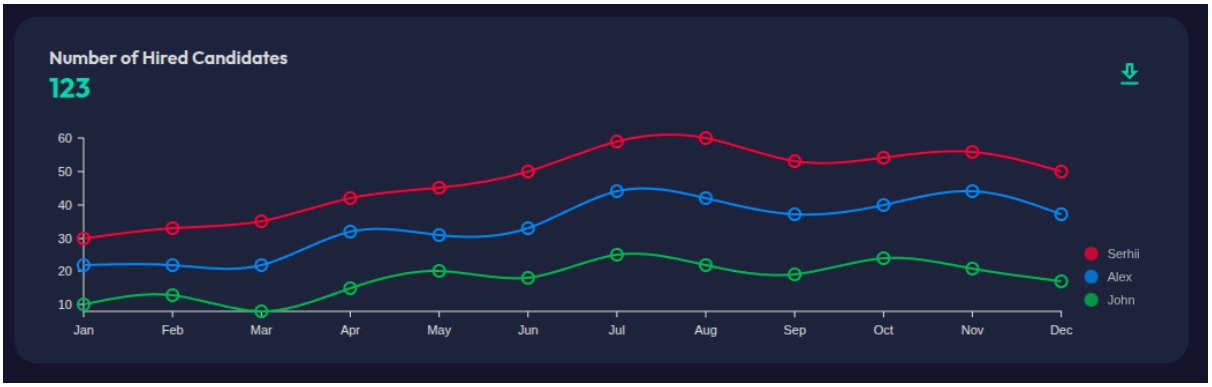

Рис. 3.3.2 Компонент Linechart

Джерело: розроблено автором

- scenes - енкапсулює усі сторінки та містить код для них.

Вищезазначені елементи були створені почерзі один за одним і скомпоновані у комплексний інтерфейс користувача.

Говорячи про компоненти, значна частина з них призначена для візуалізації даних. Включення спеціальних компонентів діаграм, кожен з яких інкапсулює логіку для візуалізації стовпчастих, секторних та лінійних діаграм відповідно, ілюструє орієнтацію додатку на приємний користувацький досвід роботи з даними. Ці компоненти побудовані за допомогою бібліотеки діаграм Nivo. Декларативний синтаксис Nivo та інтеграція з принципами адаптивного дизайну роблять його особливо підходящим вибором для динамічних бізнес-дашбордів, де показники в реальному часі або репрезентативні метрики повинні бути передані інтуїтивно та елегантно. Окрім цього часто повторювані елементи як хедер були також розроблені як компоненти щоб збільшити їхню реюзабіліті.

```
      import { Box, Typography, useTheme } from "@mui/material";
import { tokens } from "../theme";

const Header = ({ title, subtitle }) => {
   const theme = useTheme();
   const colors = tokens(theme.palette.mode);
   return (
       <Box mb="30px">
```

```
            <Typography
                variant="h2"
                color={colors.grey[100]}
                fontWeight="bold"
                sx={{ mb: "5px" }}
            >
                {title}
            </Typography>
            <Typography variant="h5" color={colors.greenAccent[400]}>
                {subtitle}
            </Typography>
        </Box>
    );
};

export default Header;
```

Лістинг 3.3.1 Компонент Header

У той час як компоненти визначають "що" елементів інтерфейсу користувача, сцени забезпечують структурний контекст для того, "де" і "як" ці компоненти збираються в цілісні сторінки, призначені для користувача. Цей архітектурний рівень реалізується через каталог сцен. Сцени задумані як повносторінкові конструкції, які представляють основні функціональні області в CRM, такі як інформаційна панель, огляд кандидатів, проекти тощо. У кожній сцені компоненти складаються і розташовуються так, щоб сформувати семантично багаті і цілеспрямовані інтерфейси. Наприклад, dashboard/index.jsx, функціонує як початкова сторінка, об'єднуючи кілька компонентів візуалізації, щоб надати користувачам загальний огляд активності компанії та показники рекрутерів.

```
      const Topbar = () => {
 const theme = useTheme();
 const colors = tokens(theme.palette.mode);
```

```
const colorMode = useContext(ColorModeContext);

return (
  <Box display="flex" justifyContent="space-between" p={2}>
    {/* SEARCH BAR */}
    <Box
      display="flex"
      backgroundColor={colors.primary[400]}
      borderRadius="3px"
    >
      <InputBase sx={{ ml: 2, flex: 1 }} placeholder="Search" />
      <IconButton type="button" sx={{ p: 1 }}>
        <SearchIcon />
      </IconButton>
    </Box>

    {/* ICONS */}
    <Box display="flex">
      <IconButton onClick={colorMode.toggleColorMode}>
        {theme.palette.mode === "dark" ? (
          <DarkModeOutlinedIcon />
        ) : (
          <LightModeOutlinedIcon />
        )}
      </IconButton>
      <IconButton>
        <NotificationsOutlinedIcon />
      </IconButton>
      <IconButton>
        <SettingsOutlinedIcon />
      </IconButton>
      <IconButton>
        <PersonOutlinedIcon />
      </IconButton>
    </Box>
  </Box>
);
```

```
};

export default Topbar;
```

Лістинг 3.3.2 Глобальна сцена Topbar

Логіка навігації в додатку керується за допомогою React Router, стандартної бібліотеки для маршрутизації на стороні клієнта в екосистемах React. У цьому додатку React Router полегшує відображення окремих URL шляхів до відповідних компонентів сцени.

Важливою детталю даного поекту є візуальний стиль та його узгодженість і брендинг. Вони управляються за допомогою централізованої системи тем, реалізованої у файлі theme.js. Цей файл слугує центром конфігурації для глобальних маркерів дизайну, таких як типографіка, кольорові палітри, інтервали тощо.

Система тем базується на можливостях Material UI, які дозволяють декларативно налаштовувати елементи інтерфейсу користувача у всьому додатку. За допомогою функції createTheme() і ThemeProvider розробники можуть застосовувати власні колірні схеми.

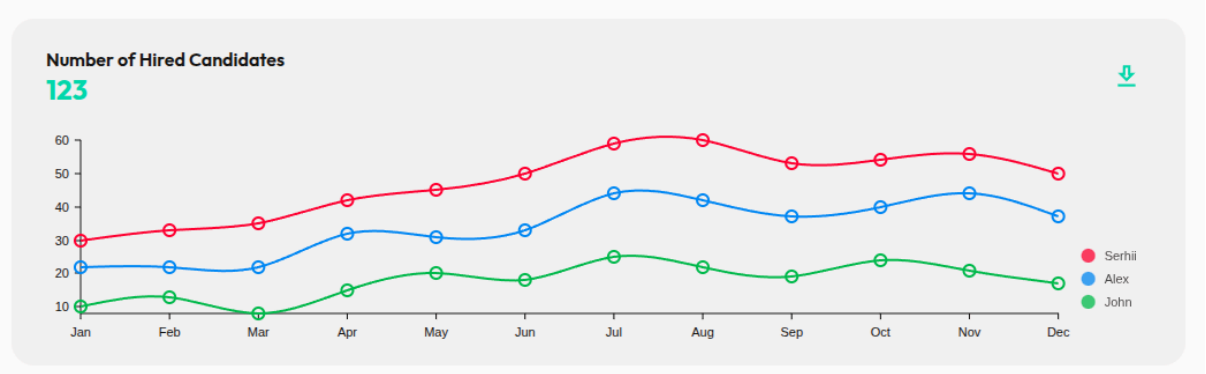

Рис. 3.3.2 Компонент Linechart у свійтлій темі

Джерело: розроблено автором

На ранніх стадіях розробки фронтенду, наприклад коли бекенд відсутній або перебуває на стадії розробки хорошою практикою є використання штучних даних для симуляції реального вигляду системи та тестування. Відвідні файли та mock data були розроблені для досягнення цих цілей.

Після реалізації користувацього інтерфейсу настав час для створення серверної частини, яку реалізовано на Python, з використанням структури схожої до мікросервісної, що часто використовується у Django, проте звісно до межі допоки це доцільно щоб не створювати надто складну структуру для проекту та не мати надто багато непотрібного коду. Також була використана стандартна структура models.py, serializers.py та views.py, характерна для Django REST Framework щоб розробити API.

У загальному Django слідує варіанту парадигми MVC, відомої як архітектура MVT (model view template):

- Модель: Керує структурою даних та абстракцією бази даних за допомогою ORM
- Шаблон: Керує рівнем представлення за допомогою мови шаблонів Django, хоча у моєму випадку цей рівень відсутній (замінений на React)
- Подання (View): Містить бізнес-логіку і виступає посередником між моделлю і шаблоном, також замінений на django-rest-framework views.

ORM Django забезпечує рівень абстракції над базами даних SQL, дозволяючи розробникам визначати моделі як класи Python і взаємодіяти з базою даних, використовуючи синтаксис Python. Вона підтримує складні запити, зв'язки (наприклад, один-до-багатьох, багато-до-багатьох) та міграції, не вимагаючи сирого SQL.

Розташовані у файлі core/models.py, моделі даних (сутності) визначають схему бази даних за допомогою ORM від Django. Зокрема до цих моделей входять як усі необхідні для системи сутності (Користувач, компанія, роль) так і сутності що будуть зберігатись у системі та з якими будуть працювати користувачі (працівник, проект, вакансія тощо).

Ще однією з особливостей Django є його автоматично згенерований інтерфейс адміністратора. З мінімальною конфігурацією Django може створити безпечну і повнофункціональну внутрішню інформаційну панель для керування даними програми, а також для адміністрування всього додатку пізніше.

```
      class Candidate(models.Model):
   company = models.ForeignKey(Company, on_delete=models.CASCADE)
   first_name = models.CharField(max_length=100)
   last_name = models.CharField(max_length=100)
   phone_number = models.CharField(max_length=50)
   email = models.EmailField()
   city = models.CharField(max_length=100)
   address = models.CharField(max_length=255)
   relocation_ready = models.BooleanField(default=False)
   nationality = models.CharField(max_length=100)
   specialty = models.CharField(max_length=100)
   additional_info = models.TextField(blank=True)
   status = models.ForeignKey(Status, on_delete=models.CASCADE,
related_name='candidates')
   housing_status = models.ForeignKey(HousingStatus, on_delete=models.CASCADE,
related_name='candidates')
   recruiter = models.ForeignKey('Recruiter', on_delete=models.CASCADE,
related_name='candidates')
   def __str__(self): return f"{self.first_name} {self.last_name}
({self.company})"

class CandidateDocument(models.Model):
```

Лістинг 3.3.3 Сутність Кандидат

Хоча Джанго пропонує використання вже готової моделі користувача, для того щоб правильно налаштувати дозволи та RBAC (role based access control - контроль доступу на основі ролей) було створено власну кастомну модель, що наслідує вбудовану `AbstractBaseUser` та `PermissionsMixin` для роботи з дозволами.

Для створення API щоб ефективно передавати дані між елементами програми, зокрема серверною та користувацькою частинами використовувався Django REST Framework (DRF). Для перетворення моделей даних (сутностей) в JSON і навпаки використовувались серіалайзери. Вони служать двом важливим цілям:

- Перевірка вхідних даних: Забезпечення цілісності даних під час операцій створення/оновлення.
- Представлення даних: Структурування відповідей API для узгодженості та зрозумілості.

Також, на щастя Django пропонує вбудований захист від поширених веб-уразливостей, в тому числі:

- Міжсайтовий скриптинг (XSS)
- Міжсайтові підробки запитів (CSRF)
- SQL ін'єкції
- Clickjacking

Що означає відсутність потреби звертати на це увагу розробникам та писати це все з нуля. Надійна система автентифікації користувачів також підтримує реєстрацію користувачів, вхід/вихід з системи та хешування паролів, з можливістю розширення моделей користувачів (як зроблено у моєму випадку) та інтеграції сторонніх OAuth.

Для того щоб інтегрувати у майбутньому RAG було зроблено наступні кроки:

- Налаштовано базу даних
  - База даних PostgreSQL налаштована з розширенням pgvector для зберігання векторних включень високої розмірності.
  - Розширення pgvectorscale додано для оптимізації продуктивності пошуку за схожістю, що дозволяє ефективно отримувати релевантні фрагменти даних.
- Вставляння даних до БД:
  - Текст сегментується на керовані фрагменти, які потім перетворюються на векторні вбудовування за допомогою моделей на кшталт OpenAI's text-embedding-3-small.
  - Ці вбудовування разом з відповідними метаданими зберігаються у базі даних PostgreSQL.

- Пошук за схожістю:
  - Коли отримується запит користувача, він перетворюється у векторне відображення з використанням тієї ж моделі.
  - Пошук схожості виконується серед збережених у базі даних вбудовувань, щоб отримати найбільш релевантні фрагменти документа.
- Генерація відповідей:
  - Отримані фрагменти документа компілюються в запит, який потім передається мовній моделі для генерації контекстно-інформованої відповіді.

```
      import logging
import os
from datetime import timedelta
from functools import lru_cache
from typing import Optional

from dotenv import load_dotenv
from pydantic import BaseModel, Field

load_dotenv(dotenv_path="./.env")


def setup_logging():
   """Configure basic logging for the application."""
   logging.basicConfig(
       level=logging.INFO, format="%(asctime)s - %(levelname)s - %(message)s"
   )


class LLMSettings(BaseModel):
   """Base settings for Language Model configurations."""

   temperature: float = 0.0
   max_tokens: Optional[int] = None
```

```
    max_retries: int = 3


class OpenAISettings(LLMSettings):
    """OpenAI-specific settings extending LLMSettings."""

    api_key: str = Field(default_factory=lambda: os.getenv("OPENAI_API_KEY"))
    default_model: str = Field(default="gpt-4o")
    embedding_model: str = Field(default="text-embedding-3-small")


class DatabaseSettings(BaseModel):
    """Database connection settings."""

    service_url: str = Field(default_factory=lambda:
os.getenv("TIMESCALE_SERVICE_URL"))


class VectorStoreSettings(BaseModel):
    """Settings for the VectorStore."""

    table_name: str = "embeddings"
    embedding_dimensions: int = 1536
    time_partition_interval: timedelta = timedelta(days=7)


class Settings(BaseModel):
    """Main settings class combining all sub-settings."""

    openai: OpenAISettings = Field(default_factory=OpenAISettings)
    database: DatabaseSettings = Field(default_factory=DatabaseSettings)
    vector_store: VectorStoreSettings = Field(default_factory=VectorStoreSettin


@lru_cache()
def get_settings() -> Settings:
    """Create and return a cached instance of the Settings."""
```

```
    settings = Settings()
    setup_logging()
    return settings
```

Лістинг 3.3.4 Файл settings.py

Вищенаведений код показує налаштування для роботи з ШІ, зокрема яку модель використовувати , базу даних тощо.

Для того щоб конвертувати дані для зберігання у векторній базі даних вбудовання (embeddings), у файлі vector_store.py створений відповідний функціонал, що використовує модель від OpenAI, а має методи для зберігання всього в БД. У файлі insert_vectors.py ми власне викликаємо данні методи та зберігаємо дані до бази даних.

Висновки

У даному розділі було детально проаналізовано технічне та програмне забезпечення, необхідне для реалізації CRM-системи, включаючи вибір інструментів розробки, вимоги до технічного обладнання, а також процес розробки даної системи, включаючи клієнтську та серверну частини.

ВИСНОВКИ

У даній кваліфікаційній роботі було розроблено веб CRM-систему для малого та середнього бізнесу у сфері рекрутингу та підбору персоналу, орієнтованих на підбір низькокваліфікованого персоналу в Польщі. Проєкт реалізовано з урахуванням як сучасних технологічних стандартів та трендів, так і практичних вимог і висновків базованих на попередньому глибокому аналізі, із використанням фреймворку Django для реалізації серверної частини, бази даних PostgreSQL та інтерфейсу на React. Основною метою розробки було створення інструменту, який би враховував особливості ринку праці Польщі і Європи, забезпечував простоту та зручність користування, аналітику та інтеграцію з елементами штучного інтелекту.

У процесі роботи проведено аналіз предметної області та конкурентних рішень на ринку CRM/ATS систем, що дало змогу сформулювати вимоги до функціоналу, інтерфейсу та архітектури системи. Було спроєктовано інформаційну модель, що охоплює ключові об’єкти домену – кандидатів, вакансії, проєкти, компанії, рекрутерів – із чітким розмежуванням доступу на основі ролей (RBAC), що відповідає сучасним вимогам безпеки.

Математичне та алгоритмічне забезпечення системи включає використання векторних баз даних та Retrieval-Augmented Generation (RAG) для реалізації семантичного пошуку та генерації відповідей на основі корпоративних даних. Це дозволяє значно підвищити ефективність доступу до інформації в CRM та забезпечити новий рівень автоматизації аналітики в рекрутингу.

Розроблена система відповідає потребам сучасного ринку праці у сфері масового підбору персоналу, є масштабованою, безпечною та адаптованою до локального ринку. Проєкт демонструє практичне застосування знань і навичок, здобутих під час навчання, а також підтверджує здатність автора до комплексної розробки сучасних інформаційних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Огляд ринку програмного забезпечення для рекрутингу - People Managing People. URL: https://peoplemanagingpeople.com/employee-lifecycle/recruiting-hiring/recruitment-software-market/#:~:text=Expected%20growth (дата звернення: 01.05.2025)

2. Online Recruitment Software Market Expected to Reach $4 Billion by 2032. URL: alliedmarketresearch.com (дата звернення: 01.05.2025)

3. Applicant Tracking System Statistics (Updated for 2025). URL: selectsoftwarereviews.com (дата звернення: 02.05.2025)

4. Systemy ATS – jak mogą pomóc w rekrutacji pracowników? URL: praca.pl (дата звернення: 05.05.2025)

5. Ranking najlepszych systemów ATS 2024. URL: rankingats.pl (дата звернення: 05.05.2025)

6. Система E-recruiter. URL: erecruiter.pl (дата звернення: 05.05.2025)

7. Система HRAppka. URL: hrappka.pl (дата звернення: 05.05.2025)

8. Teamtailor — Система ATS нового покоління та брендинг роботодавця. URL: https://teamtailor.com (дата звернення: 05.05.2025)

9. Mordor Intelligence Research & Advisory. (2024 , June). Recruitment Software Market Size & Share Analysis - Growth Trends & Forecasts (2025 - 2030). Mordor Intelligence. Retrieved May 17, 2025, from https://www.mordorintelligence.com/industry-reports/recruitment-software-market

10. What is RAG (Retrieval-Augmented Generation)? URL: aws.amazon.com (дата звернення: 10.05.2025)

11. A Guide to Cosine Similarity. URL: timescale.com (дата звернення: 10.05.2025)

12. Vector embeddings. URL: platform.openai.com (дата звернення: 05.05.2025)

13. What is a Vector Database & How Does it Work? Use Cases + Examples. URL: pinecone.io (дата звернення: 10.05.2025)

14. An Introduction to Vector Databases. URL: qdrant.tech (дата звернення: 10.05.2025)

15. Rynek usług HR 2022. URL: https://polskieforumhr.pl/wp-content/uploads/2023/03/RAPORT-_RYNEK_USLUG_HR_2022.pdf (дата звернення: 10.05.2025)

16. Multi-Tenant Database Architecture Patterns Explained. URL: https://www.bytebase.com/blog/multi-tenant-database-architecture-patterns-explained/ (дата звернення: 10.05.2025)

17. Designing a Database for a Recruitment System. URL: vertabelo.com (дата звернення: 07.05.2025)

18. PostgreSQL Extensions: Turning PostgreSQL Into a Vector Database With pgvector. URL: https://www.timescale.com/learn/postgresql-extensions-pgvector (дата звернення: 10.05.2025)

19. Документація та репозиторій pgvectorscale. URL: https://github.com/timescale/pgvectorscale (дата звернення: 11.05.2025)

20. Lott, S. F., & F., L. S. (2014). *Mastering object-oriented python : If you want to master object-oriented python programming this book is a must-have. with 750 code samples and a relaxed tutorial, it's a seamless route to programming python*. Packt Publishing, Limited.

21. *Making queries: Django documentation*. Django Project. (n.d.). https://docs.djangoproject.com/en/5.1/topics/db/queries

22. Martin, R. C. (2018). *Clean architecture: A craftsman's guide to software structure and Design*. Prentice Hall.

23. McLaughlin, B., Pollice, G., & West, D. (2006). *Head first object-oriented analysis and design : A brain friendly guide to ooa&d*. O'Reilly Media, Incorporated.