МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
УКРАЇНИ «КПІ імені Ігоря Сікорського»
факультет «Інформатики та обчислювальної техніки»
кафедра «Автоматики та управління в технічних системах»

КУРСОВА РОБОТА
по курсу «Бази даних-2»
на тему: «База даних Київської обласної дирекції служби зайнятості населення»

Студента III курсу ІТ-51 групи
спеціальності «Програмна інженерія»
Бессмертного Романа Сергійовича

Керівник:

_____

Національна шкала _____

Кількість балів: _____Оцінка: ECTS _____

Члени комісії: _____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

м. Київ – 2017 рік

ЗМІСТ

ВСТУП 3

1 ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ 4

1.1 Аналіз інформаційних потреб та визначення предметної області дослідження 4

1.2 Архітектура та засоби реалізації бази даних Київської обласної дирекції служби зайнятості населення 10

1.3 Опис Microsoft SQL Server 13

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ 14

2.1 Аналіз інформаційних та датологічних процесів 14

2.2 Проектування структури бази даних Київської обласної дирекції служби зайнятості населення 15

3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ 23

3.1 Проектування інтерфейсу обробки даних 23

3.2 Фізична реалізація бази даних 24

3.3 Виконання основних запитів 29

3.4 Підтримка міграцій бази даних 37

4 АДМІНІСТРУВАННЯ БАЗИ ДАНИХ 39

4.1 Порядок налаштування сервера 39

4.1 Налаштування прав доступу 41

ВИСНОВКИ 42

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 43

ДОДАТОК А 46

ВСТУП

На всій планеті веб-сайти пошуку роботи є широко відомою частиною інтернету. Великі гравці, такі як «Indeed» та «Monster», перетворили пошук роботи та вербування в справжню онлайн-індустрію.

Люди прагнуть економити час, використовуючи технологічні нововведення; Портал пошуку роботи є ще одним способом працювати розумніше, не важче. Шукачі роботи та компанії усвідомлюють цінність роботи в Інтернеті: вони отримують кращий результат при більшій швидкості та зниженні витрат.

Сфера роботи порталу зараз досить стабілізована, принаймні стосовно обсягів руху. Люди в пошуках роботи використовують ці портали для пошуку позицій у багатьох галузях промисловості, виходячи за межі IT до таких галузей, як інжиніринг, продаж, виробництво та фінансові послуги. Тим не менше, вони отримують жорстку конкуренцію з боку соціальних та професійних мереж, таких як «LinkedIn». Але на ринку ще існують потенційні можливості, такі як розширення їх проникнення в сільські райони та менші міста.

Метою даної курсової роботи є розробка бази даних Київської обласної дирекції служби зайнятості населення. База даних має надавати дані для управління користувачами, створення профілів та резюме, а також для розміщення, пошуку та подання заявок на роботу.

1 ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1 Аналіз інформаційних потреб та визначення предметної області дослідження

Для висунення функціональних та не функціональних вимог слід чітко сформувати предметну область, від якої можна буде відштовхуватись і при розробці бази даних та програмного засобу. Опишемо предметну область.

І роботодавці, і шукачі роботи очікують наступні функціональні можливості від веб-сайту:

- Люди можуть зареєструватися як шукачі роботи, створювати свої профілі та шукати роботу, відповідну їх сукупності навичок.
- Користувачі можуть завантажувати свої існуючі резюме. Якщо їх немає, вони повинні мати змогу заповнити форму та отримати відповідне резюме.
- Люди можуть звертатися безпосередньо до існуючих вакансій.
- Компанії можуть реєструвати, розміщувати роботи та шукати профілі шукачів роботи.
- Кілька представників компанії повинні мати можливість зареєструвати та розміщувати роботи.
- Представники компанії можуть переглядати список претендентів на роботу та можуть зв'язатися з ними, ініціювати співбесіду або виконувати інші дії, пов'язані з їх посадою.
- Зареєстровані користувачі повинні мати можливість шукати роботу та відфільтрувати результати на основі місця розташування, необхідних навичок, заробітної плати, рівня досвіду тощо.

Покажемо варіанти використання за допомогою діаграми Use Case:

На рисунку 1.1 зображено Use Case діаграми веб-сайту Київської обласної дирекції служби зайнятості населення.

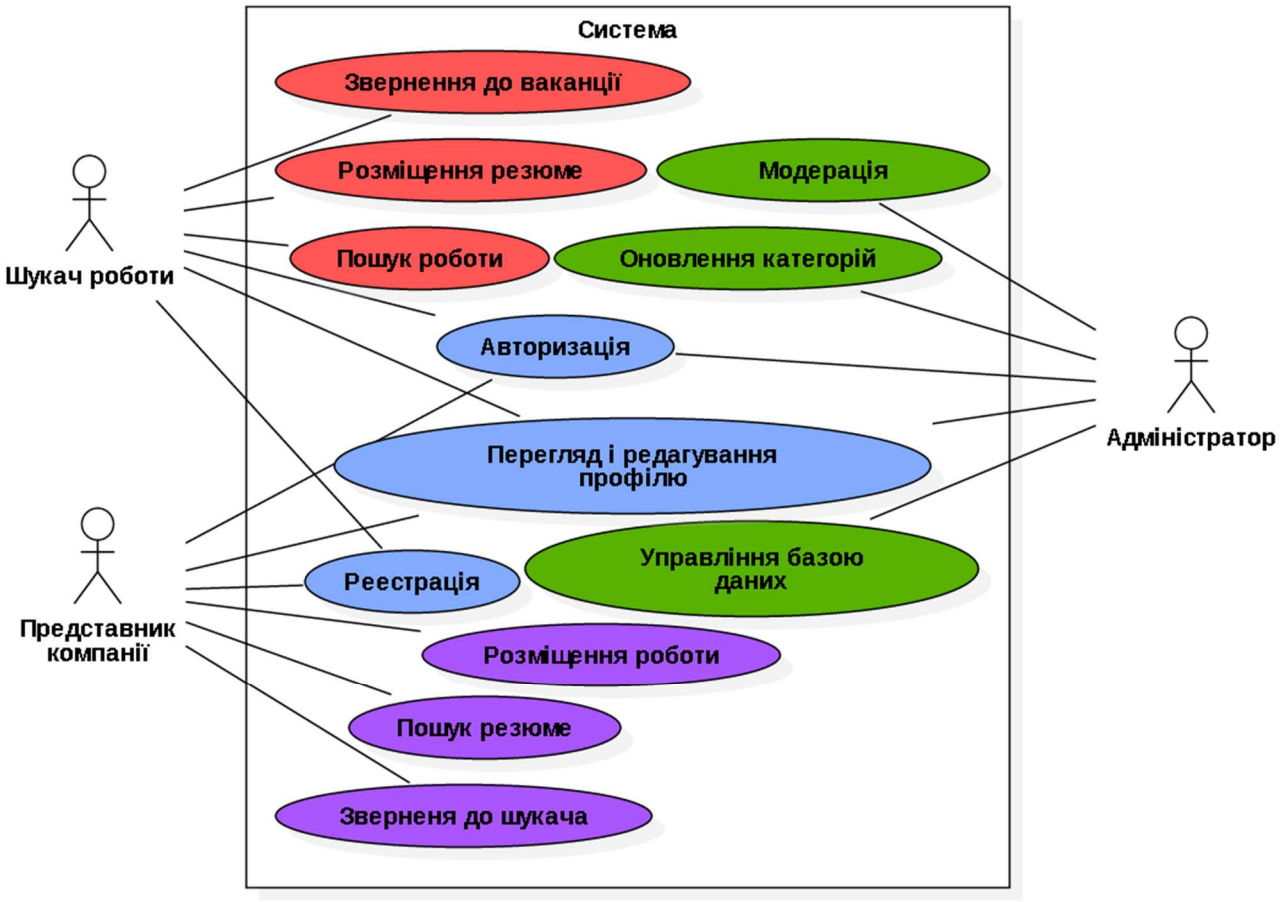

Рисунок 1.1 – Use Case діаграма веб-сайту Київської обласної дирекції служби зайнятості населення.

Опишемо наведену діаграму більш детально. Першою основною функціональною категорією є управління користувачами - як саме портал обробляє користувачів, тобто шукачів роботи, робітників HR, а також незалежних або допоміжних рекрутерів.

Користувачі повинні зареєструватися перш ніж вони зможуть користуватися порталом. Для реалізації реєстрації та авторизації потрібно зберігати основні дані – логін і пароль - незалежно від типу користувачів. Усі інше дані користувачів, такі як телефон та місце проживання, також заносяться при реєстрації. Потрібно зауважити, що оскільки певна інформація, наприклад освіта, залежать від типу користувача, процес реєстрації також змінюється в залежності від типу.

Якщо користувач вже зареєстрований, йому потрібно авторизуватися. Авторизація включає в себе введення логіну та паролю, і перевірити їх валідність за допомогою збережених раніше відповідних даних.

Незалежно від типу, користувач може переглянути і редагувати свій профіль. Тип користувача визначає, яку додаткову інформацію потрібно зберегти, і, відповідно, яку додаткову інформацію користувач може редагувати.

Як ми можемо побачити на рисунку 1.1, усі інші варіанти використання залежать від типу користувача. Розглянемо їх окремо відповідно до типів.

На рисунку 1.2 зображено діаграму активності для шукача роботи.

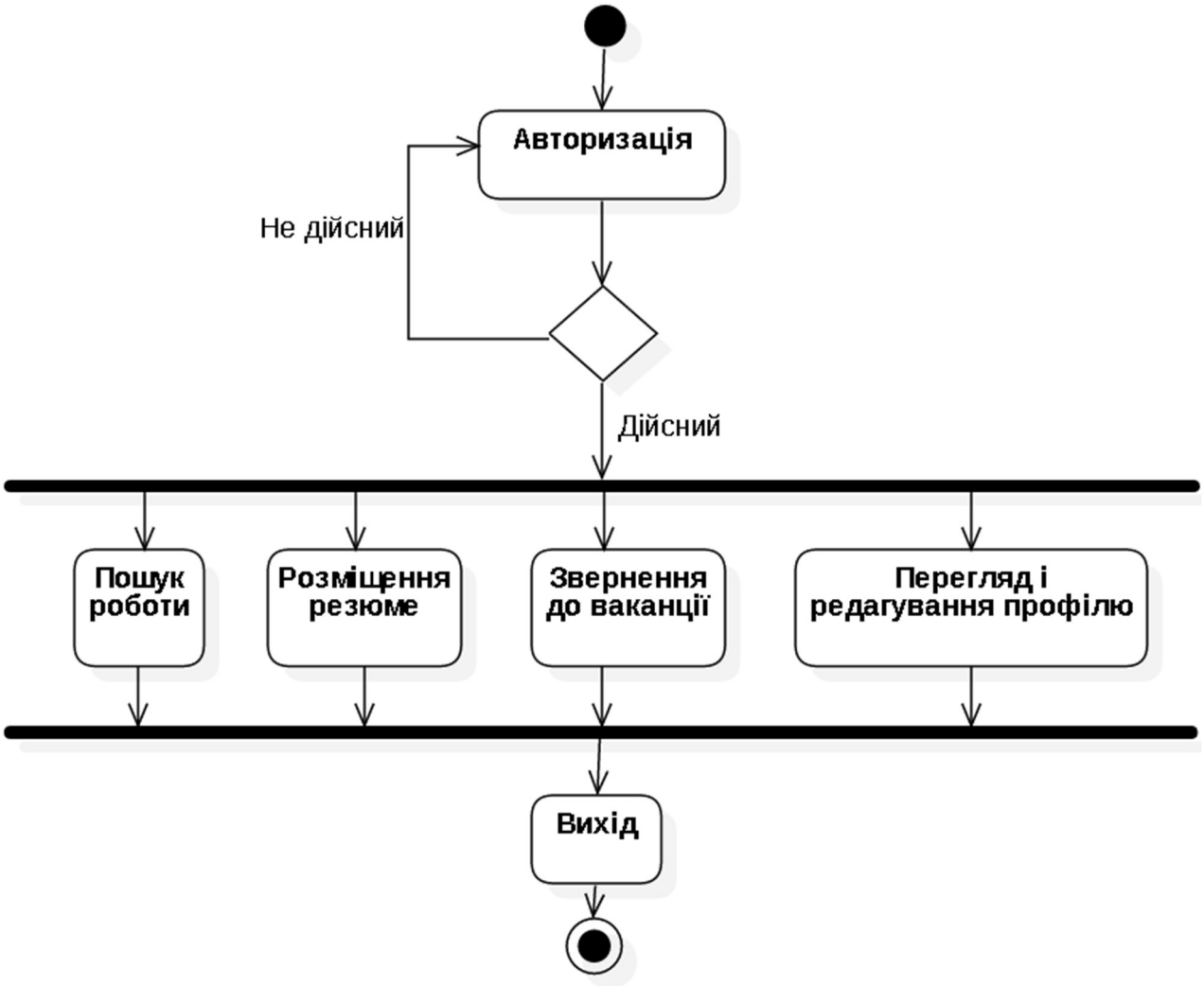

Рисунок 1.2 – Діаграма активності для шукача роботи веб-сайту Київської обласної дирекції служби зайнятості населення.

Відповідно до рисунку 1.2, унікальними для шукача роботи є пошук роботи, розміщення резюме та звернення до існуючої вакансії. Розглянемо ці дії більш детально.

Виконуючи пошук роботи, користувач хоче бачити вакації в такому вигляді, що дозволяє знайти роботу, що якомога краще відповідає його потребам, за найменшої витрати часу і сил. Для досягнення цієї мети, необхідно надати користувачу можливість переглядати список існуючих вакансій, а також можливість відфільтрувати та відсортувати доступні дані відповідно до потреб шукача роботи - місця розташування, необхідних навичок, заробітної плати, рівня досвіду тощо. Викладання повної інформації про кожну роботу у список не є доцільним. Кращим варіантом буде використання лише основних даних для процесу пошуку. Тоді якщо користувач зацікавлений певною роботою, він має мати можливість переглянути детальну інформацію.

Резюме – найбільш критична частина системи. Якщо сайт не зможе зберегти якомога більше інформації про шукачів роботи, рекрутерам буде важко оптимально знаходити кандидатів. Необхідно зберігати детальну інформацію про людину, її освіту, спеціалізацію, досвід роботи. Будемо вважати, що шукач роботи бажає влаштуватись на чітко визначену спеціальність. Це дозволить скоротити кількість інформації, і підвищити її якість. Наприклад, для одного інтерв’ю необхідно вказувати тільки ту освіту і досвід роботи, що чітко мають відношення до бажаної роботи.

Якщо користувач зацікавлений вакансією і бажає встановити зв’язок, він має мати можливість відправити резюме представникам компанії, які, в свою чергу, будуть відповідно мати можливість зв'язатися з ним, ініціювати співбесіду або виконувати інші дії, пов'язані з їх посадою.

Ці та інші дії представникам компанії розглянемо більш детально на діаграмі активності представників компанії, зображеної на рисунку 1.3.

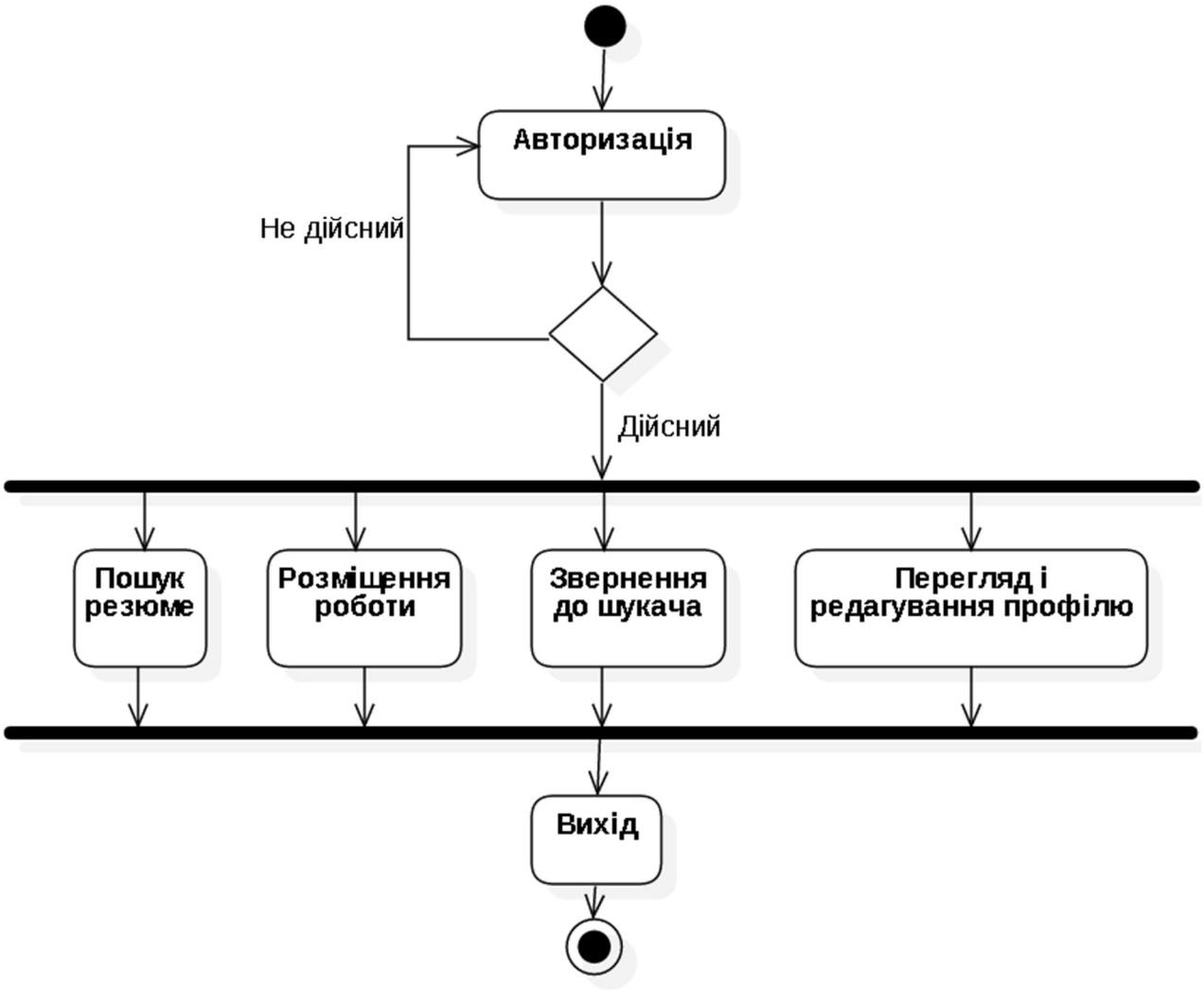

Рисунок 1.3 – Діаграма активності для представника компанії веб-сайту Київської обласної дирекції служби зайнятості населення.

Виконуючи пошук потенційного робітника, представник компанії хоче бачити резюме в такому вигляді, що дозволяє знайти людину, що якомога краще відповідає його потребам, за найменшої витрати часу і сил. Для досягнення цієї мети, необхідно надати представнику можливість переглядати список існуючих резюме, а також можливість відфільтрувати та відсортувати доступні дані відповідно до потреб компанії - місця розташування, освіти, спеціалізації, досвіду роботи тощо. Викладання повної інформації про кожне резюме у список не є доцільним. Кращим варіантом буде використання лише основних даних для процесу пошуку. Тоді якщо представник зацікавлений певним користувачем, він має мати можливість переглянути детальну інформацію.

Розміщення, і, відповідно, пошук робіт – головна особливість порталу. Необхідно зберігати детальну інформацію про тип роботи, необхідну освіту, необхідну сукупність навичок, місце роботи. Необхідно пам'ятати, що не вся інформація має бути доступною для шукачів роботи. Наприклад, представник компанії може забажати використовувати терміни виду «глобальна автомобільна компанія», «IT-компанія що базується в Києві» тощо.

Якщо рекрутер знайшов відповідне резюме і бажає встановити зв’язок, він має мати можливість зв'язатися з ним, ініціювати співбесіду або виконувати інші дії, пов'язані з їх посадою.

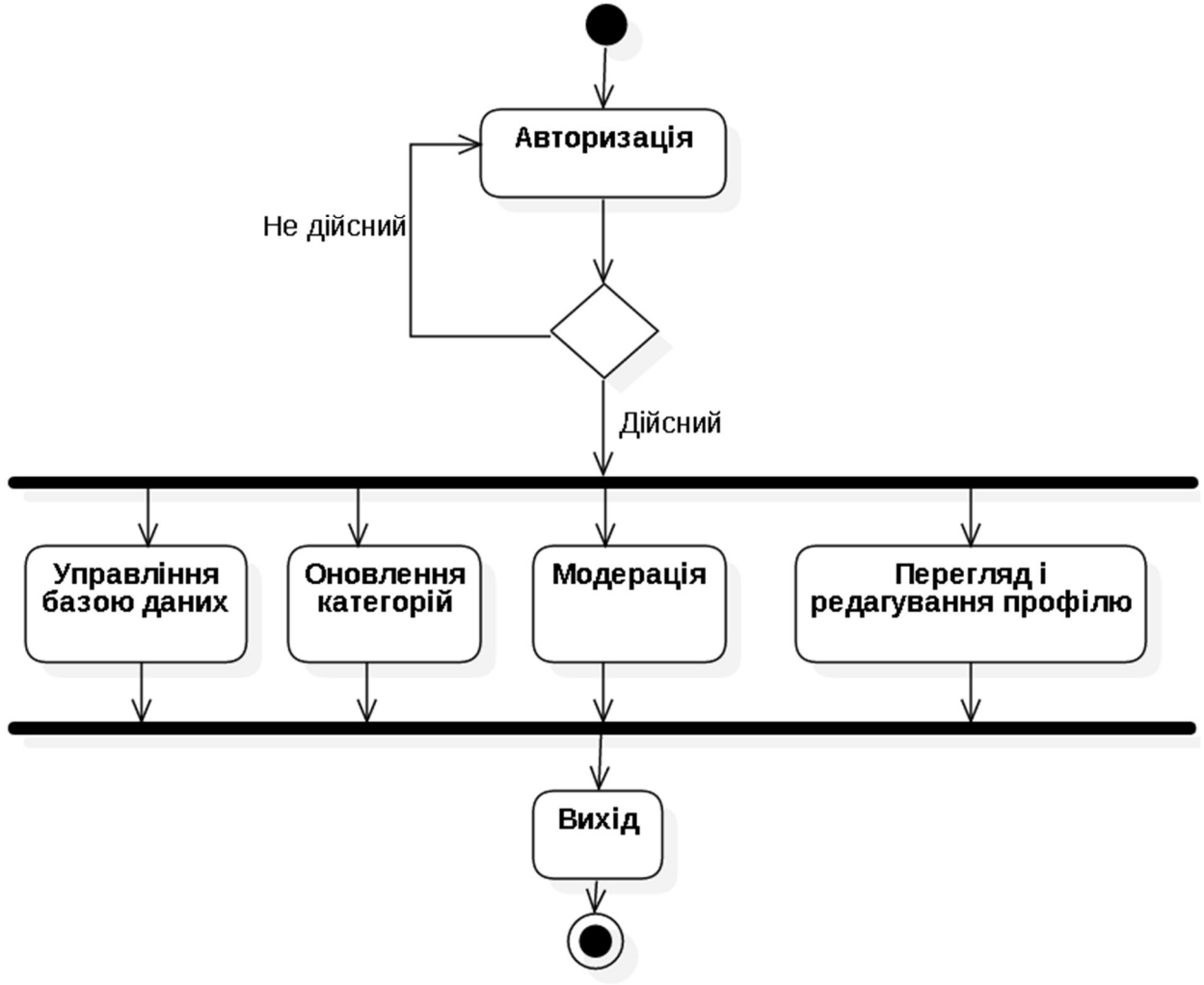

Рисунок 1.4 – Діаграма активності для адміністратора веб-сайту Київської обласної дирекції служби зайнятості населення.

Адміністратор сайту стежить за працездатністю сервера (серверного устаткування і програм), на якому знаходиться сайт. Адміністратор несе відповідальність за мережеву безпеку. В деяких випадках він стає веб-майстером і може займатися розкручуванням, вести статистику відвідуваності, виконувати обов'язки контент-менеджера, стежачи за своєчасним оновленням інформації.

У нашому випадку ми можемо виділити три основні дії, які виконує адміністратор веб-порталу – управління базою даних, оновлення категорій та модерація.

Задля забезпечення працездатності сервера адміністратор має доступ до усіх записів бази даних, і може редагувати їх як за допомогою інтерфейсу, так і напряму через SQL запити.

Адміністратор має слідкувати за актуальністю категорій – види роботи, можливі місця роботи, тощо. Таким чином виконується своєчасне оновлення інформації, що знаходиться на порталі.

Необхідно слідкувати за відповідністю поведінки користувачів правилам сайту та чинному законодавству, тому адміністратор має можливість переглядати активність користувачів і відстежувати виконання правил.

1.2 Архітектура та засоби реалізації бази даних Київської обласної дирекції служби зайнятості населення

Основною операцією для розроблюваної бази даних є запис нових профілів та резюме. Так, за достатнього трафіку, нові дані створюються майже неперервно. На кожну операцію проводиться звернення до бази даних.

Проте, немає необхідності постійного зв’язку апаратної частини із базою даних. Апаратна частина буде проводити усі операції через сервер, який вже, в свою чергу, буде взаємодіяти із базою даних. Тому, вибір СУБД, найбільш сумісний із серверною частиною, є доцільним.

Серверна частина буде реалізована як Web API, та може бути розгорнутою на платформі під керівництвом операційної системи Windows.

Порівняємо системи управління базами даних MS SQL SERVER та Oracle. Результати порівняння занесемо до таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика СУБД Microsoft SQL Serve та Oracle

Характеристика	Microsoft SQL Server	Oracle
Адміністративне керування	Відмінно	Відмінно
Графічні інструменти	Відмінно	Добре
Простота обслуговування	Відмінно	Відмінно
Механізм данних	Добре	Відмінно
Робота с декількома ЦП	Задовільно	Відмінно
Функції з'єднання і вибір індексів	Відмінно	Відмінно
Одночасний доступ декількох користувачів	Добре	Відмінно
Обробка даних мультимедіа	Не задовільняє	Відмінно
Подключення к Web	Відмінно	Відмінно
Повнотекстовий пошук	Добре	Відмінно
Функціональна сумісність	Добре	Добре
Інтеграція з іншими СУБД	Добре	Добре
Єдина реєстрація	Добре	Добре
Робота під керівництвом ОС	Задовільно	Добре
Можливості програмування	Задовільно	Відмінно
Процедури, що зберігаються та тригери	Добре	Відмінно
Вбудована мова програмування	Задовільно	Відмінно
Побудова БД	Відмінно	Відмінно
Мова SQL	Відмінно	Відмінно
Підтримка об'єктно-орієнтованої парадигми	Задовільно	Відмінно
Робота у режимі віддаленого доступу	Відмінно	Відмінно
Тиражування	Відмінно	Відмінно
Розподілена обробка транзакцій	Відмінно	Відмінно
Дистанційне адміністрування	Добре	Відмінно
Організація сховищ даних і підготовка звітів	Відмінно	Відмінно
Засоби завантаження	Відмінно	Відмінно
Засоби аналізу	Відмінно	Відмінно

Також, порівняємо системи управління базами даних за вимогами технічного завдання.

Таблиця 1.2 – Порівняння СУБД за вимогами технічного завдання

Необходимые требования	MS SQL Server	Oracle
Локалізація інтерфейсу користувача, можливість побудови і сортування полів БД, що містять символи кирилиці	+	+
Підтримка структури відносних даних	+	+
Підтримка технології клієнт/сервер	+	+
Підтримка багатопроцесорної архітектури	+	+
Підтримка кластерної архітектури	+	+
Наявність засобів для створення індексів і кластерів для підвищення ефективності використання даних	+	+
Відновлення баз даних із використанням журнала трансакцій	+	+
Механізм блокування трансакцій під час запису або на рівні сторінок	+	+
підтримка ANSI SQL	+	+
Підтримка стандарту SQL-3 (новое название – SQL99)	+	+
підтримка ODBC	+	+
Контроль цілісності БД	+	+
Підтримка утиліт резервування БД	+	+
Імпорт/експорт таблиць БД	+	+
Сумісність з ОС модулів користувача та сервера	+	+
Підтримка визначених мережевих протоколів	+	+
Наявність графічного інтерфейсу для адміністраторів БД	+	+
Контроль доступу до даних. Аутентифікація засобами СУБД	+	+
Централізоване керування користувачами	+	+
Наявність оптимізатора запитів для оптимізації планів виконання	+	+
Підтримка великих двійкових об’єктів (BLOB)	+	+
Підтримка OLAP технологій, спеціалізованих засобів OLAP-аналізу	+	+
Підтримка протоколів VI SAN (Virtual Interface System Area Network)	+	+
Відлагоджений механізм реплікації даних	+	+
Підтримка служби єдиного каталогу	+	+

З порівняльної таблиці бачимо, що обидві СУБД добре вправляються із необхідними задачами. Хоча Oracle в деяких моментах є кращим за Microsoft SQL Server, остання легша в користуванні та простіше інтегрується з ASP Web API, тому для реалізації бази даних буде обрано Microsoft SQL Server.

1.3 Опис Microsoft SQL Server

Це система керування базами даних, розроблена корпорацією Microsoft. Основний використовуваний мову запитів - Transact-SQL, створений Microsoft. Transact-SQL є реалізацією стандарту по структурованій мові запитів (SQL) з розширеннями. Використовується для роботи з невеликими і середніми за розміром базами даних.

Завдяки інтеграції з середовищем розробки Microsoft Visual Studio SQL Server дозволяє розробляти керовані додатками дані з широкими можливостями, які забезпечують поліпшену безпеку сховищ і швидке розгортання.

У своєму складі система має засоби створення баз даних, роботи з інформацією баз даних, перенесення даних з інших систем і в інші системи, резервного копіювання та відновлення даних, розвинену систему транзакцій, систему реплікації даних, реляційну підсистему для аналізу, оптимізації та виконання запитів клієнтів, систему безпеки для управління правами доступу до об'єктів бази даних та ін. Система не містить засобів розробки клієнтських додатків.

SQL Server будується за структурою таблиці на основі рядків, а також з'єднує пов'язані елементи даних у різних таблицях, уникаючи необхідності резервного зберігання даних у кількох місцях в межах бази даних. Реляційна модель також забезпечує дотримання принципів атомності, узгодженості, ізоляції та довговічності - загальновідомі як властивості ACID і призначені для гарантування того, що транзакції бази даних надійно обробляються.

Основним компонентом Microsoft SQL Server є SQL Server Database Engine, який керує збереженням, обробкою даних та безпекою даних. Вона включає в себе реляційний движок, який обробляє команди та запити, а також двигун зберігання, який керує файлами, таблицями, сторінками, індексами, буферами даних та транзакціями. Збережені процедури, активатори, перегляди та інші об'єкти бази даних також створюються та виконуються за допомогою двигуна бази даних.

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

2.1 Аналіз інформаційних та датологічних процесів

Основні потоки даних, які наявні у розроблюваній системі були описані у попередньому пункті при описанні предметної області системи. Найбільш основні деталі були показані за допомогою діаграм UML.

У цьому пункті більшу увагу приділимо опису датологічних процесів та поступово будемо зв’язувати їх із майбутньою базою даних.

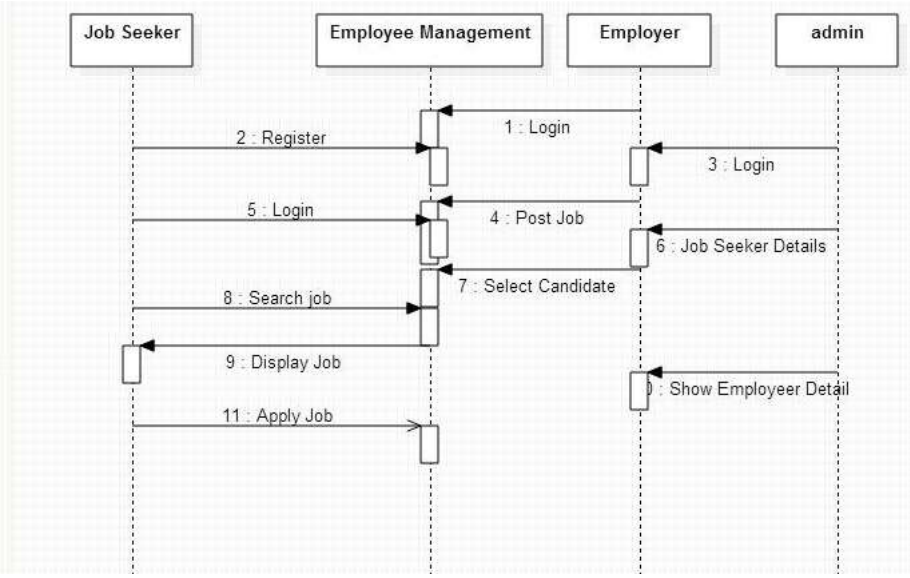

Рисунок 2.1 – Основні інформаційні стани системи та послідовність відправлених повідомлень

Опишемо основний потік даних, який буде проходити у системі.

Коли користувач заходить на портал, він має можливість зареєструватися, увійти або виконати простий пошук наявних робіт.

Для входу користувач відправляє свою електронну пошту та пароль, система валідує користувача та надає результат входу.

Увійшовши в систему, користувач може виконати різні дії в залежності від типу користувача.

Шукач роботи може використати пошук за певними критеріями. Система отримує ці критерії та повертає результат пошуку. Система показує результат пошуку.

Роботодавець заповнює усі відповідні деталі нової роботи та надсилає ці дані системі. Ці дані заносяться до бази даних, і будуть використані при пошуку роботи.

2.2 Проектування структури бази даних Київської обласної дирекції служби зайнятості населення

Аналізуючи предметну область, виведемо необхідні дані, які на необхідно зберігати. Обрані дані для збереження надалі будуть прес давлені у схемі бази даних для проектуємої системи.

Після розгляду вищезазначених вимог було виділено три широкі функціональні категорії:

- Керування користувачами - як портал управляє користувачами, тобто шукачами роботи, робітників HR, а також незалежними або консалтинговими вербувальниками. (Для цілей даної моделі окремі представники HR та незалежні або консалтингові вербувальники розглядаються як компанії, принаймні з точки зору використання порталу.)
- Створення профілів - як портал дозволяє шукачам роботи та організаціям створювати профілі та резюме.
- Публікація та пошук робочих місць - як портал полегшує процес публікації, пошуку та подання заявки на роботу.

Зобразимо логічну структуру бази даних за допомогою Entity Relationship діаграми (рисунок 2.2).

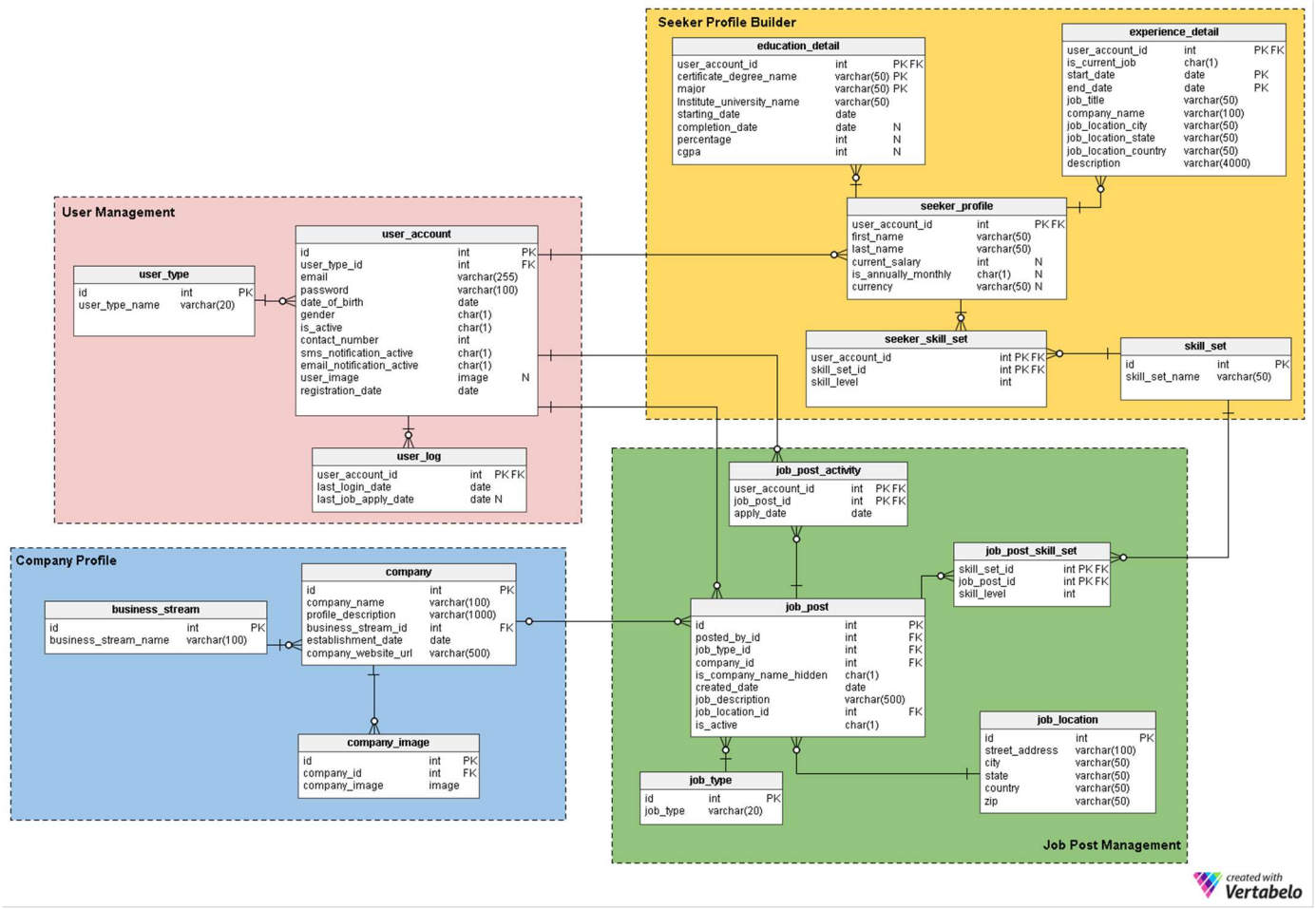

Рисунок 2.2 – Логічна структура бази даних, описана за допомогою Entity Relationship діаграми

Давайте розглянемо кожну з цих областей окремо.

- Керування користувачами

Існує, перш за все, два типи користувачів інтернет-порталу: індивідуальні шукачі роботи та HR-рекрутери (або незалежні консультанти з підбору персоналу). Давайте створимо таблицю з ім'ям user_type для зберігання цих записів. Для початку він буде мати два рекорди - одне для шукачів роботи, а інше для рекрутерів. (Ми завжди можемо створювати додаткові типи записів за потребою.)

Користувачі повинні зареєструватися, перш ніж користуватись порталом. У таблиці user_account зберігаються основні дані облікового запису.

У таблиці user_account є такі стовпці:

– id – являє собою як первинний ключ таблиці, так і унікальний ідентифікатор для кожного користувача. Цей ідентифікатор буде використаний іншими таблицями в моделі даних.

– user_type_id – визначає, чи є користувач шукачем роботи або рекрутером.

– email - цей стовпець містить електронну адресу користувача. Він діє як альтернативний ідентифікатор користувача для порталу.

– пароль - зберігає зашифрований пароль облікового запису (створеного користувачами під час реєстрації).

– date_of_birth та gender - ці стовпці містять дату народження та стать користувача.

– is_active - спочатку цей стовпець буде "Y", але користувачі можуть встановити свій профіль неактивним або "N". Цей стовпець зберігає їхній вибір.

– contact_number - це номер телефону (зазвичай мобільний), що надається під час реєстрації. Користувачі можуть отримувати SMS (текстові) сповіщення на цей номер. Це може бути той самий номер (чи ні), що шукачі роботи вказують у своєму профілі або резюме.

– sms_notification_active та email_notification_active - ці стовпці зберігають налаштування користувачів щодо одержання сповіщень через текст та / або електронну пошту.

– user_image - це атрибут типу BLOB, в якому зберігається зображення профілю кожного користувача. Оскільки цей портал допускає лише одне зображення профілю для кожного користувача, має сенс його зберігати тут.

– registration_date - цей стовпець зберігає запис про те, коли користувач зареєструвався на сайті.

Ми створимо ще одну таблицю - user_log, в якій зберігається запис останньої дати входу користувачів та їх останню дату подання заявки на роботу. Є багато функцій, які можуть бути побудовані з цих знань. Наприклад, ми можемо використовувати цю інформацію, щоб відповісти на запитання «Чи активно користувач X шукає роботу?» Якщо так, то їм може запропонувати продукт для створення ефективного резюме. Користувачі, які не активно шукають роботу, не отримають такої пропозиції.

- Створення профілів

Ми можемо додатково поділити цей розділ на дві області: профіль компанії або організації, а також профілі шукачів роботи.

Профіль компанії.

Зазвичай HR команди формують профілі компаній, вносячи деталі про свою організацію та зображення своїх офісів, будівель тощо. Їх головна мета - залучення хороших талантів. Коли рекрутери реєструються на порталі, вони також можуть створювати профілі своїх компаній (або їх особистий бренд, якщо вони незалежні), надаючи деякі основні відомості про те, як довго вони займаються бізнесом, їх місцезнаходження та основний бізнес-потік (наприклад, виробництво, IT-послуги, фінансові та інші).

Портал дозволяє HR та консультуючим рекрутерам завантажувати стільки зображень, скільки їм подобається (на відміну від тих, хто шукає роботу, вони можуть завантажувати лише одне). Тому ми створили таблицю company_image для зберігання кількох зображень для кожного облікового запису рекрутера. Стовпець company_id у цій таблиці є зовнішнім ключем, який посилається на унікальний ідентифікатор, який використовується в таблиці company.

У таблиці company є такі стовпці:

– id - первинний ключ цієї таблиці, також використовується для унікального визначення компаній.

– company_name - як випливає з назви стовпця, це - юридична назва компанії.

– profile_description - короткий опис кожної компанії.

– business_stream_id - у цьому стовпчику показано, який бізнес-потік належить компанії. Наприклад, компанія з розвідки нафти та газу може найняти інженерів з інформаційних технологій, але основним напрямком їх діяльності залишається "Нафта та газ".

– establishment_date - у цьому стовпчику повідомляється, скільки років компанії.

– company_website_url - це обов'язковий (не підлягає скасуванню) стовпець. Він вказує на офіційний веб-сайт компанії, тому шукачі роботи можуть отримати більше інформації.

Нарешті, таблиця business_stream має лише два атрибути: ідентифікатор, який є основним ключем для цієї таблиці, а також опис основного бізнес-потоку компанії (business_stream_name).

Профіль шукачів роботи.

Резюме – найбільш критична частина системи. Якщо сайт не зможе зберегти якомога більше інформації про шукачів роботи, рекрутерам буде важко оптимально знаходити кандидатів.

Таблиця seeker_profile містить додаткові відомості, які не були зроблені під час процесу реєстрації. Вона містить такі поля:

– user_account_id - цей стовпець передається з таблиці user_account, і він діє як основний ключ для цієї таблиці. Це гарантує тільки одного профіль для кожного шукача роботи.

– first_name and last_name - ці стовпці містять імена та прізвища шукачів роботи.

– current_salary - Цей атрибут містить поточну зарплату шукача роботи. Це значення - nullable, оскільки люди, можливо, не хочуть його розкривати.

– is_annually_monthly - визначає, чи зарплата визначається на рік або на місяць.

– currency - зберігає валюту заробітної плати.

У таблиці "education_detail" зберігаються історія освіти шукачів роботи, принаймні яку вони надають. Він має складний первинний ключ, що складається з user_account_id, name_ certificate_degree_name і major стовпчиків. Це гарантує, що користувачі вводять лише один запис за кожним ступенем або сертифікатом. Таблиця містить такі атрибути:

– user_account_id - цей стовпець передається з таблиці user_account і слугує основним ключем для цієї таблиці.

– certificate_degree_name - це сертифікат або тип ступеня; наприклад, середня школа, вища середня, аспірантура, аспірантура або професійний сертифікат.

– major - Ця колонка містить основний курс навчання за сертифікатом або ступенем - наприклад, ступінь бакалавра з фахом з комп'ютерних наук.

– institute_university_name - це інститут, школа або університет, у якому користувач отримав ступень чи сертифікат.

– start_date - цей атрибут зберігає дату, коли користувач був прийнятий у навчальну програму.

– completion_date - це дата отримання ступеня чи сертифікату. Однак цей атрибут є nullable; люди можуть продовжувати виконувати свою програму, коли шукають роботу, або вони взагалі можуть випасти з програми.

– mark – цей стовпчик містить загальну середню оцінку, що досягається користувачами на курсах їх ступеня або сертифікату.

У таблиці experience_detail зберігаються записи про минулий та поточний професійний досвід користувачів. Він містить такі важливі стовпці:

– user_account_id - цей стовпець передається з таблиці user_account і є основним ключем для цієї таблиці.

– is_current_job - це стовпчик-індикатор, який вказує поточну роботу користувача. Цей стовпець також відіграє важливу роль у виведенні поточних розташувань користувачів і про час їхньої поточної позиції.

– start_date – зберігає дату, коли користувач починає роботу.

– end_date - зберігає дату, коли користувач закінчує роботу.

– job_title - містить інформацію про ролі роботи користувача.

– company_name - атрибут містить відповідне ім'я компанії, пов'язане з роботою.

– job_location_city - місто, де була розташована робота.

– job_location_state - область, де була розташована робота.

– job_location_country - країна, де ця робота була розташована.

– description - у цьому стовпчику зберігаються відомості про ролі роботи та обов'язки, проблеми та досягнення.

Шукачі роботи можуть володіти декількома навичками. Щоб вести облік усіх цих наборів навичок, ми створимо таблицю seeker_skill_set. Колонки:

– user_account_id - цей стовпець передається з таблиці user_account і є основним ключем для цієї таблиці.

– skill_set_id - цей ідентифікатор означає, який набір навичок має користувач.

– skill_level - цей числовий атрибут кількісно визначає досвід роботи шукачів у певній кваліфікації. Число від 1 (початківець) до 10 (експерт) вказує на рівень їхньої кваліфікації.

Нарешті, таблиця skill_set містить описи всіх навичок, наведених у атрибуті skill_set_id наведеної таблиці. Вона містить лише два стовпці, ім'я skill_set_name та пов'язаний з ним ідентифікатор.

- Публікація та пошук робочих місць

Це головна особливість порталу пошуку роботи. Тільки зареєстровані вербувальники можуть розміщувати роботу на порталі, і лише зареєстровані шукачі роботи можуть звертатися до них.

Таблиця job_post є основною таблицею у цій темі. Як ви можете здогадатися, вона містить інформацію про роботу. Всі інші таблиці в цьому розділі створені навколо неї та пов'язані з нею.

– id - первинний ключ цієї таблиці. Кожен пост роботи призначається унікальним номером, а цей номер використовується в інших таблицях.

– posted_by_id - у цьому стовпці зберігається register_user_id рекрутера, який опублікував роботу.

– job_type_id - цей стовпець визначає, чи тривалість роботи є постійною або тимчасовою (контракт).

– company_id - у цьому стовпчику зберігається ідентифікатор компанії, пов'язаний з посадою вакансії. Це посилання на таблицю компаній.

– is_company_name_hidden – стовпчик-прапорець, який показує, чи повинно ім'я компанії бути показано шукачам роботи. Рекрутери можуть віддати перевагу не показувати назви компаній у своїх публікаціях. Замість цього вони використовують такі терміни, як «Глобальна автомобільна компанія».

– created_date - зберігає дату публікації.

– job_description - короткий опис роботи.

– job_location_id - посилання на атрибут у таблиці job_location, де зберігається фактичне місце роботи: адресу, місто, область, країна та поштовий індекс.

– is_active - Це означає, що робота все ще відкрита. Рекрутери можуть відмічати свої посади неактивними, як тільки позиції будуть заповнені.

Таблиця work_post_skill_set зберігає подробиці про набори навичок, необхідні для роботи. Структура таблиці ідентична таблиці seeker_skill_set.

І остання таблиця у цьому розділі, job_post_activity, містить відомості про тих, хто шукає роботу, і коли.

3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ

3.1 Проектування інтерфейсу обробки даних

Опишемо структуру інтерфейсу системи та засобів, які було використано в процесі реалізації функцій системи.

Згідно з предметною областю система взаємодіє з великою кількістю користувачів по всій Київській області, тому, для облегшення доступу до системи було прийнято рішення: створити сервер для взаємодії із базою даних. Клієнтова частина буде надсилати запити до нього та не буде обробляти багато зайвої логіки.

Опишемо логіку взаємодії через Web API.

Основний потік інформації проходить, як показано на рисунку 3.1.

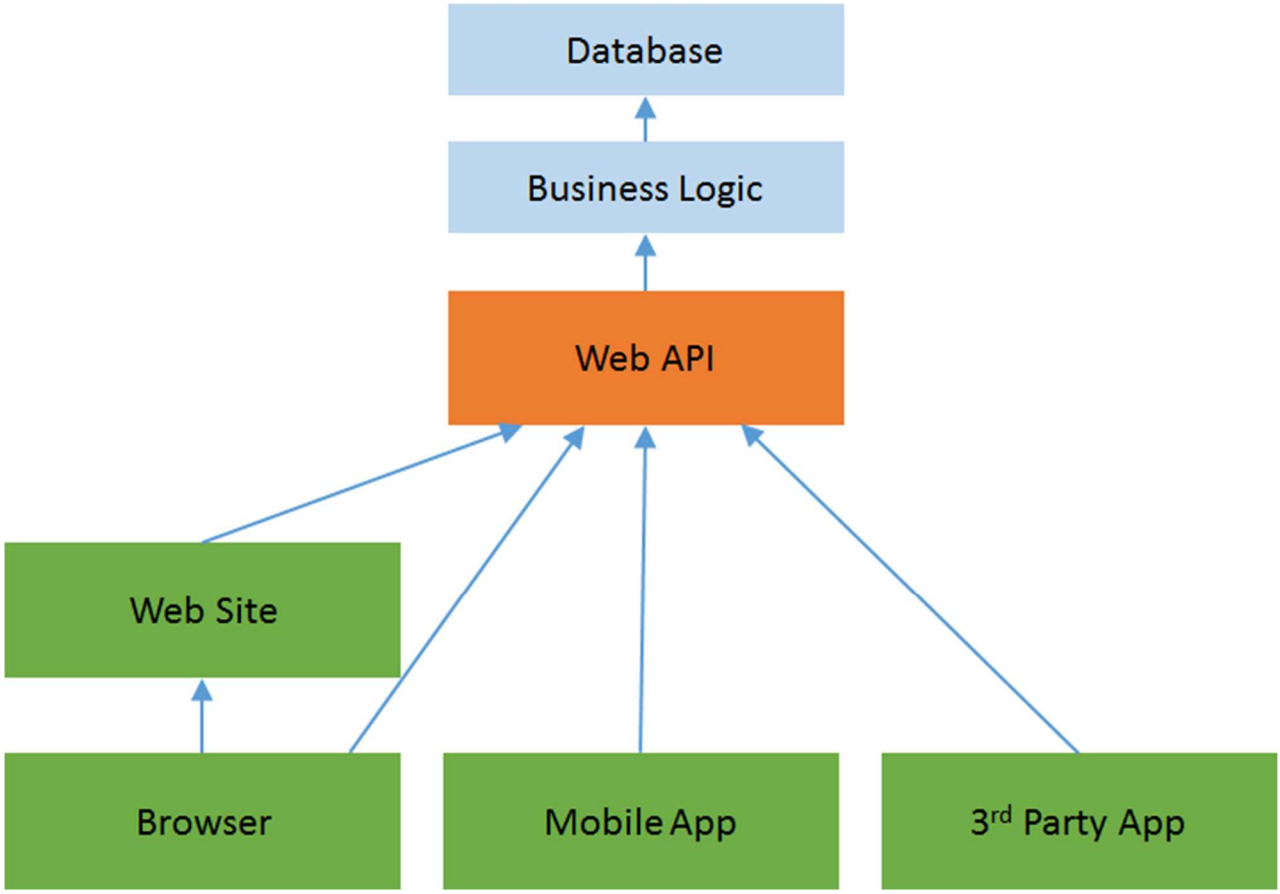

Рисунок 3.1 – Потік інформації при взаємодії із Web API

Розглянемо даний малюнок детальніше. У нашому випадку для відправлення даних та перегляду результатів використовується браузер.

Для того, щоб записати нові дані, або отримати їх – необхідно лише надіслати HTTP запит, а у відповідь прийдуть готові результати.

Уся логіка обробки запитів лягає на сторону сервера (що в нашому випадку є вигідним), а також уся взаємодія із базою даних проходить крізь нього.

Для взаємодії із базою даних був обраний Entity Framework, який легко взаємодіяти із базо даних та виконувати основні операції більш типово-залежно, тобто більш безпечно.

Також, в Entity Framework вбудований захист від SQL Injections та підтримує міграції бази даних.

3.2 Фізична реалізація бази даних

У даному пункті опишемо фізичну реалізацію бази даних. Буде наведено SQL запити для створення схеми бази даних.

Згідно з предметною областю та фізичною моделлю бази даних (та ER діаграмою) створимо SQL запит для створення визначених таблиць.

```
-- Created by Vertabelo (http://vertabelo.com)
-- Last modification date: 2018-01-02 12:39:42.833

-- tables
-- Table: business_stream
CREATE TABLE business_stream (
    id int  NOT NULL,
    business_stream_name varchar(100)  NOT NULL,
    CONSTRAINT business_stream_pk PRIMARY KEY  (id)
);

-- Table: company
CREATE TABLE company (
    id int  NOT NULL,
    company_name varchar(100)  NOT NULL,
    profile_description varchar(1000)  NOT NULL,
    business_stream_id int  NOT NULL,
    establishment_date date  NOT NULL,
    company_website_url varchar(500)  NOT NULL,
    CONSTRAINT company_pk PRIMARY KEY  (id)
);

-- Table: company_image
CREATE TABLE company_image (
    id int  NOT NULL,
```

```
    company_id int  NOT NULL,
    company_image image  NOT NULL,
    CONSTRAINT company_image_pk PRIMARY KEY  (id)
);

-- Table: education_detail
CREATE TABLE education_detail (
    user_account_id int  NOT NULL,
    certificate_degree_name varchar(50)  NOT NULL,
    major varchar(50)  NOT NULL,
    Institute_university_name varchar(50)  NOT NULL,
    starting_date date  NOT NULL,
    completion_date date  NULL,
    percentage int  NULL,
    cgpa int  NULL,
    CONSTRAINT education_detail_pk PRIMARY KEY  (user_account_id,certificate_degree_name,major)
);

-- Table: experience_detail
CREATE TABLE experience_detail (
    user_account_id int  NOT NULL,
    is_current_job char(1)  NOT NULL,
    start_date date  NOT NULL,
    end_date date  NOT NULL,
    job_title varchar(50)  NOT NULL,
    company_name varchar(100)  NOT NULL,
    job_location_city varchar(50)  NOT NULL,
    job_location_state varchar(50)  NOT NULL,
    job_location_country varchar(50)  NOT NULL,
    description varchar(4000)  NOT NULL,
    CONSTRAINT experience_detail_pk PRIMARY KEY  (user_account_id,start_date,end_date)
);

-- Table: job_location
CREATE TABLE job_location (
    id int  NOT NULL,
    street_address varchar(100)  NOT NULL,
    city varchar(50)  NOT NULL,
    state varchar(50)  NOT NULL,
    country varchar(50)  NOT NULL,
    zip varchar(50)  NOT NULL,
    CONSTRAINT job_location_pk PRIMARY KEY  (id)
);

-- Table: job_post
CREATE TABLE job_post (
    id int  NOT NULL,
    posted_by_id int  NOT NULL,
    job_type_id int  NOT NULL,
    company_id int  NOT NULL,
    is_company_name_hidden char(1)  NOT NULL,
    created_date date  NOT NULL,
    job_description varchar(500)  NOT NULL,
    job_location_id int  NOT NULL,
    is_active char(1)  NOT NULL,
    CONSTRAINT job_post_pk PRIMARY KEY  (id)
);

-- Table: job_post_activity
CREATE TABLE job_post_activity (
    user_account_id int  NOT NULL,
    job_post_id int  NOT NULL,
    apply_date date  NOT NULL,
    CONSTRAINT job_post_activity_pk PRIMARY KEY  (user_account_id,job_post_id)
);
```

```
-- Table: job_post_skill_set
CREATE TABLE job_post_skill_set (
    skill_set_id int  NOT NULL,
    job_post_id int  NOT NULL,
    skill_level int  NOT NULL,
    CONSTRAINT job_post_skill_set_pk PRIMARY KEY  (skill_set_id,job_post_id)
);

-- Table: job_type
CREATE TABLE job_type (
    id int  NOT NULL,
    job_type varchar(20)  NOT NULL,
    CONSTRAINT job_type_pk PRIMARY KEY  (id)
);

-- Table: seeker_profile
CREATE TABLE seeker_profile (
    user_account_id int  NOT NULL,
    first_name varchar(50)  NOT NULL,
    last_name varchar(50)  NOT NULL,
    current_salary int  NULL,
    is_annually_monthly char(1)  NULL,
    currency varchar(50)  NULL,
    CONSTRAINT seeker_profile_pk PRIMARY KEY  (user_account_id)
);

-- Table: seeker_skill_set
CREATE TABLE seeker_skill_set (
    user_account_id int  NOT NULL,
    skill_set_id int  NOT NULL,
    skill_level int  NOT NULL,
    CONSTRAINT seeker_skill_set_pk PRIMARY KEY  (user_account_id,skill_set_id)
);

-- Table: skill_set
CREATE TABLE skill_set (
    id int  NOT NULL,
    skill_set_name varchar(50)  NOT NULL,
    CONSTRAINT skill_set_pk PRIMARY KEY  (id)
);

-- Table: user_account
CREATE TABLE user_account (
    id int  NOT NULL,
    user_type_id int  NOT NULL,
    email varchar(255)  NOT NULL,
    password varchar(100)  NOT NULL,
    date_of_birth date  NOT NULL,
    gender char(1)  NOT NULL,
    is_active char(1)  NOT NULL,
    contact_number int  NOT NULL,
    sms_notification_active char(1)  NOT NULL,
    email_notification_active char(1)  NOT NULL,
    user_image image  NULL,
    registration_date date  NOT NULL,
    CONSTRAINT user_account_pk PRIMARY KEY  (id)
);

-- Table: user_log
CREATE TABLE user_log (
    user_account_id int  NOT NULL,
    last_login_date date  NOT NULL,
    last_job_apply_date date  NULL,
    CONSTRAINT user_log_pk PRIMARY KEY  (user_account_id)
```

```
);
-- Table: user_type
CREATE TABLE user_type (
    id int  NOT NULL,
    user_type_name varchar(20)  NOT NULL,
    CONSTRAINT user_type_pk PRIMARY KEY  (id)
);

-- foreign keys
-- Reference: company_business_stream (table: company)
ALTER TABLE company ADD CONSTRAINT company_business_stream
    FOREIGN KEY (business_stream_id)
    REFERENCES business_stream (id);

-- Reference: company_image_company (table: company_image)
ALTER TABLE company_image ADD CONSTRAINT company_image_company
    FOREIGN KEY (company_id)
    REFERENCES company (id);

-- Reference: educ_dtl_seeker_profile (table: education_detail)
ALTER TABLE education_detail ADD CONSTRAINT educ_dtl_seeker_profile
    FOREIGN KEY (user_account_id)
    REFERENCES seeker_profile (user_account_id);

-- Reference: exp_dtl_seeker_profile (table: experience_detail)
ALTER TABLE experience_detail ADD CONSTRAINT exp_dtl_seeker_profile
    FOREIGN KEY (user_account_id)
    REFERENCES seeker_profile (user_account_id);

-- Reference: job_post_act_user_register (table: job_post_activity)
ALTER TABLE job_post_activity ADD CONSTRAINT job_post_act_user_register
    FOREIGN KEY (user_account_id)
    REFERENCES user_account (id);

-- Reference: job_post_activity_job_post (table: job_post_activity)
ALTER TABLE job_post_activity ADD CONSTRAINT job_post_activity_job_post
    FOREIGN KEY (job_post_id)
    REFERENCES job_post (id);

-- Reference: job_post_company (table: job_post)
ALTER TABLE job_post ADD CONSTRAINT job_post_company
    FOREIGN KEY (company_id)
    REFERENCES company (id);

-- Reference: job_post_job_location (table: job_post)
ALTER TABLE job_post ADD CONSTRAINT job_post_job_location
    FOREIGN KEY (job_location_id)
    REFERENCES job_location (id);

-- Reference: job_post_job_type (table: job_post)
ALTER TABLE job_post ADD CONSTRAINT job_post_job_type
    FOREIGN KEY (job_type_id)
    REFERENCES job_type (id);

-- Reference: job_post_skill_set_job_post (table: job_post_skill_set)
ALTER TABLE job_post_skill_set ADD CONSTRAINT job_post_skill_set_job_post
    FOREIGN KEY (job_post_id)
    REFERENCES job_post (id);

-- Reference: job_post_skill_set_skill_set (table: job_post_skill_set)
ALTER TABLE job_post_skill_set ADD CONSTRAINT job_post_skill_set_skill_set
    FOREIGN KEY (skill_set_id)
    REFERENCES skill_set (id);

-- Reference: job_post_user_register (table: job_post)
```

```
ALTER TABLE job_post ADD CONSTRAINT job_post_user_register
    FOREIGN KEY (posted_by_id)
    REFERENCES user_account (id);

-- Reference: seeker_profile_user_register (table: seeker_profile)
ALTER TABLE seeker_profile ADD CONSTRAINT seeker_profile_user_register
    FOREIGN KEY (user_account_id)
    REFERENCES user_account (id);

-- Reference: seeker_skill_set_skill_set (table: seeker_skill_set)
ALTER TABLE seeker_skill_set ADD CONSTRAINT seeker_skill_set_skill_set
    FOREIGN KEY (skill_set_id)
    REFERENCES skill_set (id);

-- Reference: skill_set_seeker_profile (table: seeker_skill_set)
ALTER TABLE seeker_skill_set ADD CONSTRAINT skill_set_seeker_profile
    FOREIGN KEY (user_account_id)
    REFERENCES seeker_profile (user_account_id);

-- Reference: use_log_user_register (table: user_log)
ALTER TABLE user_log ADD CONSTRAINT use_log_user_register
    FOREIGN KEY (user_account_id)
    REFERENCES user_account (id);

-- Reference: user_register_user_type (table: user_account)
ALTER TABLE user_account ADD CONSTRAINT user_register_user_type
    FOREIGN KEY (user_type_id)
    REFERENCES user_type (id);
```

У пункті 1.2 як система управління базами даних був обраний MS SQL SERVER. За допомогою зазначеної СУБД виконаємо даний скрипт. При виконанні даного скрипту отримаємо: Script has been executed. Return code: 0, що позначає успішне виконання.

Як підтвердження успішності виконання скрипту, переглянемо обігрівач SQL SERVER. Результат SQL SERVER показано на рисунку 3.2.

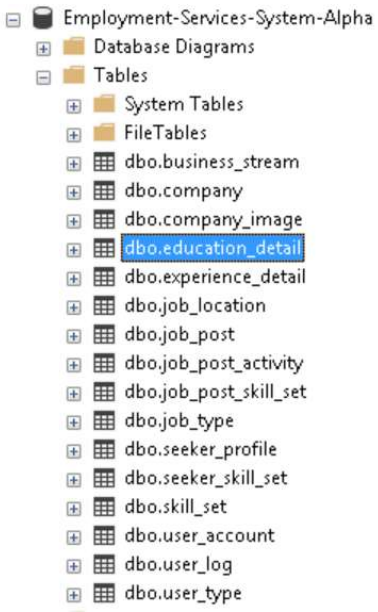

Рисунок 3.2 – Створена база даних та таблиці після успішного виконання скрипту

3.3 Виконання основних запитів

У даному розділі опишемо основні операції над базою даних, які необхідні для коректної роботи системи. Також, опишемо складні запити для генерації звітів, вказаних як функціональна вимога до системи.

Для реалізації доволі тривіальних засобів використаємо вбудовані до Entity Framework засоби, які автоматично генерують Sql команди та виконують їх.

Для підключення до бази даних через Entity Framework, нам потрібний посередник - контекст даних. Контекст даних представляє собою клас, похідний від класу DbContext. Контекст даних містить один або кілька властивостей типу DbSet <T>, де T являє собою тип об'єкта, що зберігається в базі даних. Для того, щоб зв'язати додаток, контекст даних та БД, необхідно додати в файл web.config (app. config) строку підключення до даної бази даних.

Прикладом використання автогенероравних операцій показано у коді нижче.

```
// GET: api/companies/5
        [ResponseType(typeof(company))]
        public async Task<IHttpActionResult> Getcompany(int id)
        {
            company company = await db.company.FindAsync(id);
            if (company == null)
            {
                return NotFound();
            }

            return Ok(company);
        }
```

Де db містить варіації DbSet. Вираз db.company буде перетворено у наступний SQL запит та виконано:

```
SELECT
   [Extent1].[user_account_id] AS [user_account_id],
   [Extent1].[skill_set_id] AS [skill_set_id],
   [Extent1].[skill_level] AS [skill_level]
   FROM [dbo].[seeker_skill_set] AS [Extent1]
   WHERE [Extent1].[user_account_id] = @EntityKeyValue1',N'@EntityKeyValue1 int
```

Більш складні запити потребують написання власних LINQ запитів. Реалізуємо основні запити, сказані в предметній області.

– Пошук роботи за ключовим словом і місцем знаходження.

```
[Route("api/JobSearch/Simple")]
        public IQueryable<JobPostDTO> JobSearchSimple(
            string keyword="",
            string location="")
        {
            var JobPosts = from j in db.job_post
                           .Where(job_post =>
                               (
                                   job_post.job_type.job_type1.Contains(keyword) ||
                                   job_post.company.company_name.Contains(keyword) ||
                                   job_post.job_description.Contains(keyword)
                               )
                               &&
                               (
                                   job_post.job_location.street_address.Contains(location) ||
                                   job_post.job_location.state.Contains(location) ||
                                   job_post.job_location.city.Contains(location)
                               )
                           )
                           select new JobPostDTO()
                           {
                               Id = j.id,
                               JobType = j.job_type.job_type1,
                               Company = j.company.company_name,
                               JobDescription = j.job_description,
                               JobLocationStreet = j.job_location.street_address,
                           };
            return JobPosts;
        }
```

Розглянемо скрипт більш детально.

```
var JobPosts = from j in db.job_post
```

Виконуємо вибірку усіх даних відповідно до моделі db.job_post, що

```
.Where(job_post =>
```

відповідають наступним умовам:

```
(
      job_post.job_type.job_type1.Contains(keyword) ||
      job_post.company.company_name.Contains(keyword) ||
      job_post.job_description.Contains(keyword)
)
```

Назва виду роботи, або назва компанії, або опис роботи містить ключове слово.

```
&&
(
      job_post.job_location.street_address.Contains(location) ||
      job_post.job_location.state.Contains(location) ||
      job_post.job_location.city.Contains(location)
)
```

А також, повна адреса роботи містить вказане місце знаходження.

```
select new JobPostDTO()
      {
            Id = j.id,
            JobType = j.job_type.job_type1,
            Company = j.company.company_name,
            JobDescription = j.job_description,
            JobLocationStreet = j.job_location.street_address,
      };
```

З цієї вибірки, для кожної моделі db.job_post ми створюємо Data Transfer Object для передачі даних клієнту в зрозумілому вигляді – такий об’єкт містить коротку інформацію про роботу для використання у списку знайдених робіт.

Виконаємо цей запит з параметром location = Komarova. Тоді результат виконання знаходить усі роботи, наявні на проспекі Комарова:


```
[
  {
    "$id": "1",
    "Id": 1,
    "JobType": "Accounting",
    "Company": "Illegal Metal",
    "JobDescription": "Do metal things",
    "JobLocationStreet": "Komarova 134"
  },
  {
    "$id": "2",
    "Id": 5,
    "JobType": "IT",
    "Company": "2 coders 1 deadline",
    "JobDescription": "PHP coding like no tomorrow",
    "JobLocationStreet": "Komarova 132"
  }
]
```


– Повний пошук роботи з усіма можливими параметрами

```
var JobPosts = from j in db.job_post
```

Як і в простому пошуку, виконуємо вибірку усіх даних відповідно до моделі db.job_post, у яких назва виду роботи, або назва компанії, або опис роботи містить ключове слово, і повна адреса роботи містить вказане місце знаходження.

Додатково, ми виконуємо пошук за наступними критеріями:

```
&&
(
	category != "" ? job_post.job_type.job_type1 == category : true
)
```

Тип роботи повинен відповідати зазначеному у полі «category».

```
&&
(
	freshnessDays != 0 ? job_post.created_date < MinCreationDate : true
)
```

де

```
DateTime MinCreationDate = DateTime.Now.AddDays(-freshnessDays);
```

Кількість днів після створення роботи не повинен бути більше за вказану у «freshnessDays».

```
&&
(
	industry != "" ? job_post.company.business_stream.business_stream_name == industry :
true

)
```

Основний прибуток компанії має відповідати зазначеному у полі «industry».

```
.OrderBy(job_post =>
	(freshnessPriority > expPriority)
	?
	job_post.created_date
	:
	DateTime.MinValue
	)
.OrderBy(job_post =>
	(freshnessPriority <= expPriority)
		?
		0
		:
		db.job_post_skill_set.FirstOrDefault(skill => skill.job_post_id ==
	job_post.id).skill_level
	)
```

В залежності від пріорітету, роботи будуть відсортовані або за датою створення, або за необхідним стажем.

З цієї вибірки, для кожної моделі db.job_post ми так само створюємо Data Transfer Object.

Виконаємо цей запит з параметрами freshnessDays= 2 і industry = Cleaning Services. Тоді результат виконання знаходить усі роботи у даній галузі що були додані за останні два дні:

```
[
  {
    "$id": "1",
    "Id": 3,
    "JobType": "Safety and security",
    "Company": "Hurricane",
    "JobDescription": "Office security",
    "JobLocationStreet": "Hnata Yuri 11"
  }
]
```


– Повний пошук шукача роботи з усіма параметрами:

```
[System.Web.Http.AcceptVerbs("GET")]
        [System.Web.Http.HttpGet]
        [Route("api/SeekerSearch/Advanced")]
        public IQueryable<SeekerDTO> SeekerSearchAdvanced(
            string keyword = "",
            string location = "",
            string skillSet = "",
            int xpYears = 0,
            string eduQualification = "",
            int YearOfPassingMin = 0,
            int YearOfPassingMax = 0
            )
        {
            var Seekers = from s in db.seeker_profile
                          where (
                            (
                                db.seeker_skill_set.Any(skill =>
                                    (skill.user_account_id == s.user_account_id)
                                    &&
                                    (skill.skill_set.skill_set_name.Contains(keyword)))
                                ||
                                db.experience_detail.Any(exp =>
                                    (exp.user_account_id == s.user_account_id)
                                    &&
                                    (exp.job_title.Contains(keyword)))
                                ||
                                db.experience_detail.Any(exp =>
                                    (exp.user_account_id == s.user_account_id)
                                    &&
                                    (exp.company_name.Contains(keyword)))
                                ||
                                db.experience_detail.Any(exp =>
                                    (exp.user_account_id == s.user_account_id)
                                    &&
                                    (exp.description.Contains(keyword)))
                                ||
                                db.education_detail.Any(edu =>
                                    (edu.user_account_id == s.user_account_id)
                                    &&
                                    (edu.certificate_degree_name.Contains(keyword)))
                                ||
```

```
                    db.education_detail.Any(edu =>
                        (edu.user_account_id == s.user_account_id)
                        &&
                        (edu.major.Contains(keyword)))
                    ||
                    db.education_detail.Any(edu =>
                        (edu.user_account_id == s.user_account_id)
                        &&
                        (edu.Institute_university_name.Contains(keyword)))
                )
                &&
                (
                    db.experience_detail.Any(exp =>
                        (exp.user_account_id == s.user_account_id)
                        &&
                        (exp.job_location_city.Contains(location)))
                    ||
                    db.experience_detail.Any(exp =>
                        (exp.user_account_id == s.user_account_id)
                        &&
                        (exp.job_location_country.Contains(location)))
                    ||
                    db.experience_detail.Any(exp =>
                        (exp.user_account_id == s.user_account_id)
                        &&
                        (exp.job_location_state.Contains(location)))
                )
                &&
                (
                    db.seeker_skill_set.Any(skill =>
                        (skill.user_account_id == s.user_account_id)
                        &&
                        (skill.skill_set.skill_set_name == skillSet))
                    || skillSet == ""
                )
                &&
                (
                    db.experience_detail.Any(exp =>
                        (exp.user_account_id == s.user_account_id)
                        &&
                        (DbFunctions.DiffYears(exp.end_date, exp.start_date)  >
xpYears))
                    || xpYears == 0
                )
                &&
                (
                    db.education_detail.Any(edu =>
                        (edu.user_account_id == s.user_account_id)
                        &&
                        (edu.certificate_degree_name.Contains(eduQualification)))
                    ||
                    db.education_detail.Any(edu =>
                        (edu.user_account_id == s.user_account_id)
                        &&
                        (edu.major.Contains(eduQualification)))
                    || eduQualification == ""
                )
                &&
                (
                    db.education_detail.Any(edu =>
                        (edu.user_account_id == s.user_account_id)
                        &&
                        (edu.completion_date.Value.Year < YearOfPassingMax))
                    || YearOfPassingMax == 0
                )
```

```
                    &&
                    (
                        db.education_detail.Any(edu =>
                            (edu.user_account_id == s.user_account_id)
                            &&
                            (edu.completion_date.Value.Year > YearOfPassingMin))
                        || YearOfPassingMin == 0
                    )
                )
                select new SeekerDTO()
                {
                    Id = s.user_account_id,
                    FirstName = s.first_name,
                    LastName = s.last_name,
                    SkillSetName = db.seeker_skill_set.FirstOrDefault(skill =>
                          skill.user_account_id == s.user_account_id)
                          .skill_set.skill_set_name,
                    SkillLevel = db.seeker_skill_set.FirstOrDefault(skill =>
                          skill.user_account_id == s.user_account_id)
                          .skill_level,
                    Degree = db.education_detail.FirstOrDefault(edu =>
                          edu.user_account_id == s.user_account_id)
                          .certificate_degree_name
                };

    return Seekers;
       }
```

Розглянемо скрипт більш детально.

Виконуємо вибірку усіх даних відповідно до моделі db.seeker_profile, що відповідають наступним умовам:

Тип однієї з минулих робіт, або їх назви, опис, назви компаній у яких шукач працював, спеціальність або назва вищого навчального закладу, що шукач закінчив, мають містити ключове слово, а хоча б одна з повних адрес вказаних у минулих роботах має містити місце знаходження. Хоча б одна з минулих робіт має відповідати вказаній спеціальності, а досвід роботи має бути не менше ніж зазначено у «xpYears». Напрямок вищої освіти має відповідати заданому, а рік випуску має бути в межах від «YearOfPassingMin» до «YearOfPassingMax».

З цієї вибірки, для кожної моделі db.seeker_profile ми створюємо Data Transfer Object для передачі даних клієнту в зрозумілому вигляді – такий об’єкт містить коротку інформацію про роботу для використання у списку знайдених робіт.

Виконаємо цей запит з параметрами keyword= PHP, location= Oblast і xpYears = 2. Тоді результат виконання знаходить усіх шукачів у даній галузі по Київській області що мають не менш ніж два роки досвіду:

```
[
  {
    "$id": "1",
    "Id": 5,
    "FirstName": "Gib",
    "LastName": "Jobplz",
    "SkillSetName": "PHP coding",
    "SkillLevel": 9,
    "Degree": "Computer engineering"
  }
]
```


Знаючи, що id цього шукача – 5, можемо виконати запит «http://localhost:57087/api/seeker_profile/5», і отримати детальну інформацію:


```
{
  "$id": "1",
  "Id": 5,
  "_PersonalInfo": {
    "$id": "2",
    "FirstName": "Gib",
    "LastName": "Jobplz"
  },
  "_ExperienceDetail": {
    "$id": "3",
    "CurrentSalary": 100,
    "Currency": "EUR",
    "SkillSetName": "PHP coding",
    "SkillLevel": 9,
    "JobTitle": "PHP coder",
    "CompanyName": "Best-Bros technologies",
    "JobLocationCity": "Kiev",
```

```
    "JobLocationState": "Kievskaya Oblast",
    "JobLocationCountry": "Ukraine",
    "Description": "Server side programming",
    "IsCurrentJob": false,
    "StartDate": "2015-09-19T00:00:00",
    "EndDate": "2017-09-20T00:00:00"
  },
  "_EducationDetail": {
    "$id": "4",
    "CertificateDegreeName": "Computer engineering",
    "Major": "Computer engineering",
    "InstituteUniversityName": "KPI",
    "StartingDate": "2009-09-01T00:00:00",
    "CompletionDate": "2013-06-30T00:00:00",
    "Percentage": 81,
    "Cgpa": 84
  }
}
```


3.4 Підтримка міграцій бази даних

За допомогою засобів Entity Framework можна підтримувати міграції бази даних: вказати що зміниться при підвищенні версії бази даних та що відбудеться при її пониженні.

Наприклад, однією з необхідних міграцій у проекті буда заміна стовбця «Institute_university_name» на «institute_university_name». Кожна міграція у Entity Framework має бути екземпляром класу DbMigration. Оскільки у розробці використовувався метод Database First, для міграції спочатку необхідно оновити базу даних, а потім – модель даних.

Оновимо базу даних:

```
/*
   10 января 2018 г.10:26:51
   User:
   Server: (LocalDb)\MSSQLLocalDB
   Database: Employment-Services-System-Alpha
   Application:
*/

/* To prevent any potential data loss issues, you should review this script in detail before
running it outside the context of the database designer.*/
BEGIN TRANSACTION
SET QUOTED_IDENTIFIER ON
SET ARITHABORT ON
SET NUMERIC_ROUNDABORT OFF
SET CONCAT_NULL_YIELDS_NULL ON
SET ANSI_NULLS ON
SET ANSI_PADDING ON
SET ANSI_WARNINGS ON
COMMIT
BEGIN TRANSACTION
GO
EXECUTE sp_rename N'dbo.education_detail.Institute_university_name',
N'Tmp_institute_university_name', 'COLUMN'
GO
EXECUTE sp_rename N'dbo.education_detail.Tmp_institute_university_name',
N'institute_university_name', 'COLUMN'
GO
ALTER TABLE dbo.education_detail SET (LOCK_ESCALATION = TABLE)
GO
COMMIT
```

Після цього на необхідно оновити модель даних. Відповідно виконується дія «Update model from database». Її виконання показано на рисунку 3.2

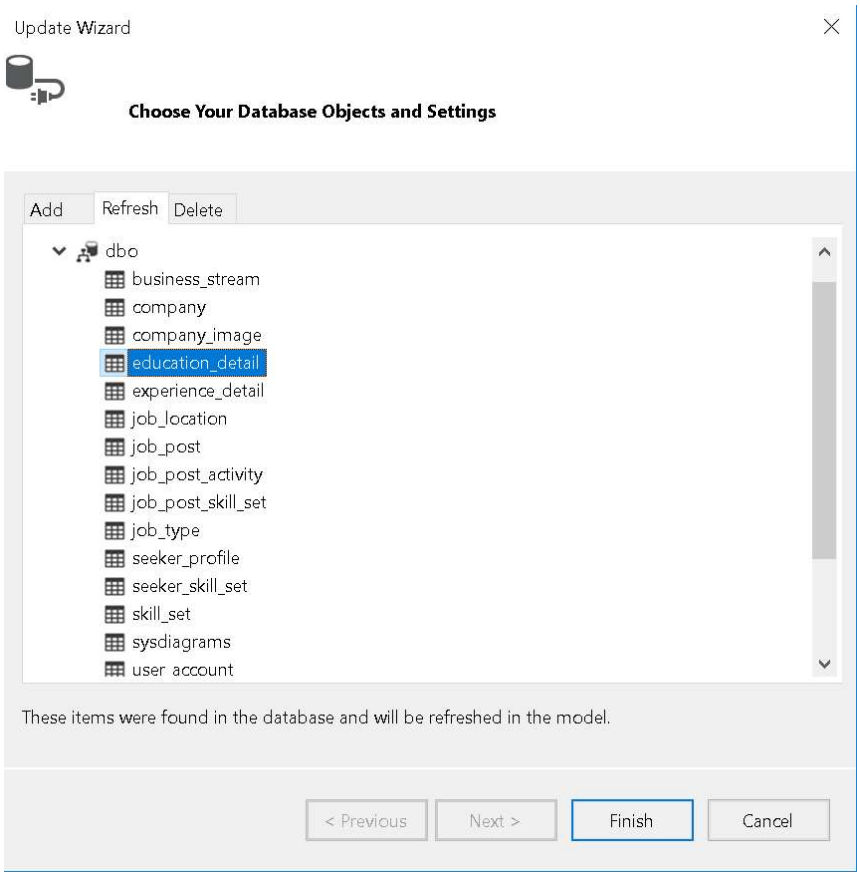

Рисунок 3.3 – Update model from database

4 АДМІНІСТРУВАННЯ БАЗИ ДАНИХ

4.1 Порядок налаштування сервера

Запускаємо інсталятор MS SQL SERVER і у вітальному діалозі вибираємо Installation -> New installation or Add features to an existing installation :

При запуску інсталяції в першу чергу проводяться перевірки сумісності операційної системи, чи вистачає прав у користувача для установки SQL Server і т.п. Якщо у вас вже стоїть екземпляр SQL Server 2008, то ви побачите попередження про те, що загальні компоненти (SQL Server Management Studio, Integration Services, Books Online тощо) будуть оновлені до версії до SQL Server 2008 R2. Якщо всі перевірки пройдено, можна переходити до наступного кроку. І після розпакування файлів, необхідних для інсталяції та ще одного етапу перевірок, почнеться найцікавіше - конфігурація установки :

Першим етапом якої буде введення ліцензійного ключа продукту, або вибору безкоштовної редакції (Evaluation, Express, Express with Advanced Services). І якщо ви ставите редакцію SQL Server, де ключ вже вбитий в полі « Enter the product key » (наприклад Developer Edition), то раджу зберегти його.

Наступним кроком буде вибір формату установки, де пропонується 3 варіанти :

- SQL Server Feature Installation - тут всі налаштування доведеться робити.
- SQL Server PowerPivot for SharePoint - окрім самого SQL Server, буде поставлений і сконфігурований PowerPivot плагін до SharePoint.
- All Features With Defaults - для установки будуть вибрані всі фічі (з можливістю прибрати те, що не потрібно) і проставлені акаунти за замовчуванням для сервісів:

На наступному екрані вибираємо ті компоненти SQL Server, які хочемо встановити. Тут доцільно вибрати все. Перелік елементі, які можна вибрати для установки (більш детальний опис компонентів можна отримати натиснувши F1 на поточному кроці):

- Database Engine Services - SQL Server.
- SQL Server Replication - компоненти реплікації SQL Server використовуються для синхронізації баз даних.
- Full - Text Search - компонент повнотекстового пошуку дозволяє організувати ефективний пошук по текстових полям бази з обліків різних мов і різних форм слова.
- Analysis Services - дозволяє будувати багатовимірні (OLAP) сховища даних і DataMining моделі для проведення аналізу та побудови прогнозів..
- Reporting Services - сервіси та інструменти для побудови та управління звітами.
- Shared Features (вони ставляться 1 раз, і будуть доступні всім екземплярам, які встановлені на машині).
- Business Intelligence Development Studio - якщо стоїть Visual Studio, то до неї додаються нові типи проектів для розробки рішень Analysis Services, Reporting Services і Integration Services. Якщо Visual Studio немає, то ставиться «міні» Visual Studio, в якій доступні тільки ці, перераховані вище типи проектів.
- Client Tools Connectivity - провайдери для з'єднання клієнтів з сервером.
- Integration Services - сервіси, що дозволяють організувати отримання, перетворення і перенесення даних з різних джерел.
- Client Tools Backwards Compatibility - SQL Distributed Management Objects (SQL - DMO), Decision Support Objects (DSO), Data Transformation Services (DTS).
- Client Tools SDK - SDK для розробників.
- SQL Server Books Online - документація по SQL Server.
- Management Tools - Basic - базовий варіант Management Studio, SQLCMD і SQL Server PowerShell provider.
- Management Tools - Complete - повноцінна Management Studio (підтримка Analysis Serveices, Integration Services, Reporting Services), Profiler, Database Engine Tuning Advisor, SQL Server Utility.
- SQL Client Tools Connectivity SDK - на Microsoft Connect є баг щодо опису цього елемента - SQL Client Connectivity SDK and Client Tools SDK DOCUMENTATION.

– Microsoft Sync Framework - багатофункціональна платформа синхронізації, що дозволяє інтегрувати будь-який додаток з будь-якими даними з будь-якого сховища, за будь-якого протоколу і в будь-якій мережі.

4.1 Налаштування прав доступу

Згідно з предметною областю, три основних сутності будуть користуватися базою даних: шукачі роботи, представники компаній та адміністратори.

Для усіх сутностей необхідним є можливість створювати нові записи у базі даних.

Для цього використаємо наступні скрипти:

```
CREATE LOGIN Seeker WITH PASSWORD='Seeker';
USE EmployeeTimeControlSystem;
CREATE USER Seeker FOR LOGIN Seeker;
EXECUTE sp_addrolemember db_datareader, "Seeker ";
CREATE LOGIN Company WITH PASSWORD= Company;
USE EmployeeTimeControlSystem;
CREATE USER Company FOR LOGIN Company;
EXECUTE sp_addrolemember db_datareader, "Seeker ";
CREATE LOGIN Admin WITH PASSWORD= Admin;
USE EmployeeTimeControlSystem;
CREATE USER Admin FOR LOGIN Admin;
EXECUTE sp_addrolemember db_datareader, "Admin ";
```

ВИСНОВКИ

У ході виконання даної курсової роботи, була спроектована та створена база даних Київської обласної дирекції служби зайнятості населення. Описані особливості предметної області, створені запити до бази даних, які надають можливість легко знайти роботу. Створення профілів - як портал дозволяє шукачам роботи та організаціям створювати профілі та резюме.

База даних надає дані для керування користувачами, створення профілів та резюме, публікації та пошуку робочих місць.

Для фізичної реалізації бази даних була вибрана система управління базою даних MS SQL SERVER – на основі порівняння з іншими СУБД. Описано основні етапи налаштування сервера та надання прав користувачам. Взаємодія між апаратною частиною, користувачем та базою даних була полегшена за допомогою Web API, який в свою чергу взаємодіє із базою даних за допомогою Entity Framework.

У ході роботи надано сніпети із вихідного кода, надано LINQ скрипти основних запитів до бази даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційні технології. Процеси життєвого циклу програмного забезпечення (ISO/IEC 12207:1995): ДСТУ 3918-1999. – [Чинний від 2000-01-01]. – К.: Держстандарт України, 2000. – 50 с. – (Національний стандарт України).

2. Інформаційні технології. Основні напрямки оцінювання та відбору CASE-інструментів (ISO/IEC 14102:1995) – ДСТУ 3919-1999 [Чинний від 2000-01-01]. – К.: Держстандарт України, 2000. – 470 с. – (Національний стандарт України).

3. Вендров А. М. CASE-технологии – современные методы и средства проектирования информационных систем / Вендров А. М. – М.: Финансы и статистика, 1998 – 171 с.

4. Кальянов Г. Н. CASE. Структурный системный анализ (автоматизация и применение) / Кальянов Г. Н. – М.: Лори, 1996. – 242 с.

5. Марка Д., Методология структурного анализа и проектирования / Д. Марка, К. МакГоуэн. – М.: МетаТехнология, 1993. – 240 с.

6. Черемных С. В. Структурный анализ систем: IDEF-технологии / Черемных С. В., Семенов И. О., Ручки В. С. – М.: Финансы и статистика, 2003. – 208 с.

7. Кватрани Терри. Визуальное моделирование с помощью Rational Rose 2002 и UML / Кватрани Т.; пер. с англ. – М.: Издательский дом „Вильямс”, 2003. – 192 с.

8. Новоженов Ю. В. Объектно-ориентированные технологии разработки сложных программных систем / Ю. В. Новоженов. – М.: Наука, 1996 – 356 с.

9. Фаулер М, Скотт К. UML в кратком изложении. Применение стандартного языка объектного моделирования / М. Фаулер, К. Скотт. – М: «Мир», 1999 – 191 с.

10. Дейт К. Введение в системы баз данных / Дейт К. – [8-е изд.] – СПб.: Вильямс, 2005 – 1328 с.

11. Праг К. Н. Access 2002 / К. Н. Праг, М. Р. Ирвин; пер. с англ. – М. Диалектика, 2003. – 1216 с. – (Серия “Библия пользователя”).

12. Фаронов В. Программирование баз данных Delphi 7. Учебный курс / Фаронов В.В. – СПб: Питер, 2005. – 459 с.

13. Чкалов А. П. Базы данных: от проектирования до разработки приложений / Чкалов А. П. – СПб.: БХВ-Петербург, 2003. – 384 с.

14. Искусственный интеллект: справочник в 3 кн. / [под ред. Э. В. Попова, Д. А. Поспелова, В. Н. Захарова, В. Ф. Хорошевского] – М.:. - Радио и связь, 1990. -

15. Кн. 2: Модели и методы / [под ред. Д. А. Поспелова]. – 1990. – 304 с.

16. Кн. 3: Программные и аппаратные средства: справочник / [под ред. В. Н. Захарова, В. Ф. Хорошевского] – 1990. – 368 с.

17. Глущенко В. В. Исследование множеств и разработка одного типа представления и метода планирования испытания сложных технических систем / В. В. Глущенко // Кибирнетика и системный анализ. – 1992. – № 2. – С. 27-28

18. Сугоняк І. І. Структура та сценарії роботи системи підтримки прийняття рішень з оптимального керування інноваційними процесами підприємств / І. І. Сугоняк // Наукові проблеми модернізації та застосування інформаційних систем: XVII наук. – тех. конф: 24–25 квіт. 2008 р.: тези доп. Ч. І – Житомир, 2008. – С. 86 - 87.

19. Кириллов В. В. Структуризованный язык запросов (SQL): учебн. пособ.: [Електронный ресурс] / В. В. Кириллов, Г. Ю. Громов. – СПб: Санкт-Петерб. госуд. техн. универ. , каф. выч. техники, 1998. – Режим доступа к пособию:

20. http://www.citforum.ru/database/sql_kg/.

21. Иванова В. Б. Алгоритм принятия решения о внедрении инновационного проекта [Электронный ресурс] / В. Б. Иванова // Вісн. міжнар. слов`ян. ун-ту. Серія: Екон. науки. – 2006. – 9, № 1. – С. 20-24 – Режим доступу до статті: http://www.nbuv.gov.ua/articles/2006/06ivboip. html

22. Шилдт Г. C# 4.0 Полное руководство [Текст] / Г. Шилдт. — М.: Издательский

23. дом «Вильямс», 2011. — 1056 с.

24. Троелсен Е. Язык программирования C# 5.0 и платформа .NET 4.5 [Текст] / Е. Троелсен – Москва: Издательский дом “Вильямс”, 2013. — 810 с.

25. Сайт Microsoft Developer Network [Електронний ресурс]. — Режим доступу: https://msdn.microsoft.com/en-us/library/system.runtime.serialization

26. Черемних С.В Моделирование и анализ систем [Текст] / С.В. Черемних – Москва: Финансы и статистика, 2006. – 192 с.

27. Сайт Microsoft Docs [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/en-us/ef

28. Сайт веб-API ASP.NET [Електронний ресурс]. — Режим доступу: https://msdn.microsoft.com/ru-ru/library/hh833994(v=vs.108).aspx

29. Дейт К. Дж. Введение в системы баз данных / [Дейт К. Дж] 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильяме", 2005. — 1328 с: ил.

30. Харрингтон Дж. Л. Проектирование реляционных баз данных/ [Харрингтон Дж. Л.] – М.: Издательство "Лори", 2006 – 232 с.

ДОДАТОК А

Діаграми бази даних

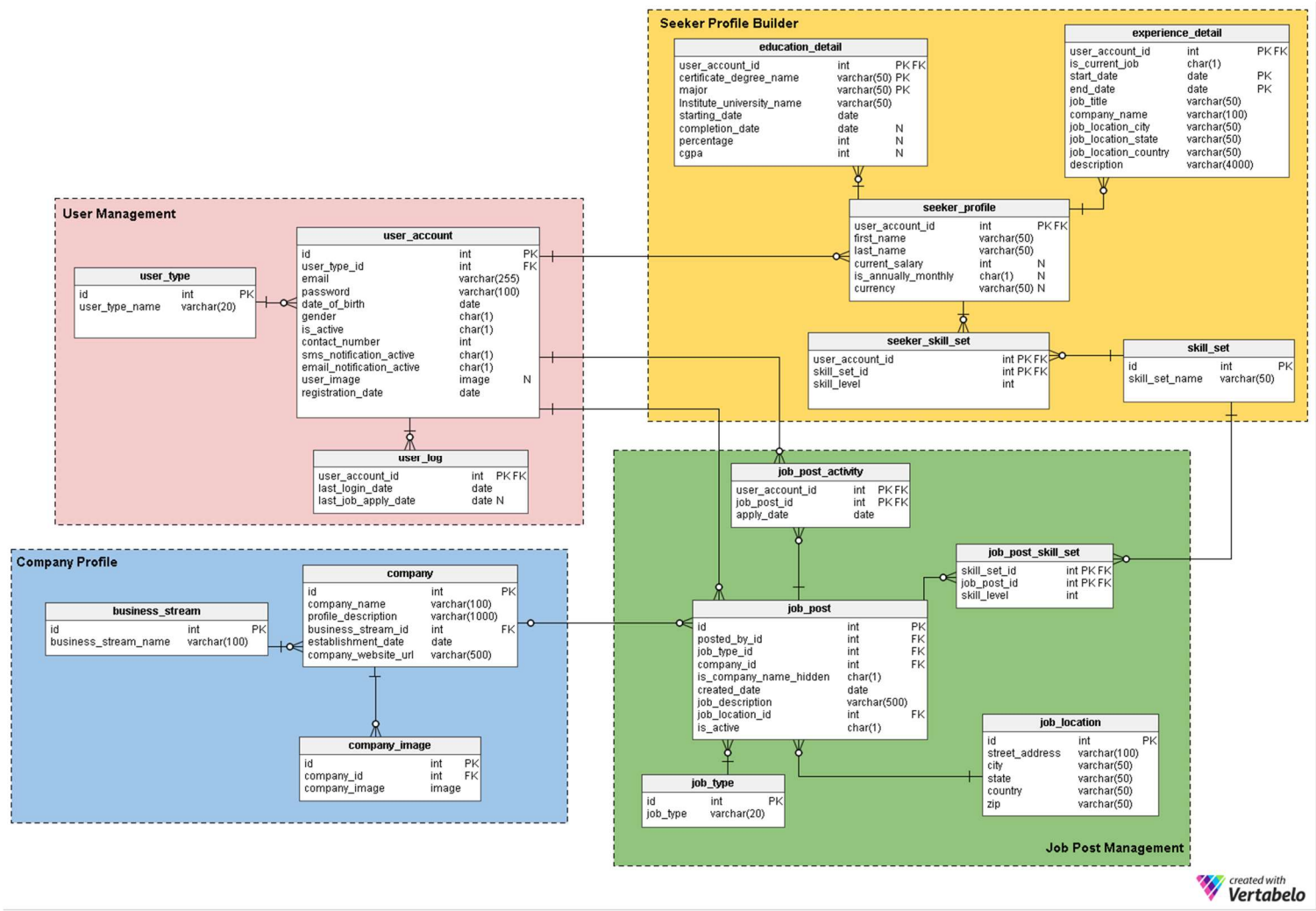

Рисунок А.1 – ER-діаграма бази даних

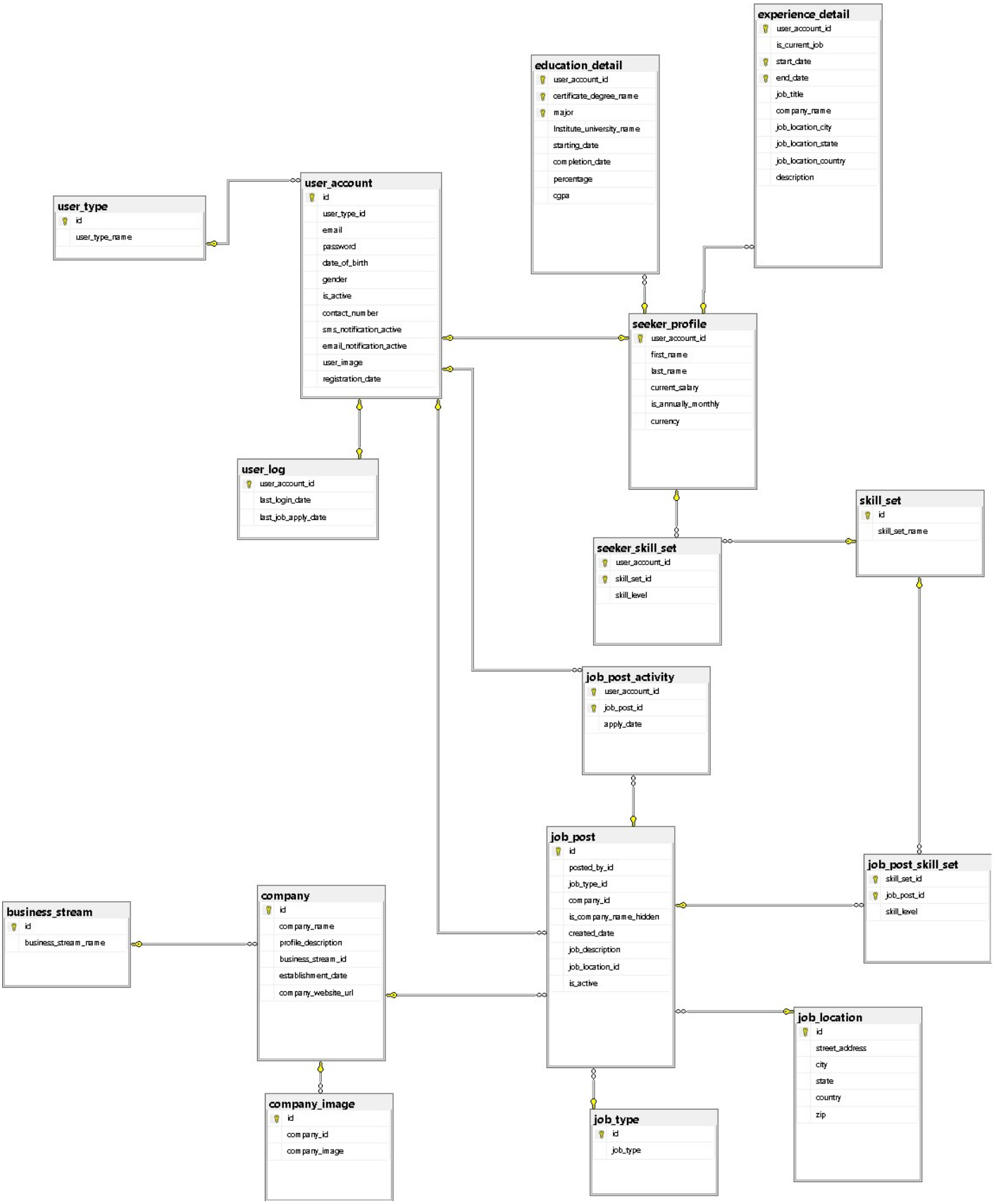

Рисунок А.2 – Фізична діаграма бази даних