Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для автоматизації онлайн-рекрутингу. Back-end
(тема)

Виконав:
студентка 4 курсу, групи ПЗПІ-20-4

Стрюкова Д.В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Побіженко І.О.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенці Стрюкової Дар’ї В’ячеславівни
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для автоматизації онлайн-рекрутингу. Back-end
Затверджена наказом по університету від 20.05.2024 № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 19.06.2024
3. Вихідні дані до роботи у програмній системі передбачити: Створення серверної частини на платформі .NET 6, публічний API, налаштування бази даних PostgreSQL, автоматизація рекрутингового процесу, захист системи. Використовувати Visual Studio 2022, PgAdmin, Docker Desktop.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	15.04.2024	*виконано*
3	Проектування ПЗ	24.04.2024	*виконано*
4	Розробка ПЗ	20.05.2024	*виконано*
5	Тестування ПЗ	30.05.2024	*виконано*
6	Оформлення пояснювальної записки	05.06.2024	*виконано*
7	Підготовка презентації та доповіді	06.06.2024	*виконано*
8	Попередній захист	09.06.2024	*виконано*
9	Нормоконтроль, рецензування	13.06.2024	*виконано*
10	Здача роботи у електронний архів	14.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	19.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка)_____ Стрюкова Д.В.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Побіженко І.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра: 72 стор., 18 рис., 7 джерел.

ДОШКА, ВАКАНСІЯ, КАЛЕНДАР, КАНДИДАТ, РЕЗЮМЕ, РЕКРУТИНГ, C#, ENTITY FRAMEWORK, IDENTITY, MONGODB, MULTITIER, POSTGRESQL.

Предметом вивчення є програмна система для автоматизації рекрутингового процесу. Об’єкт розробки має бути Rest API, який дозволить комунікацію між серверною та клієнтською частинами додатку.

Розробка серверної частини основується на мовах програмування C#, .NET та такими технологіями як MongoDB, POSTGRESQL, Entity Framework, Identity Framework.

У результаті роботи реалізовано API для автоматизації рекрутингового процесу. З програмою може взаємодіяти три ролі – адміністратор додатку, який має бути один, та безліч зареєстрованих користувачів: кандидати та рекрутери. Кожен користувач має пройти реєстрацію та авторизацію, щоб отримати доступ до окремих функціональних можливостей, які розділені за ролями.

Програмний продукт складається з серверної частини.

BOARD, VACANCY, CALENDAR, CANDIDATE, RESUME, RECRUITING, C#, ENTITY FRAMEWORK, IDENTITY, MONGODB, MULTITIER, POSTGRESQL.

The subject of study is a software system for automating the recruiting process. The object of development is a Rest API that allows communication between the server and client parts of the application.

The development of the server part is based on programming languages such as C#, .NET, and technologies such as MongoDB, PostgreSQL, Entity Framework, and Identity.

As a result of the work, an API for automating the recruiting process has been implemented. The program can interact with three roles - the application administrator, who should be unique, and an unlimited number of registered users: candidates and recruiters. Each user must register and authenticate to access specific functional capabilities, which are divided by roles.

The software application consists of a server part.

Я, Стрюкова Дар'я В'ячеславівна, студентка гр. ПЗПІ-20-4, здобувачка вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для автоматизації онлайн-рекрутингу. Back-end», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(а) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 7
ВСТУП 1
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ 9
1.1 Аналіз предметної галузі 9
1.2 Аналіз існуючих рішень 10
1.3 Постановка задачі 12
1.4 Виявлення проблем 14
2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ 16
2.1 Функціональні вимоги 16
2.2 Нефункціональні вимоги 17
3 АРХІТЕКТУРА І ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 19
3.1 UML проєктування ПЗ 19
3.2 Вибір архітектури 21
3.3 Бази даних 24
4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ 31
4.1 Налаштування основних частин серверу 31
4.2 Реалізація бізнес-логіки 32
4.3 Інтеграція з базами даних MongoDB та PostgreSQL 35
4.4 Розробка API та обробка запитів 39
5 ТЕСТУВАННЯ РОЗРОБЛЕННОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .. 42
5.1 Тестування API серверної частини 42
ВИСНОВКИ 46
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 47
ДОДАТОК А 48
ДОДАТОК Б 49
ДОДАТОК В 56
ДОДАТОК Г 72

ПЕРЕЛІК СКОРОЧЕНЬ

API - Application Programming Interface

SOLID - Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion

HTTPS - HyperText Transfer Protocol Secure

HR - Human Resources

JWT - JSON Web Token

REST - Representational State Transfer

ORM - Object-Relational Mapping

DTO - Data Transfer Object

BLL - Business Logic Layer

DAL - Data Access Layer

LINQ - Language Integrated Query

XML - Extensible Markup Language

ВСТУП

В сучасному світі, де конкуренція на ринку праці постійно зростає, програми для автоматизації онлайн-рекрутингу стають невід'ємною складовою ефективного управління кадрами. Впровадження таких систем стає необхідним для підтримки конкурентоспроможності підприємств у залученні та утриманні кваліфікованого персоналу.

Ця система має на меті полегшити та оптимізувати кожен етап рекрутингового процесу - від розміщення вакансій та відбору кандидатів до оцінки їхньої придатності та прийняття рішення щодо прийняття на роботу. Її основною метою є забезпечення ефективності, точності та прозорості у всіх аспектах рекрутингу.

Основна мета програми для автоматизації онлайн-рекрутингу полягає в створенні ефективного та ергономічного інструменту для всіх етапів процесу управління персоналом: від планування потреб у кадрах до виведення інформації про обраного кандидата. Вона повинна спрощувати та оптимізувати всі етапи рекрутингового процесу, забезпечуючи якісний та результативний відбір кадрів.

Система повинна бути легко інтегрованою з існуючими системами управління персоналом та бути зрозумілою для користувачів різного рівня технічної підготовки. Крім того, вона має бути гнучкою, щоб відповідати унікальним потребам та процесам кожної конкретної компанії.

Робота не є ізольованою, але має взаємозв'язок з іншими дослідженнями та розробками в сфері інформаційних технологій та управління бізнесом. Це дозволить використовувати найновіші розробки та досягнення для створення найбільш ефективної та сучасної програмної системи для автоматизації онлайн-рекрутингу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

В сучасному світі, де технології стають невід’ємною складовою в усіх сферах життя, рекрутинг не є винятком. Онлайн-рекрутинг стає все більш популярним і ефективним способом пошуку працівників для компаній у всьому світі. Програмна система для автоматизації онлайн-рекрутингу є ключовим інструментом у цьому процесі, допомагаючи організаціям знаходити, відбирати і залучати найбільш підходящих кандидатів для вакантних посад.

Однією з основних переваг програмних систем для автоматизації онлайн-рекрутингу є їх здатність зменшувати час, необхідний для проведення рекрутингового процесу. Ці системи забезпечують автоматизовані інструменти для розміщення вакансій, відбору резюме, проведення співбесід і оцінки кандидатів. Вони також надають аналітичні звіти, які допомагають підвищити ефективність рекрутингових стратегій компанії.

Ключовою функцією таких систем є їх здатність до інтеграції з іншими інструментами та платформами. Наприклад, такі системи можуть інтегруватися з соціальними мережами, сайтами для пошуку роботи, тестувальними системами та іншими ресурсами, щоб максимізувати охоплення аудиторії та забезпечити доступ до найбільш різноманітних кандидатів. Зокрема, важливою функцією є можливість автоматичної фільтрації резюме та аналізу кандидатів з використанням інструментів штучного інтелекту та машинного навчання. Це дозволяє рекрутерам швидше та ефективніше відбирати кандидатів, враховуючи важливі параметри, такі як досвід, навички, освіта та культурна відповідність.

Однак, існують певні виклики і проблеми, пов'язані з програмними системами для автоматизації онлайн-рекрутингу. Однією з них є необхідність забезпечення конфіденційності та захисту даних кандидатів, оскільки ці системи обробляють великий обсяг конфіденційної інформації. Іншою проблемою є ризик впливу попередніх упереджень або недоліків у алгоритмах машинного навчання на об'єктивність процесу відбору.

Загалом, програмні системи для автоматизації онлайн-рекрутингу є важливим інструментом для сучасних компаній, які прагнуть залучити та утримати найкращих кандидатів на вакантні посади. Однак важливо враховувати етичні та правові аспекти використання таких систем, а також постійно вдосконалювати їх функціонал з урахуванням змін у вимогах та технологіях.

1.2 Аналіз існуючих рішень

Систем автоматизації процесу прийому на роботу існує безліч, та всі вони дуже схожі за метою та принципом своєї роботи.

Найбільш поширені системи:

RECRUITEE [1] - це сучасний інструмент для автоматизації рекрутингу. Система надає можливість налаштувати автоматичні дії, створити шаблони для повторюваних структур конвеєрів (див. рис. 1.1).

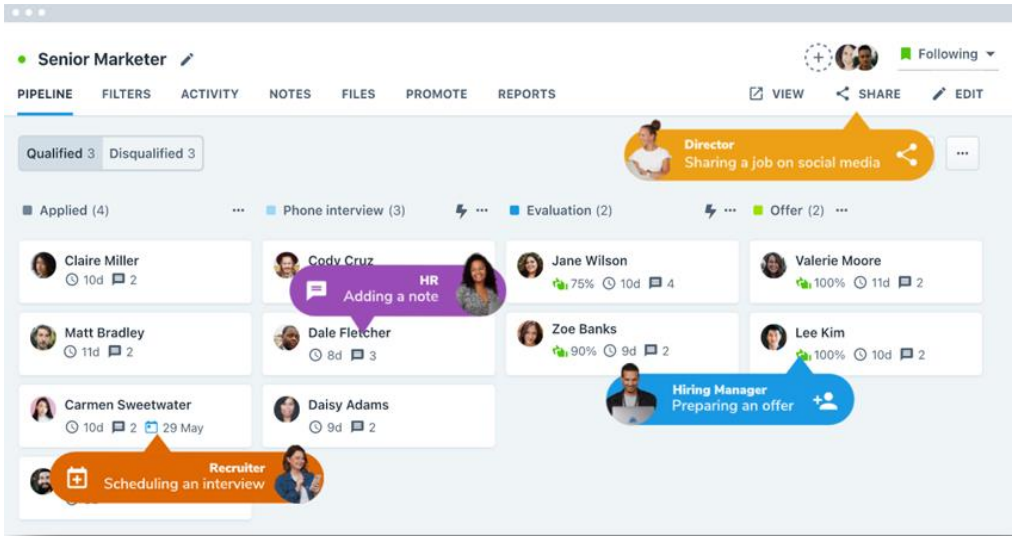


Рисунок 1.1 – Аналог RECRUITEE (за даними [1])

З переваг цієї програми можна виділити наступні:

— пробна безкоштовна підписка;

— шаблони дій, які будуть використовуватися постійно;

— створення графіків та аналіз роботи.

Недоліки:

— немає зв’язку з іншими системами, такими як LinkedIn та іншими, для

заохочення більшої кількості користувачів;

— немає прямої комунікації з клієнтами.

MANATAL [2] – додаток, призначений для швидшого пошуку та найму кандидатів, спеціально для команд відділу кадрів, кадрових агентств і хедхантерів (див. рис. 1.2).

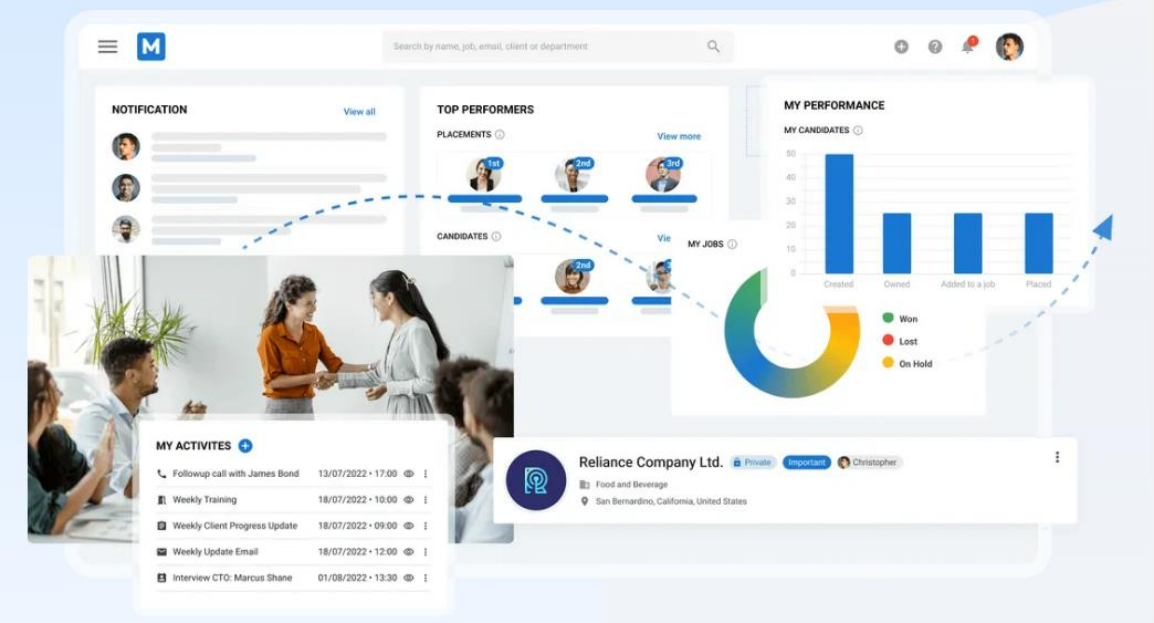


Рисунок 1.2 – Аналог MANATAL (за даними [2])

З переваг цієї програми можна виділити наступні:

— пробна безкоштовна підписка;

— зв’язок з іншими системами для заохочення більшої кількості користувачів;

— Manatal сканує опис посад і виділяє основні навички та вимоги, необхідні для кандидатів автоматично.

Недоліки:

— немає прямої комунікації з клієнтами, тобто не можна відправити лист з самої системи або інформувати клієнта про ваші наміри та дії.

Workday Recruiting: Workday є відомою платформою управління та розробки кадрів, яка включає в себе модуль рекрутингу. Їх програмне

забезпечення дозволяє рекрутерам ефективно керувати процесом відбору, включаючи публікацію вакансій, відбір резюме та взаємодію з кандидатами.

BambooHR: Ця платформа спрямована на малі та середні підприємства і надає інструменти для управління персоналом, включаючи модуль рекрутингу. BambooHR дозволяє автоматизувати багато аспектів рекрутингового процесу, від публікації вакансій до оцінки кандидатів.

iCIMS: Ця платформа спеціалізується на рекрутингу та управлінні талантами. Вона надає широкий спектр функцій, включаючи пошук кандидатів, відстеження прогресу вакансій та звітність.

Greenhouse: Greenhouse відомий своєю фокусуванням на рекрутингу та відборі кандидатів. Вони пропонують інструменти для ефективного керування вакансіями, проведення співбесід та оцінки кандидатів.

SmartRecruiters: Ця платформа ставить перед собою завдання спростити рекрутинговий процес за допомогою інтелектуальних інструментів та аналітики. Вони надають можливості для автоматизації багатьох аспектів відбору та розширюють можливості для взаємодії з кандидатами.

Кожен з цих конкурентів має свої сильні та слабкі сторони, і важливо вивчити їхні продукти та стратегії, щоб розробити конкурентоздатний продукт. Також варто звернути увагу на новаторські технології та тренди в цій галузі.

1.3 Постановка задачі

Мета дипломної роботи це розробка веб-додатку для різних типів користувачів з потребами пошуку роботи чи пришвидшення роботи HR відділу в великих компаніях, яким необхідні працівники. Мета роботи полягає в розробці серверної частини багатошарового веб-додатку на платформі .NET з використанням мови програмування C#. Додаток буде побудований за принципом Multitier [5] (багатошарова архітектура), що дозволить розділити функціональність на різні рівні і виконувати їх незалежно один від одного. Основними характеристиками проекту будуть:

— використання двох різних баз даних PostgreSQL та MongoDB дозволить оптимізувати роботу з даними залежно від їхнього типу та вимог проекту;
— використання Swagger дозволить автоматично створювати документацію API, що полегшить розробку та роботу з API;
— використання патернів проектування та принципів SOLID допоможе створити добре структурований, легко розширюваний та підтримуваний код;
— можливість взаємодії з сервером за допомогою контролерів та захищеної комунікації за протоколом HTTPS, забезпечуючи безпеку передачі даних.

В цілому, розробка додатку буде спрямована на створення високопродуктивного, масштабованого та зручного в управлінні веб-додатку з використанням сучасних технологій та кращих практик програмування. Основні функції системи включають:

— можливість додавання, редагування та видалення вакансій, створення списків обов'язків та вимог для кандидатів;
— зберігання та оновлення профілів кандидатів, пошук та фільтрація за різними критеріями, рейтингування та порівняння кандидатів;
— планування та керування співбесідами, створення розкладів, запрошення учасників;
— чат для спілкування з кандидатами та учасниками процесу, відправлення сповіщень та нагадувань.

Необхідно також забезпечити безпеку даних та конфіденційність інформації, а також швидку та ефективну роботу системи навіть при великій кількості записів та користувачів, застосовуючи сучасні методи шифрування та оптимізації баз даних.

1.3.1 Цільова аудиторія

Аналізуючи ринок, можна стверджувати, що система надасть зручний інтерфейс та усі потрібні інструменти для прискорення процесу прийому на роботу. Технології автоматизації рекрутингу потрібні компаніям, які прагнуть збільшити свою конкурентну перевагу при наймі. Ці компанії прагнуть досягти кращих результатів за менший проміжок часу та часто з меншими ресурсами порівняно з цілями, які вони намагаються досягти.

Можна виділити дві групи користувачів:

а) роботодавці (HRs), що потребують:

1) зручний пошук найманців;
2) зрозуміле відображення процесу найму;
3) зручний перегляд анкет та резюме;
4) можливість створення графіку та додавання шаблонних задач до нього;
5) можливість комунікації з клієнтами за допомогою шаблонних або власних листів, які автоматично відсилаються їм;
6) зручного документування та корегування інформації під час інтерв’ю;

б) найманці, що потребують:

1) зручне розміщення анкет та резюме;

в) адміністратори, що потребують:

1) можливість редагування даних додатку;

Платформа повинна бути якісною та зручною для всіх типів користувачів.

1.4 Виявлення проблем

Виявлення проблем та актуалізація рішень є критичним етапом для успішного управління проектом. Важливо розглянути кожну з проблем і знайти можливі шляхи вирішення:

— невиконання прогнозу доходів: потенційно, це може бути через недооцінку ринкових умов, зміни у споживчому попиті або недостатню

ефективність маркетингових стратегій. Необхідно переглянути прогнози, враховуючи нові тренди на ринку та зміни у попиті та підвищити ефективність маркетингу, звертаючи увагу на цільову аудиторію та приваблення нових клієнтів.

— нестача спонсорів для підтримки сервісу: недостатність фінансової підтримки може виникнути через неправильне просування проекту або недостатню привабливість для потенційних спонсорів. Необхідно підвищити видимість вашого проекту через маркетингові кампанії та співпрацю з партнерами, з'ясувати, які можливості спонсорства приваблять потенційних спонсорів та розробити привабливі пропозиції співпраці.
— реалізація недостатньо зрозумілої та зручної системи: зрозумілість або неефективність системи може призвести до низької придатності для користувачів та низької лояльності. Необхідно здійснити аудит системи та відгуків користувачів, щоб ідентифікувати слабкі місця.
— конкуренція на ринку аналогів: наявність багатьох аналогічних продуктів може ускладнити просування та здобуття позицій на ринку. Необхідно здійснити аналіз конкурентів та обрати стратегії диференціації, які допоможуть виділитися серед конкуренції.

Загалом, для успішного вирішення цих проблем важливо аналізувати ситуацію, впроваджувати відповідні стратегії та стежити за результатами.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Додаток для автоматизації онлайн-рекрутингу спрямований на надання компаніям зручного та ефективного інструменту для керування процесом найму персоналу. Він має бути оснащений рядом ключових функцій, що дозволять виконувати різноманітні завдання пов'язані з рекрутингом.

Однією з ключових функцій має бути дошка відображення процесу найму, аналогічна дошці у Jira. Ця функція спростить відстеження статусів вакансій та кандидатів, забезпечуючи зручне управління рекрутинговим процесом. Крім того, додаток повинен мати інтегрований календар, що дозволить планувати та відстежувати онлайн зустрічі з кандидатами та інші важливі події в рекрутинговому процесі. Це допоможе забезпечити ефективну організацію часу та координацію дій між учасниками процесу. Це спростить організацію та проведення співбесід та зустрічей, забезпечуючи зручну взаємодію з кандидатами. Також, додаток має дозволяти відправку електронних листів з системи на пошту, використовуючи шаблони для зручної комунікації з кандидатами та іншими учасниками процесу.

Програмна система також має забезпечувати розрахунок заробітної плати, що дозволить оцінювати витрати на найм нового персоналу та адаптувати пропозиції відповідно до бюджету компанії. Додаток має надавати можливість завантаження та збереження резюме кандидатів, проведення тестів для вакансій, спілкування з рекрутером у реальному часі через чат, а також генерацію документів, таких як контракти та листи пропозиції на основі даних кандидата.

Крім того, додаток має на меті забезпечувати детальну статистику по вакансіям, компаніям та кандидатам, а також можливість порівняння кандидатів та їх резюме. Звіти по вакансіям дозволять швидко аналізувати ефективність рекрутингових стратегій. Система також підтримуватиме відгуки від користувачів та підписку на вакансії з розсилкою імейлів. Розроблені алгоритми

також забезпечуватимуть релевантність вакансій вимогам кандидатів, що підвищує ефективність рекрутингового процесу.

2.2 Нефункціональні вимоги

Система повинна забезпечити високий рівень безпеки та конфіденційності даних кандидатів, тобто має використовувати методи шифрування даних, обмеження доступу до конфіденційної інформації лише авторизованим користувачам з відповідними правами доступу, використовувати JWT токени для авторизації.

Система має забезпечити швидке завантаження сторінок та ефективну роботу з базою даних, навіть при великій кількості записів. Система повинна використовувати оптимізовані запити до бази даних, кешування та інші техніки для забезпечення максимальної продуктивності системи.

Додаток має надавати можливість обміну даними з іншими системами та сервісами через різні інтерфейси комунікації. Це включає в себе використання стандартних протоколів передачі даних, таких як HTTP/HTTPS для взаємодії з веб-серверами та REST для спілкування з іншими веб-службами.

Багатошарова архітектура є ключовим принципом проектування програмних систем, який полягає в розділенні функціональності програми на логічні шари або рівні.

- розділення відповідальності: Multitier дозволяє розділити функціональність програми на різні логічні шари, такі як презентаційний, бізнес-логіки та доступ до даних. Це дозволяє кожному шару виконувати свою конкретну відповідальність, забезпечуючи чітку структуру та зменшуючи складність коду;
- масштабованість: розділення системи на різні шари дозволяє масштабувати окремі компоненти незалежно один від одного. Це робить можливим горизонтальне масштабування;
- підвищення модульності та перевикористання коду: розділення системи

на компоненти дозволяє підвищити модульність програми та зробити її більш гнучкою для змін. Кожен шар може бути реалізований як окремий модуль, що сприяє повторному використанню коду та спрощує процес тестування та розробки;

— забезпечення безпеки: Multitier дозволяє визначити границі доступу до різних рівнів функціональності, що дозволяє забезпечити безпеку даних та захист від несанкціонованого доступу.

Складові багатошарової архітектури включають:

— API шар: цей шар відповідає за взаємодію з іншими додатками та сервісами через програмні інтерфейси. Він приймає запити від зовнішніх систем, передає їх до бізнес-логіки, а потім повертає результати назад до запитуючих систем. API шар забезпечує безпечний і структурований спосіб доступу до функціональності програми, забезпечуючи одночасно рівень абстракції від внутрішніх деталей реалізації;

— бізнес-логіка (Business Logic) шар: цей шар містить бізнес-правила та логіку програми. Він відповідає за обробку та аналіз даних, взаємодію з базою даних та виконання бізнес-логіки додатку;

— доступ до даних (Data Access) шар: цей шар забезпечує доступ до даних та баз даних, включаючи зчитування, запис, видалення та оновлення даних. Він може включати об'єктно-реляційне відображення (ORM), запити SQL та інші методи взаємодії з даними.

Для даного додатку використання Multitier архітектури дозволить створити добре організовану та легко розширювану систему, яка забезпечить ефективну роботу з даними та високий рівень безпеки.

3 АРХІТЕКТУРА І ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Діаграми є ефективним інструментом для спілкування між членами команди розробки, клієнтами та іншими зацікавленими сторонами. Вони надають загальний засіб вираження ідей, вимог та концепцій проекту, що дозволяє всім зацікавленим сторонам зрозуміти його обсяг та цілі. Діаграми допомагають при розробці архітектури програмного забезпечення, визначаючи його складові частини, взаємозв'язки та структуру. Вони дозволяють розробникам та архітекторам зробити правильний вибір технологій, вирішити архітектурні проблеми та забезпечити ефективність та масштабованість системи.

3.1 UML проєктування ПЗ

Use Case діаграми допомагають описати функціональність системи з точки зору користувача. Вони відображають, як користувачі взаємодіють з системою та як система реагує на їхні дії. Use case діаграми допомагають уточнити вимоги до системи та визначити основні функціональні можливості. Було реалізовано діаграму (див. рис. 3.1), на якій зображена взаємодія різних видів користувачів з системою.

Незареєстрований користувач додатку не має доступу до нього, тому в нього є можливість тільки пройти етап реєстрації, заповнити всі дані про себе та обрати роль, за якою він хоче бути зареєстрований: кандидат або рекрутер. Після цього новому користувачу треба авторизуватися. Тепер він матиме можливість до таких функцій додатку, як перегляд всього списку резюме та вакансій системи, а також відфільтрувати та виконати пошук за цими даними; переглянути свій профіль та відредагувати його дані.

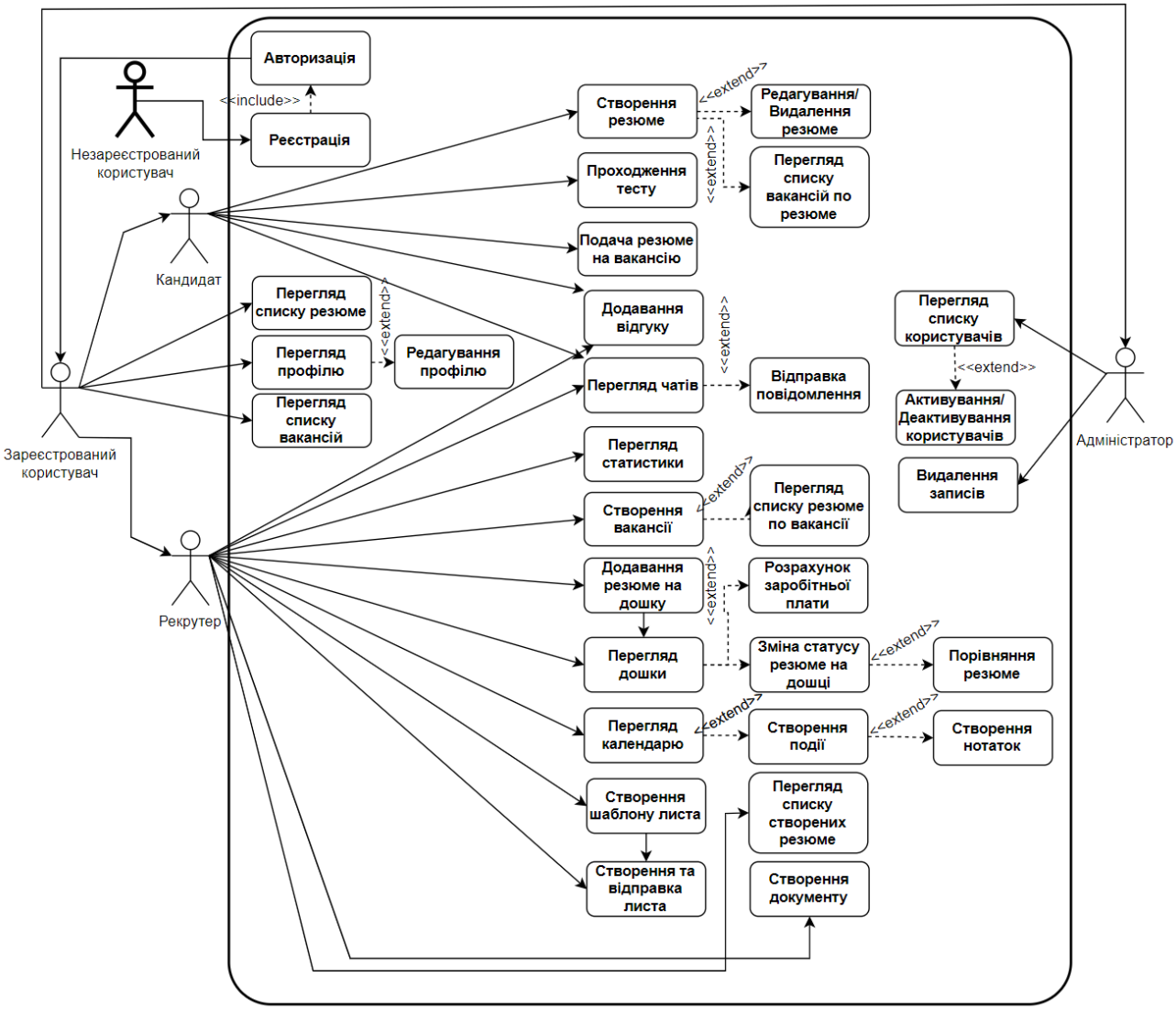


Рисунок 3.1 – Діаграма Use Case (рисунок виконаний самостійно)

Всі інші функціональні можливості доступні тільки окремим ролям користувачів. Кандидат може створити резюме або знайти в пошуку потрібні вакансії та додати своє резюме напряму туди. Кожне резюме може бути видалено та відредаговано, а також воно має весь список пов’язаних вакансій, які може побачити кандидат. Частиною резюме може бути тестування, тому кандидат має доступ до його проходження.

Кандидат та рекрутер можуть спілкуватися використовуючи чат додатку, а також мають можливість залишати відгуки вакансіям, резюме та іншим користувачам.

Рекрутеру доступні такі функціональні можливості як створення та редагування вакансій, перегляд пов’язаних до вакансії резюме, перегляд статистики по всім даним додатку, додавання резюме на дошку та обробку всіх резюме на ній. Також функціональність дошки дозволяє порівняти дані резюме та кандидатів та розрахувати заробітну плату найманця. Окрім чату є можливість у рекрутера відправити імейл іншому користувачеві одразу із системи, а також створити шаблони імейлів для подальшого користування. Додатково він може створювати події в календарі та додавати нотатки до них.

Лише адміністратор має доступ до списку всіх користувачів та може змінювати налаштування доступу для всіх інших.

Розподілення прав доступу та обов'язків за ролями допомагає забезпечити безпеку системи. Користувачам надаються лише ті права, які необхідні для виконання їхніх функцій, що дозволяє уникнути неповноважного доступу до конфіденційної інформації чи можливих порушень безпеки.

3.2 Вибір архітектури

При розгляді вибору архітектури для програмної системи для автоматизації онлайн-рекрутингу, деякі важливі аспекти потрібно розглянути, такі як масштабованість, надійність, продуктивність, безпека та вартість.

Взаємодію та склад частин програмного продукту можна розглянути на діаграмі (див. рис. 3.2).

Багаторівнева архітектура є однією з найпоширеніших архітектурних концепцій у розробці програмного забезпечення. Вона дозволяє відокремити функціональність програми на окремі рівні абстракції, що спрощує розробку, розподілення обов'язків та підтримку коду.

API відповідає за взаємодію з клієнтами та обробку їхніх запитів. Цей рівень включає контролери, валідатори, профілі маппінгу, розширення, DTO (Data Transfer Objects) та логування. Контролери приймають запити від клієнтів, використовують JWT токени для авторизації та викликають відповідні сервіси.

Валідатори перевіряють коректність вхідних даних. Профілі маппінгу використовуються для мапування між DTO та доменними об'єктами. Розширення додають додаткову функціональність до контролерів або інших компонентів. DTO використовуються для передачі даних між різними частинами системи.

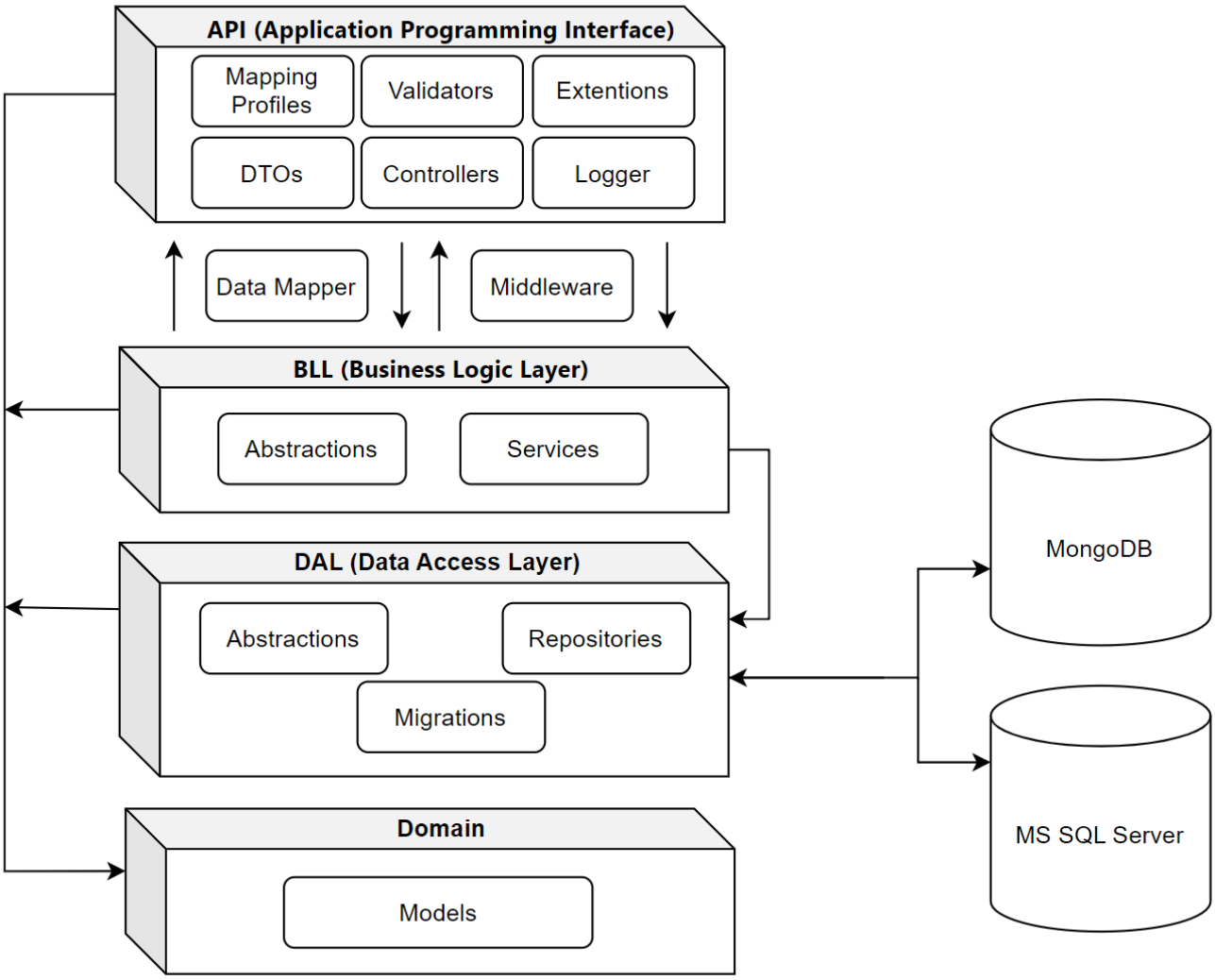


Рисунок 3.2 – Архітектура серверної частини додатку (рисунок виконаний самостійно)

Логування відповідає за запис подій та інформації про виконання додатку. API є центральною точкою взаємодії з системою, тому він є ключовим компонентом багаторівневої архітектури. Це дозволяє відокремити бізнес-логіку від інтерфейсу користувача та забезпечує більшу гнучкість та модульність.

BLL (Business Logic Layer) містить бізнес-логіку та сервіси, які виконують операції з даними та реалізують бізнес-правила. Цей рівень включає абстракції та сервіси. Абстракції визначають контракти для реалізації бізнес-логіки, а сервіси забезпечують реалізацію цих контрактів. BLL відокремлює бізнес-логіку від інших компонентів системи, таких як API та DAL. Це дозволяє зберегти чистоту коду, полегшує тестування та забезпечує однорідний доступ до функціональності бізнесу.

DAL (Data Access Layer) відповідає за доступ до даних та їх збереження. У вашій архітектурі цей рівень включає абстракції, репозитарії та міграції. Абстракції визначають контракти для доступу до даних, репозитарії забезпечують реалізацію цих контрактів та взаємодію з базами даних, а міграції відповідають за збереження структури баз даних.

Domain містить моделі домену, які представляють об'єкти та концепції, специфічні для бізнес-домену. Ці моделі відтворюють структуру даних та бізнес-правила в системі. Domain допомагає відокремити моделі домену від інших компонентів системи, таких як API та DAL.

API викликає методи BLL для обробки запитів та звертається до DAL для доступу до даних. BLL використовує моделі домену для виконання бізнес-логіки та взаємодіє з DAL для збереження та отримання даних. DAL надає доступ до бази даних та взаємодіє з нею для виконання операцій з даними.

Отже, багаторівнева архітектура дозволяє відокремити функціональність системи на різні рівні абстракції, що сприяє полегшенню розробки, підтримці та розширенню програмного забезпечення. Кожен рівень відповідає за свою частину логіки, що дозволяє зменшити взаємозалежність між компонентами та підвищити модульність системи. Це також сприяє покращенню тестування, оскільки можна перевіряти окремі компоненти незалежно один від одного. Такий підхід підвищує загальну надійність та гнучкість системи, забезпечуючи можливість її адаптації до нових вимог.

3.3 Бази даних

ER-діаграма (Entity-Relationship Diagram) є ключовим інструментом для аналізу та проектування баз даних, оскільки дозволяє визначити сутності, їх атрибути та зв'язки між ними.

Для реалізації було обрано два види баз даних. PostgreSQL є потужною та надійною реляційною системою керування базами даних, яка широко використовується для побудови реляційних баз даних. Вона має розширені можливості для роботи зі складними структурами даних, транзакціями та безпекою. MongoDB [7] є популярною NoSQL базою даних, яка забезпечує гнучку та масштабовану зберігання даних у вигляді документів. Вона особливо корисна для зберігання даних зі змінною структурою або великим обсягом даних. PostgreSQL може бути використаний для зберігання даних про користувачів та інші об’єкти системи, тоді як MongoDB може використовуватися для зберігання реалізації чату додатку. Це дозволить ефективно керувати як структурованими даними, так і даними зі змінною структурою, що характерно для чатових додатків.

Основна частина, реалізована за допомогою реляційної бази даних, матиме таку структуру:

- AspNetRoles: Id (унікальний ідентифікатор ролі), Name (назва ролі), NormalizedName (нормалізована назва ролі), ConcurrencyStamp (для контролю конкурентних змін);
- AspNetUsers: Id (унікальний ідентифікатор користувача), FirstName (ім'я користувача), LastName (прізвище користувача), BirthDate (дата народження), Country (країна проживання), City (місто проживання), UserName (ім'я користувача для входу), NormalizedUserName (нормалізоване ім'я користувача для входу), Email (електронна пошта користувача), NormalizedEmail (нормалізована електронна пошта користувача), EmailConfirmed (підтвердження електронної пошти), PasswordHash (хеш пароля користувача), SecurityStamp (для

забезпечення безпеки);

- AspNetUserRoles: UserId (ідентифікатор користувача), RoleId (ідентифікатор ролі);
- Recruiters: Id (унікальний ідентифікатор рекрутера), UserId (ідентифікатор користувача, пов'язаний з рекрутером), CalendarId (ідентифікатор календаря рекрутера), DashboardId (ідентифікатор дошки рекрутера), Company (назва компанії рекрутера);
- Calendars: Id (унікальний ідентифікатор календаря), OwnerId (ідентифікатор власника календаря);
- Dashboards: Id (унікальний ідентифікатор дошки), OwnerId (ідентифікатор власника дошки);
- EmailTemplates: Id (унікальний ідентифікатор шаблону електронного листа), Theme (тема шаблону), RecruiterId (ідентифікатор рекрутера, що створив шаблон), Text (текст шаблону);
- EventTemplates: Id (унікальний ідентифікатор шаблону події), Theme (тема шаблону), MeetingLink (посилання на зустріч, якщо є), RecruiterId (ідентифікатор рекрутера, що створив шаблон), Text (текст шаблону);
- Mails: Id (унікальний ідентифікатор листа), Text (текст листа), FromRecruiterId (ідентифікатор рекрутера, що відправив лист), ToUserEmail (адреса електронної пошти користувача, що отримав лист), SentDateTime (дата та час відправлення листа);
- Events: Id (унікальний ідентифікатор події), StartDateTime (дата та час початку події), EndDateTime (дата та час закінчення події), NoteId (ідентифікатор нотатки, пов'язаної з подією, якщо є), CalendarId (ідентифікатор календаря, до якого належить подія), Theme (тема події), MeetingLink (посилання на зустріч, якщо є);
- Candidates: Id (унікальний ідентифікатор кандидата), UserId (ідентифікатор користувача, пов'язаний з кандидатом), DashboardId (ідентифікатор дошки, пов'язаної з кандидатом, якщо є), Status (статус

кандидата);

- Notes: Id (унікальний ідентифікатор нотатки), EventId (ідентифікатор події, пов'язаної з нотаткою), CreatedDateTime (дата та час створення нотатки), NoteText (текст нотатки);
- Applications: Id (унікальний ідентифікатор заявки), ApplicantId (ідентифікатор кандидата, що подав заявку), Status (статус заявки), ApplicationText (текст заявки), CreatedDateTime (дата та час створення заявки), JobPosition (посада, на яку подається заявка), JobDepartment (відділ, до якого відноситься посада), Location (місце розташування роботи), Email (електронна пошта заявника), PhoneNumber (номер телефону заявника), FullTime (чи повна зайнятість), YearsOfExperience (роки досвіду), Hobbies (захоплення), Languages (мови, якими володіє заявник), Skills (навички заявника);
- Vacancies: Id (унікальний ідентифікатор вакансії), Title (назва вакансії), Description (опис вакансії), Location (місце розташування роботи), JobPosition (посада), JobDepartment (відділ, до якого відноситься посада), IsRemote (чи вакансія віддалена), MinSalary (мінімальна зарплата, якщо вказана), MaxSalary (максимальна зарплата, якщо вказана), CompanyName (назва компанії, що пропонує вакансію), RecruiterId (ідентифікатор рекрутера, який розмістив вакансію);
- ApplicationVacancy: ApplicationsId (ідентифікатор заявки), VacanciesId (ідентифікатор вакансії);
- Reviews: Id (унікальний ідентифікатор відгуку), ReviewText (текст відгуку), UserId (ідентифікатор користувача, який написав відгук), EntityType (тип сутності, до якої відноситься відгук), EntityId (ідентифікатор сутності, до якої відноситься відгук);
- QuizResults: Id (унікальний ідентифікатор результату тестування), ResumeId (ідентифікатор резюме, пов'язаний з результатом тестування);
- Quizzes: Id (унікальний ідентифікатор тесту), Title (назва тесту),

Description (опис тесту), VacancyId (ідентифікатор вакансії, пов'язаний з тестом);

— Questions: Id (унікальний ідентифікатор питання), Title (текст питання), QuizId (ідентифікатор тесту, до якого належить питання);

— Answers: Id (унікальний ідентифікатор відповіді), Title (текст відповіді), IsCorrect (прапорець, що вказує на правильність відповіді), QuestionId (ідентифікатор питання, пов'язаний з відповіддю);

— AnswerQuizResult: AnswersId (ідентифікатор відповіді), QuizResultsId (ідентифікатор результату тестування, до якого належить відповідь).

Структура бази даних представлена на рисунку 3.3.

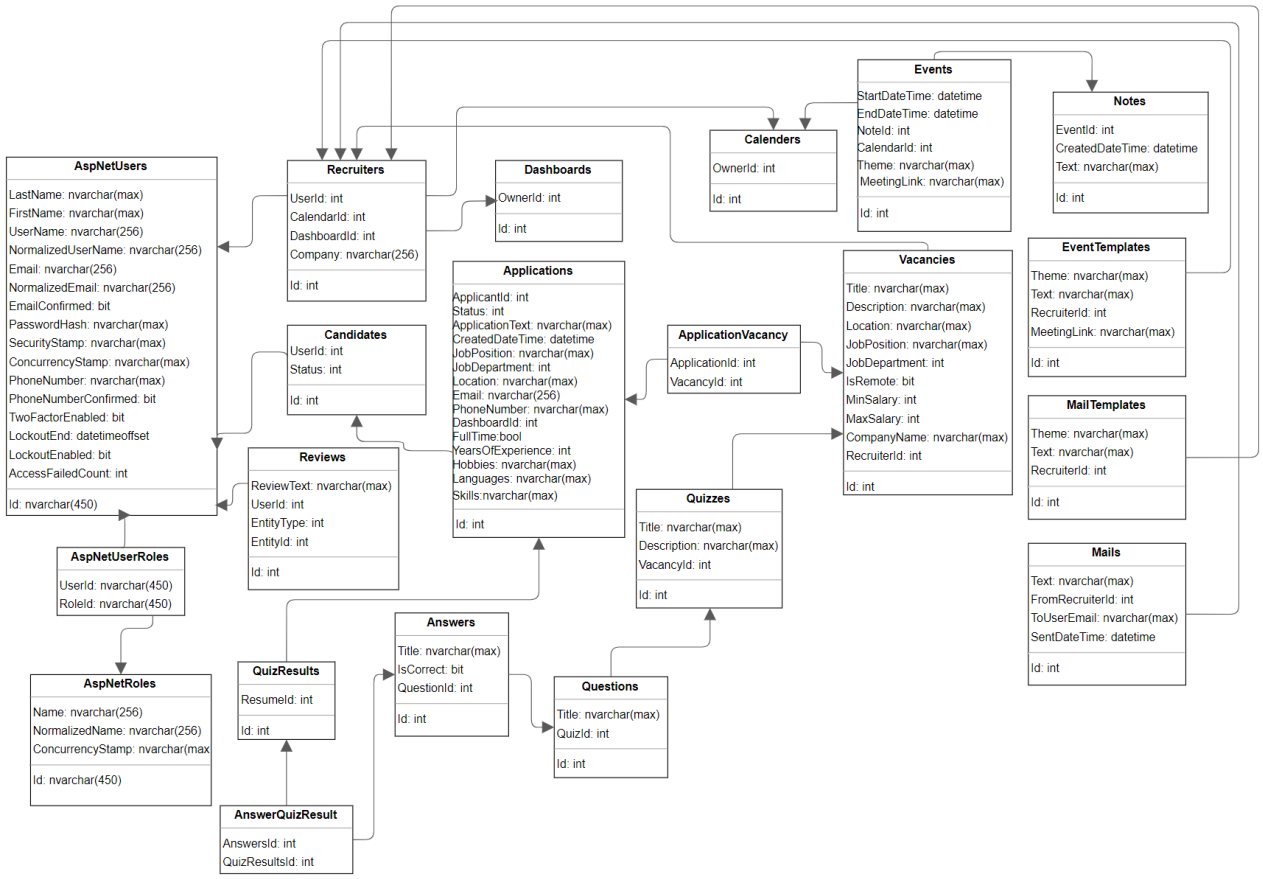


Рисунок 3.3 – ER діаграма (рисунок виконаний самостійно)

Для реалізації зв’язку many-to-many використовується проміжна таблиця ApplicationVacancy. Це дозволить системі і користувачам мати можливість

додавати до вакансій не обмежену кількість резюме, так само як і навпаки кожне резюме може бути подано на не обмежену кількість вакансій.

Ця структура бази даних розроблена для забезпечення функціональності системи рекрутингу, включаючи збереження даних про користувачів, рекрутерів, кандидатів, вакансії, події, електронні листи, шаблони, нотатки та зв'язки між ними.

У проекті для роботи з SQL базою даних використовується Entity Framework, Identity Framework та підхід Code First разом із міграціями. Цей вибір зроблений з урахуванням кількох факторів, які визначають ефективність, зручність та безпеку роботи з даними в програмній системі для автоматизації онлайн-рекрутингу.

Entity Framework є потужним інструментом для роботи з базами даних у середовищі .NET. Він надає різноманітні можливості для взаємодії з даними, включаючи створення моделей даних, виконання запитів та керування зв'язками між об'єктами. Використання Entity Framework дозволяє спростити розробку, забезпечуючи високий рівень абстракції над базою даних і дозволяючи розробникам концентруватися на бізнес-лозіці програми.

Identity Framework використовується для реалізації аутентифікації та авторизації користувачів у системі. Це рішення надає готовий функціонал для роботи зі збереженням даних користувачів, їхніх ролей та дозволів. Використання Identity дозволяє легко і безпечно управляти доступом користувачів до різних ресурсів системи.

Вибір підходу Code First означає, що база даних створюється на основі класів моделей даних. Це дає можливість розробникам визначити структуру даних у вигляді об'єктів коду, замість ручного створення та оновлення схеми бази даних. Крім того, Code First дозволяє зберігати версії бази даних в системі контролю версій, що полегшує управління змінами та спільною роботу над проектом.

Механізм міграцій, який надає Entity Framework, дозволяє автоматизувати процеси створення та оновлення структури бази даних на основі змін в коді. Це дозволяє зручно втілювати зміни в схемі бази даних без необхідності втручання в SQL скрипти. Механізм міграцій спрощує роботу розробників та зменшує ймовірність помилок при впровадженні змін в структуру бази даних.

Узагальнюючи, використання Entity Framework, Identity Framework, підходу Code First разом із механізмом міграцій в проекті дозволяє забезпечити зручну, безпечну та ефективну роботу з даними, що є важливим аспектом в розробці програмної системи для автоматизації онлайн-рекрутингу. Структура документу представлена на рисунку 3.4.

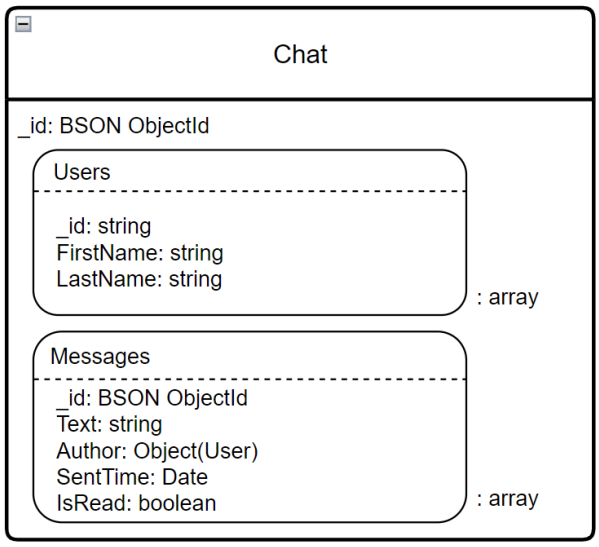


Рисунок 3.4 – Документ чату (рисунок виконаний самостійно)

Також структура документу в MongoDB:

— _id: це унікальний ідентифікатор документу в колекції. Використовується BSON ObjectId;
— Users: це масив об'єктів, кожен з яких містить інформацію про користувача чату. Кожен об'єкт має такі поля:_id (унікальний ідентифікатор користувача чату), FirstName (ім'я користувача), LastName (прізвище користувача);
— Messages: це масив об'єктів, кожен з яких представляє повідомлення в

чаті. Кожен об'єкт має такі поля: _id (унікальний ідентифікатор повідомлення), Text (текст повідомлення), Author (об'єкт, що містить інформацію про автора повідомлення), SentTime (час відправлення повідомлення), IsRead (прапорець, що вказує на те, чи було прочитане повідомлення).

MongoDB дозволяє зберігати документи з різною структурою в одній колекції. Це дозволяє зберігати повідомлення з різними полями, наприклад, текстові повідомлення, зображення, відео тощо, без необхідності використання реляційних зв'язків. Гнучкість MongoDB також дозволяє швидко адаптуватися до змін у вимогах бізнесу, додаючи нові поля до документів без впливу на існуючі дані. Така структура є ідеальною для динамічних додатків, де схема даних може часто змінюватися. Крім того, MongoDB забезпечує високу продуктивність і масштабованість завдяки горизонтальному розподілу даних. Завдяки вбудованій підтримці індексів і пошуку, MongoDB може ефективно обробляти великі обсяги даних. Ще однією перевагою є можливість використовувати вбудовані функції агрегації, що дозволяє виконувати складні запити і аналітику без необхідності переміщення даних до окремого аналітичного інструменту.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Налаштування основних частин серверу

У проекті використовуються кілька важливих технологій та бібліотек для забезпечення ефективної роботи з базою даних, аутентифікації користувачів, валідації даних, логування та документації API.

Для реалізації аутентифікації та авторизації використовується JWT (JSON Web Tokens). Конфігурація JWT здійснюється за допомогою налаштування JwtBearer аутентифікації. Коли користувач входить у систему, сервер перевіряє його облікові дані. Якщо вони правильні, сервер створює JWT, що містить деяку інформацію про користувача та підписує його своїм секретним ключем. Кожен раз, коли клієнт робить запит до сервера на захищений ресурс, він додає JWT до заголовку авторизації запиту. Сервер отримує токен і перевіряє його валідність та підпис, розшифровуючи корисне навантаження. Якщо токен валідний, сервер обробляє запит та надає доступ до запитуваного ресурсу. Якщо токен недійсний або відсутній, сервер повертає помилку авторизації.

```
builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
    options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.SaveToken = true;
    options.TokenValidationParameters = new
TokenValidationParameters()
    {
        ValidateIssuerSigningKey = true,
        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["JwtSe
ttings:Secret"])),
        ValidateIssuer = false,
        ValidateAudience = false,
        RequireExpirationTime = false,
    };
});
```

Ми налаштовуємо схему аутентифікації за замовчуванням як JwtBearer та визначаємо параметри валідації токена. Використовуємо симетричний секретний ключ для підпису та перевірки токенів.

Для валідації даних використовується бібліотека FluentValidation, яка забезпечує зручний та розширюваний спосіб валідації вхідних даних. Валідація конфігурується у контролерах, що дозволяє автоматично перевіряти дані, передані у запитах.

```
builder.Services.AddControllers()
    .AddFluentValidation(fv =>
    {
        fv.ImplicitlyValidateChildProperties = true;
        fv.ImplicitlyValidateRootCollectionElements = true;

fv.RegisterValidatorsFromAssembly(Assembly.GetExecutingAssembly());
    })
    .AddJsonOptions(options =>
    {
        options.JsonSerializerOptions.ReferenceHandler =
ReferenceHandler.IgnoreCycles;
    });
```

AutoMapper використовується для мапінгу об'єктів між різними рівнями додатку (наприклад, між DTO та моделями домену). Це забезпечує чистоту коду та спрощує процес трансформації даних.

4.2 Реалізація бізнес-логіки

Шар бізнес-логіки (BLL) є ключовим компонентом архітектури додатку, відповідальним за реалізацію бізнес-правил і логіки, що керує операціями та процесами в системі. У даному проекті BLL включає абстракції та реалізації, які взаємодіють з рівнями даних та презентації для забезпечення цілісності та функціональності бізнес-процесів. Абстракції, тобто інтерфейси, визначають контракт для сервісів бізнес-логіки, який включає методи, що повинні бути реалізовані в конкретних класах сервісів (див. рис. 4.1).

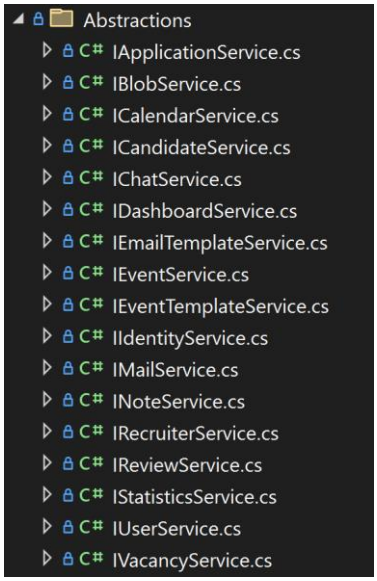


Рисунок 4.1 – Абстрактні класи сервісів (рисунок виконаний самостійно)

Реалізації, тобто конкретні класи, реалізують методи, визначені в інтерфейсах, і виконують основні операції бізнес-логіки (див. рис. 4.2).

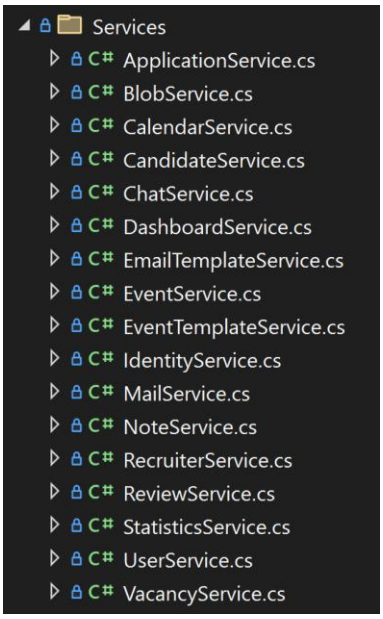


Рисунок 4.2 – Реалізація абстракцій (рисунок виконаний самостійно)

Це забезпечує гнучкість і можливість легкого заміщення реалізацій без зміни коду, що використовує ці інтерфейси.

Використання методу контракту і реалізації дозволяє створити добре структуровану та підтримувану систему. Наприклад, існує контракт у вигляді

інтерфейсу IMailService, який визначає методи для роботи з поштовими повідомленнями, і його реалізація MailService, яка містить конкретну реалізацію цих методів. Один з методів цього сервісу — Save, який демонструє, як реалізована бізнес-логіка в цьому проекті (див. рис. 4.3).

```
public async Task<Mail> Save(Mail mailToCreate)
{
    var recruiter = await _recriuterService.Get(mailToCreate.FromRecruiterId);

    var sender = new SmtpSender(() => new SmtpClient("smtp.gmail.com", 587)
    {
        Credentials = new NetworkCredential(login, password),
        EnableSsl = true,
        DeliveryMethod = SmtpDeliveryMethod.Network,
        PickupDirectoryLocation = directory
    });

    Email.DefaultSender = sender;

    var email = await Email
        .From(fromEmail)
        .To(mailToCreate.ToUserEmail)
        .Subject("Recrutire")
        .Body(mailToCreate.Text)
        .SendAsync();

    var mail = await _mailRepository.Create(mailToCreate);

    if (mail == null)
    {
        _logger.LogInformation($"Mail was not saved");
    }
    else
    {
        _logger.LogInformation($"Mail with id = {mail.Id} was saved");
    }

    return mail;
}
```

Рисунок 4.3 – Метод Save у класі MailService (рисунок виконаний самостійно)

Метод Save у класі MailService реалізує функціональність для відправки електронного листа і збереження інформації про цей лист у базі даних. Для цього налаштовується SMTP-клієнт для відправки електронних листів через сервер Gmail. Також використовується клас SmtpSender, який є частиною бібліотеки FluentEmail. SMTP-клієнт аутентифікується за допомогою облікових даних і використовує SSL для безпечного з'єднання. Після відправки листа інформація про нього зберігається в базі даних за допомогою репозитарію пошти.

Прикладом бізнес-логіки також є реалізації статистичних даних. Для отримання статистики подій і вакансій використовуються методи EventStatistics і VacancyStatistics (див. рис. 4.4). Ці методи демонструють, як можна агрегувати

і обробляти дані для отримання аналітичної інформації, необхідної для підтримки бізнес-процесів.

```
public async Task<IEnumerable<EventStatisticsModel>> EventStatistics(int recruiterId)
{
    var calendar = (await _calendarRepository.GetAll(
        calendar => calendar.OwnerId == recruiterId,
        calendar => calendar.Events)).FirstOrDefault();

    if (calendar == null)
    {
        return new List<EventStatisticsModel>();
    }

    var endDate = DateTime.UtcNow;
    var startDate = endDate.AddDays(-(12 * 7)); // 12 weeks * 7 days

    var filteredEvents = calendar.Events
        .Where(e => e.StartDateTime.Date >= startDate && e.StartDateTime.Date <= endDate)
        .ToList();

    var weeks = Enumerable.Range(0, 12)
        .Select(i => StartOfWeek(endDate, DayOfWeek.Monday).AddDays(-7 * i))
        .OrderBy(weekStart => weekStart)
        .ToList();

    var eventStatistics = weeks
        .Select(weekStart => new EventStatisticsModel
        {
            Name = weekStart.ToString("yyyy-MM-dd"),
            Series = Enumerable.Range(0, 7)
                .Select(offset => weekStart.AddDays(6 - offset))
                .Select(date => new EventStatisticsChild
                {
                    Date = date,
                    Name = date.DayOfWeek.ToString(),
                    Value = (int)Math.Round(filteredEvents.Where(e => e.StartDateTime.Date == date.Date)
                            .Sum(e => (e.EndDateTime - e.StartDateTime).TotalHours),
                        MidpointRounding.AwayFromZero)
                }).ToList()
        }).ToList();

    _logger.LogInformation($"Recruiter event statistics was got for the recruiter with id = {recruiterId}");
    return eventStatistics;
}
```

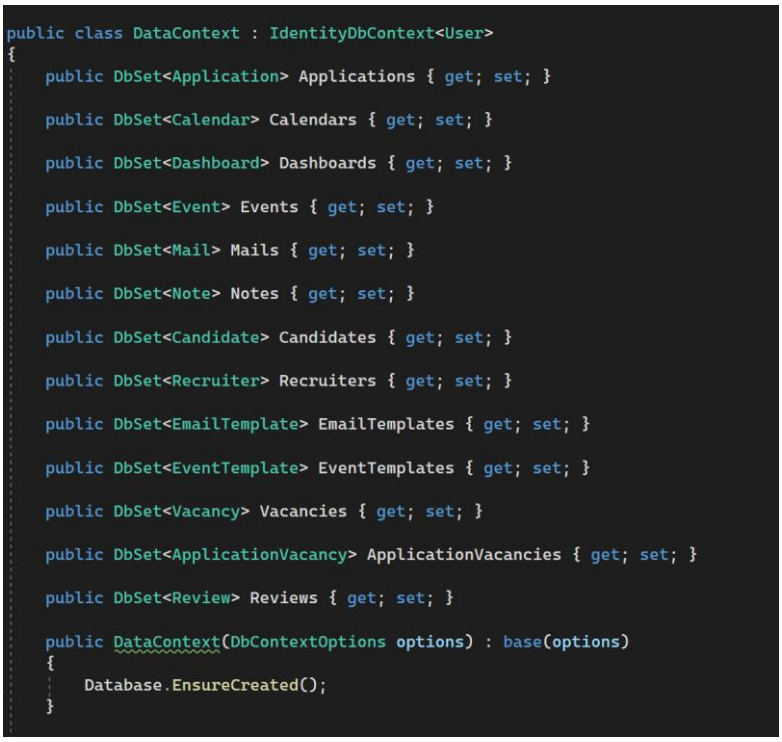
Рисунок 4.4 – Реалізація статистики (рисунок виконаний самостійно)

Метод EventStatistics призначений для отримання статистики подій рекрутера за останні 12 тижнів. Він обробляє дані з бази даних, групує їх по тижнях та обчислює кількість годин, витрачених на події кожного дня тижня. Це дозволяє рекрутерам отримати вичерпну картину своєї діяльності за вибраний період. Метод використовує репозитарії для доступу до даних, LINQ для фільтрації та групування даних, а також асинхронні операції для ефективної роботи з базою даних.

4.3 Інтеграція з базами даних MongoDB та PostgreSQL

Для роботи з базою даних використовується Entity Framework Core — сучасна об'єктно-реляційна модель надає розробникам можливість працювати з базою даних. Для керування базою даних використовується контекст даних

DataContext, що дозволяє використовувати вбудовані можливості аутентифікації та авторизації, надані ASP.NET Identity (див. рис. 4.5).



```
public class DataContext : IdentityDbContext<User>
{
    public DbSet<Application> Applications { get; set; }

    public DbSet<Calendar> Calendars { get; set; }

    public DbSet<Dashboard> Dashboards { get; set; }

    public DbSet<Event> Events { get; set; }

    public DbSet<Mail> Mails { get; set; }

    public DbSet<Note> Notes { get; set; }

    public DbSet<Candidate> Candidates { get; set; }

    public DbSet<Recruiter> Recruiters { get; set; }

    public DbSet<EmailTemplate> EmailTemplates { get; set; }

    public DbSet<EventTemplate> EventTemplates { get; set; }

    public DbSet<Vacancy> Vacancies { get; set; }

    public DbSet<ApplicationVacancy> ApplicationVacancies { get; set; }

    public DbSet<Review> Reviews { get; set; }

    public DataContext(DbContextOptions options) : base(options)
    {
        Database.EnsureCreated();
    }
```

Рисунок 4.5 – Контекст даних DataContext (рисунок виконаний самостійно)

Для створення та управління базою даних використовується підхід Code-First, що дозволяє визначати модель даних за допомогою класів C# та автоматично створювати відповідні таблиці у базі даних.

Розробка починається з визначення класів моделей, які представляють таблиці бази даних. У коді класи моделей визначають структуру бази даних. EF Core забезпечує інструменти для створення та застосування міграцій, які автоматично оновлюють структуру бази даних відповідно до змін у моделях. Це дозволяє легко підтримувати та розвивати структуру бази даних без необхідності писати SQL-запити вручну.

В даному проекті використовуються різні типи зв'язків між таблицями: один-до-одного, один-до-багатьох та багато-до-багатьох. Приклад налаштування зв'язків між таблицями наведено у методі OnModelCreating (див. рис. 4.6).

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    modelBuilder.Entity<Recruiter>()
    .HasOne(a => a.Calendar)
    .WithOne(a => a.Owner)
    .HasForeignKey<Calendar>(c => c.OwnerId)
    .OnDelete(DeleteBehavior.Cascade);

    modelBuilder.Entity<Event>()
    .HasOne(a => a.Note)
    .WithOne(a => a.Event)
    .HasForeignKey<Note>(c => c.EventId)
    .OnDelete(DeleteBehavior.Cascade);

    modelBuilder.Entity<Calendar>()
    .HasMany(a => a.Events)
    .WithOne(a => a.Calendar)
    .OnDelete(DeleteBehavior.Cascade);

    modelBuilder.Entity<Event>()
    .HasOne<Calendar>(a => a.Calendar)
    .WithMany(a => a.Events)
    .HasForeignKey(a => a.CalendarId)
    .IsRequired(true)
    .OnDelete(DeleteBehavior.Cascade);

    modelBuilder.Entity<Recruiter>()
    .HasOne(a => a.Dashboard)
    .WithOne(a => a.Owner)
    .HasForeignKey<Dashboard>(c => c.OwnerId)
    .OnDelete(DeleteBehavior.Cascade);

    modelBuilder.Entity<Recruiter>()
    .HasMany(a => a.Mails)
    .WithOne(a => a.FromRecruiter)
    .OnDelete(DeleteBehavior.Cascade);
```

Рисунок 4.6 – Налаштування зв'язків між таблицями (рисунок виконаний самостійно)

Інтерфейс IRepository<TEntity> є частиною шару доступу до даних (Data Access Layer, DAL) в архітектурі додатка. Він забезпечує загальні методи для взаємодії з базою даних, абстрагуючи бізнес-логіку від деталей реалізації збереження даних. Використання цього інтерфейсу дозволяє зменшити дублювання коду та спрощує підтримку і розширення додатка. Інтерфейс IRepository<TEntity> є прикладом реалізації шаблону репозитарію, який забезпечує універсальний спосіб роботи з будь-яким типом даних в контексті Entity Framework (див. рис. 4.7). Він визначає набір методів для виконання операцій CRUD (Create, Read, Update, Delete) та інших типових операцій з базою даних.

```
public interface IRepository<TEntity> where TEntity : BaseEntity
{
    public Task<IEnumerable<TEntity>> GetAll(
        Expression<Func<TEntity, bool>> filter = null,
        Expression<Func<TEntity, object>> include = null);

    public IQueryable<TEntity> GetAll(Expression<Func<TEntity, bool>> expression);

    public Task<TEntity> Get(
        int id,
        IEnumerable<Expression<Func<TEntity, IEnumerable<object>>>> collections = null,
        Expression<Func<TEntity, object>> include = null);

    public Task<TEntity> FirstOrDefault(Expression<Func<TEntity, bool>> filter = null);

    public Task<TEntity> Create(TEntity entity);

    public Task<TEntity> Update(TEntity entity);

    public Task<bool> Delete(int id);

    public Task<int> Count(Expression<Func<TEntity, bool>> condition = null);
}
```

Рисунок 4.7 Інтерфейс репозитарія (рисунок виконаний самостійно)

У проекті інтеграція з MongoDB здійснюється через клас ChatRepository, який використовує параметри конфігурації, визначені в класі MongoOptions (див. рис. 4.8). Це дозволяє взаємодіяти з колекцією чатів у базі даних MongoDB. Такий підхід забезпечує централізоване керування налаштуваннями підключення і полегшує процес роботи з базою даних, надаючи стандартизовані методи для основних операцій CRUD.

```
public ChatRepository(IOptions<MongoOptions> mongoSettings)
{
    _mongoSettings = mongoSettings.Value;
    var client = new MongoClient(_mongoSettings.ConnectionString);
    var db = client.GetDatabase(_mongoSettings.DatabaseName);
    _chats = db.GetCollection<Chat>(_mongoSettings.ChatCollection);
}

public async Task<Chat?> FirstOrDefault(Expression<Func<Chat, bool>> filter)...

public async Task<IEnumerable<Chat>> Get(Expression<Func<Chat, bool>> filter = null)
{
    filter ??= chat => true;

    var sortDefinition = new SortDefinitionBuilder<Chat>().Descending("Messages.SendTime");
    var result = await _chats.FindAsync(filter, new FindOptions<Chat>() { Sort = sortDefinition });

    return await result.ToListAsync();
}

public async Task<Chat?> Get(string id)...

public async Task<Chat?> Create(Chat chat)...

public async Task<Chat?> Update(Chat chat)
{
    var result = await _chats.ReplaceOneAsync(
        comparisonChat => comparisonChat.Id == chat.Id,
        chat,
        new ReplaceOptions { IsUpsert = true }
    );

    return result.IsAcknowledged && result.MatchedCount > 0 ? chat : null;
}

public async Task<bool> Delete(string id)...

public async Task<long> Count(Expression<Func<Chat, bool>> condition = null)
{
    condition ??= chat => true;
    return await _chats.CountDocumentsAsync(condition);
}
```

Рисунок 4.8 Репозитарій чату (рисунок виконаний самостійно)

Інтеграція з MongoDB є важливою частиною архітектури додатка, яка дозволяє зберігати та обробляти неструктуровані дані. У випадку реалізації чату,

кожне повідомлення може бути представлене як окремий документ у колекції MongoDB (див. рис. 4.9).

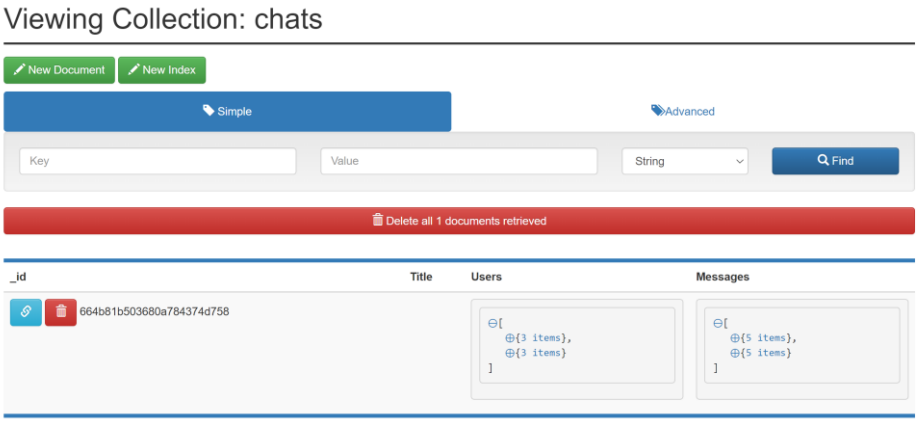


Рисунок 4.9 Колекція чатів (рисунок виконаний самостійно)

База даних MongoDB відноситься до категорії NoSQL баз даних і відрізняється від звичайних реляційних баз, таких як SQL, тим, що вона структурує дані у форматі документів, а не таблиць.

4.4 Розробка API та обробка запитів

Реалізація API проекту включає різні контролери, кожен з яких виконує певну функцію і надає кінцеві точки для взаємодії з різними ресурсами. Система включає такі кінцеві точки:

— ApplicationController: цей контролер відповідає за керування заявками на роботу;

— CalendarController: цей контролер відповідає за керування календарними подіями, такими як організація зустрічей, планування інтерв'ю та інші події, пов'язані з наймом персоналу;

— ChatController: цей контролер відповідає за функціонал чату, забезпечуючи можливість обміну повідомленнями між користувачами системи;

— DashboardController: цей контролер відповідає за керування панелями управління, які надають користувачам зручний інтерфейс для

відстеження статусу заявок, відгуків та інших важливих аспектів найму персоналу;

— EmailTemplateController: цей контролер відповідає за керування шаблонами email-повідомлень, що використовуються для надсилання повідомлень про стан заявок та іншу важливу інформацію;
— EventController: цей контролер відповідає за керування подіями, пов'язаними з процесом найму персоналу, такими як інтерв'ю, зустрічі та інші важливі події;
— EventTemplateController: цей контролер відповідає за керування шаблонами подій, які використовуються для автоматизації процесу створення нових подій;
— IdentityController: цей контролер відповідає за функціонал аутентифікації та авторизації користувачів. Він надає можливість реєстрації, входу та керування обліковими записами користувачів;
— MailController: цей контролер відповідає за керування електронною поштою, забезпечуючи можливість відправлення та отримання поштових повідомлень;
— RecruiterController: цей контролер відповідає за керування рекрутерами, які взаємодіють з системою;
— ReviewController: цей контролер відповідає за керування відгуками, що залишаються користувачами системи про заявки та інші аспекти найму персоналу;
— StatisticsController: цей контролер відповідає за надання статистичних даних про різні аспекти діяльності системи, такі як кількість заявок, кількість подій, розподіл вакансій за періодом часу тощо;
— UserController: цей контролер відповідає за керування обліковими записами користувачів системи;
— CandidateController: цей контролер відповідає за керування кандидатами, що подали заявки на роботу;

— VacancyController: цей контролер відповідає за керування вакансіями, які доступні для заповнення;
— NoteController: цей контролер відповідає за керування записами, пов'язаними зі заявками, подіями та іншими об'єктами систем;
— QuizController: цей контролер відповідає за керування даними тестувань.

У проекті використовуються атрибути доступу, такі як [AllowAnonymous], щоб дозволити не авторизованим користувачам отримати доступ до певних кінцевих точок без аутентифікації, наприклад, для сторінок реєстрації або входу. Атрибути [AuthorizeRoles] використовуються для обмеження доступу до певних кінцевих точок лише користувачам з певними ролями, забезпечуючи таким чином безпеку додатку та контроль над доступом до функціональності.

5 ТЕСТУВАННЯ РОЗРОБЛЕННОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення – це процес оцінки і перевірки функціональності, продуктивності та безпеки додатка. Для серверної частини додатка тестування API є критично важливим. API (Application Programming Interface) – це інтерфейс, що дозволяє різним програмам взаємодіяти між собою. Тестування API включає в себе перевірку того, що різні запити та відповіді працюють належним чином, дані передаються коректно, а також дотримуються вимоги до безпеки.

5.1 Тестування API серверної частини

Тестування серверного API важливе з кількох причин. Воно гарантує коректність роботи API, допомагаючи переконатися, що API повертає правильні дані у відповідь на запити, обробляє всі можливі помилки та працює відповідно до специфікацій. Тестування допомагає виявити вразливості, які можуть бути використані зловмисниками для атак на систему. Регулярне тестування дозволяє оцінити, наскільки ефективно працює API під навантаженням, що важливо для масштабування додатка. Автоматизовані тести спрощують процес внесення змін до коду, оскільки розробники можуть швидко перевірити, чи не порушені існуючі функціональності.

Swagger – це потужний інструмент для документування та тестування API. Кожна кінцева точка має детальний опис, включаючи тип запиту (GET, POST, PUT, DELETE), необхідні параметри, приклади запитів і відповідей. У Swagger UI можна легко тестувати запити, безпосередньо взаємодіючи з API. Для цього потрібно вибрати кінцеву точку, заповнити необхідні параметри та натиснути кнопку "Execute". Swagger відобразить результати запиту, включаючи статусний код, заголовки та тіло відповіді

Документація контролерів є важливим аспектом будь-якого API. Вона надає інформацію про кожен метод контролера, включаючи його призначення, типи запитів, параметри та можливі відповіді. У даному проекті кожен метод

контролера документується за допомогою XML-коментарів, які описують, що робить цей метод, які параметри він приймає і які відповіді можуть бути повернені.

У проекті використовуються атрибути для контролю доступу до методів API, такі як [AllowAnonymous] та [AuthorizeRoles]. Атрибут [AllowAnonymous] дозволяє доступ до методу без необхідності авторизації. Він використовується для публічно доступних кінцевих точок, таких як реєстрація чи отримання загальної інформації. Атрибут [AuthorizeRoles(Role.Candidate, Role.Recruiter)] обмежує доступ до методу тільки для користувачів з певними ролями. Наприклад, метод з таким атрибутом буде доступний лише кандидатам та рекрутерам.. Для тестування таких методів у Swagger необхідно пройти авторизацію. Це здійснюється шляхом надання токену JWT, який отримується після успішного входу в систему. У Swagger є можливість додати токен у заголовки запитів, що дозволяє тестувати методи, які вимагають авторизації.

На прикладі контролеру Identity, протестуємо реєстрацію в додатку. Для цього нам не потрібна авторизація, лише валідні дані. Якщо дані пройшли валідацію, то буде створений користувач під однією з ролей та повернеться 200 Success, а якщо ні – API поверне 400 Bad Request. Протестуємо обидва варіанти.

Для початку введемо всі необхідні дані, вважаючи, що вони унікальні в системи і виконаємо запит, отримуємо позитивний результат та токен, використовучи який далі можемо авторизуватися та отримати доступ до всіх інших запитів (див. рис. 5.1). Поки користувач немає цього токену, він не може звернутися до інших контролерів, бо отримає помилку та статус 401, який дозволяє зрозуміти, що користувач не авторизований (див. рис. 5.2).

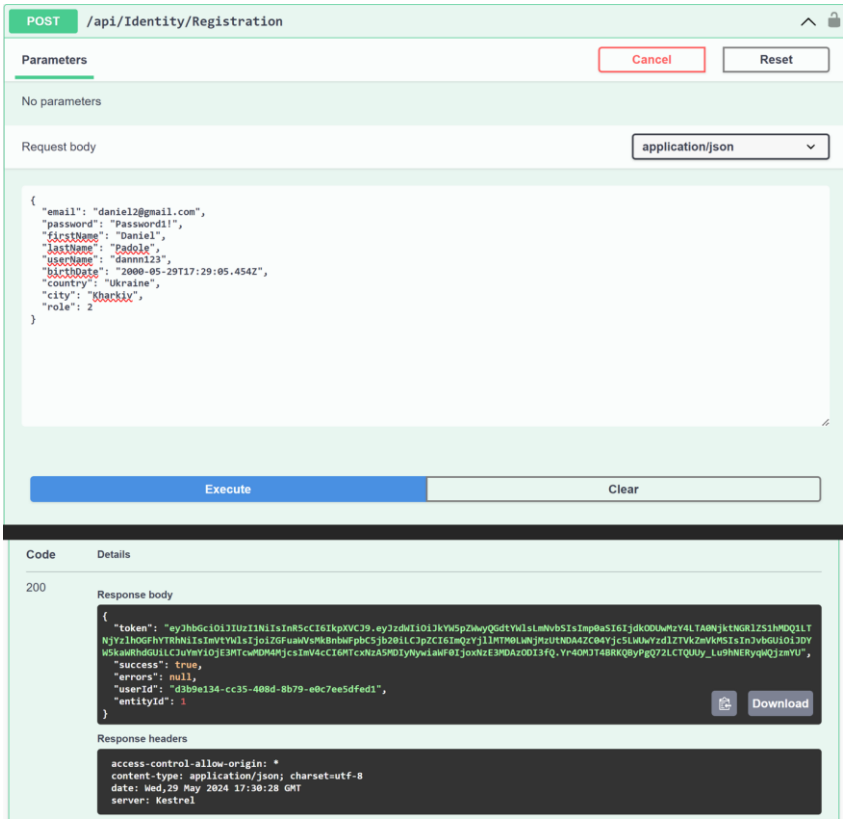


Рисунок 5.1 Позитивний результат запиту (рисунок виконаний самостійно)

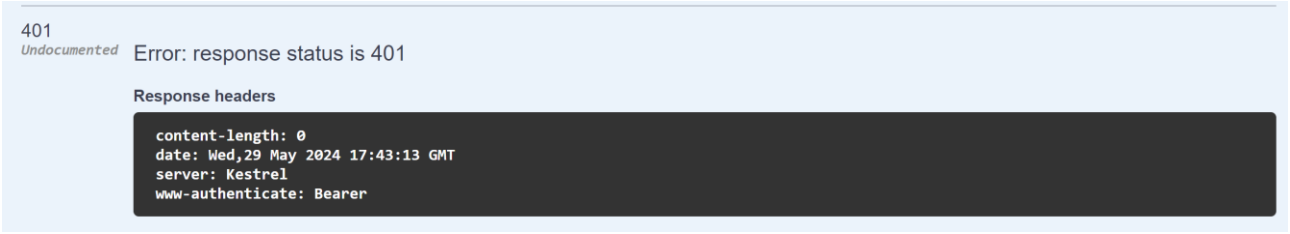


Рисунок 5.2 Помилка відсутності доступу (рисунок виконаний самостійно)

Тестування методу Registration включає перевірку різних сценаріїв, у тому числі і випадків введення некоректних даних. Один із важливих аспектів тестування - перевірка валідаційних помилок, які виникають при введенні невірних даних. Розглянемо сценарій, коли користувач вводить некоректний пароль та електронну пошту. У цьому прикладі, поле email не відповідає стандартному формату електронної пошти, а пароль занадто короткий і не

відповідає мінімальним вимогам до безпеки. Запит відправляється на сервер для обробки. Сервер перевіряє дані та виявляє, що вони не відповідають вимогам валідації. Сервер повертає відповідь із валідаційною помилкою, вказуючи, що електронна пошта та пароль мають некоректний формат (див. рис. 5.3).

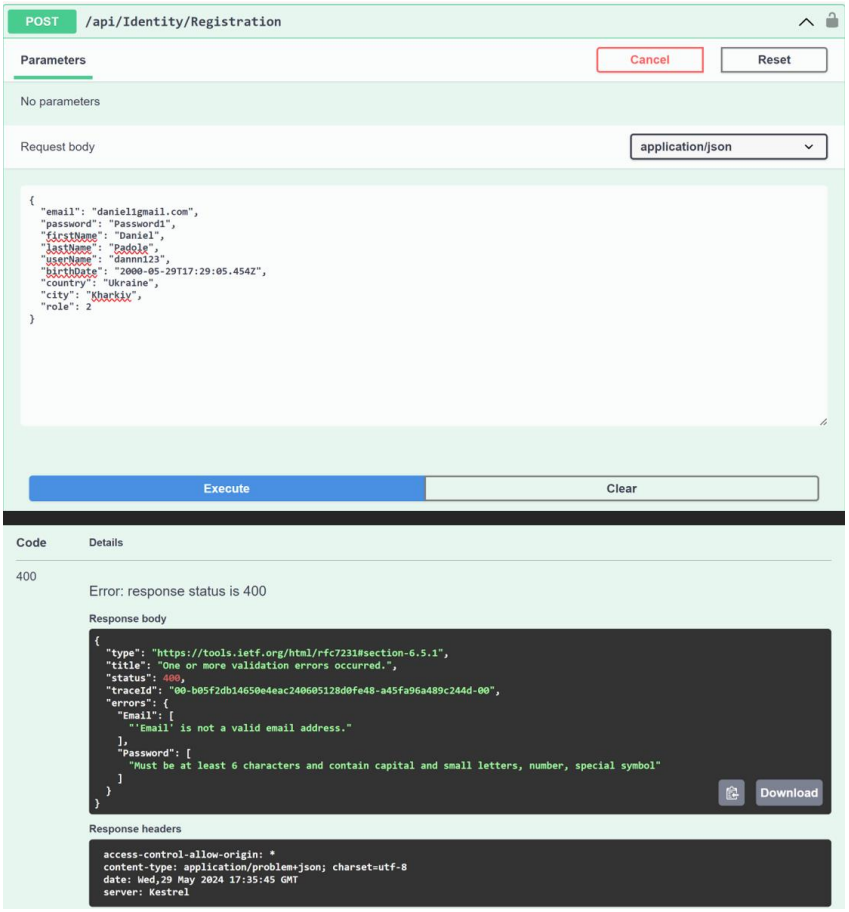


Рисунок 5.3 Помилка валідації (рисунок виконаний самостійно)

Це повідомлення допомагає користувачу зрозуміти, які дані потрібно виправити для успішної реєстрації. Таким чином, валідаційні помилки забезпечують зворотний зв'язок і підвищують зручність використання системи.

ВИСНОВКИ

Реалізована система рекрутингу відзначається високою інноваційністю і вигідністю порівняно з іншими підходами завдяки комплексному підходу до вирішення проблем в сфері підбору персоналу. Також вона поєднує в собі передові технології баз даних, використання розподіленого зберігання даних та швидкодійних серверних архітектур, що дозволяє нам забезпечувати високу ефективність, масштабованість і безпеку.

Однією з ключових переваг системи є її інноваційний підхід до обробки та зберігання даних. Система використовує MongoDB як базу даних, що дозволяє забезпечити гнучкість у зберіганні даних різної структури. MongoDB дає можливість також зберігати документи з різними полями в одній колекції, що особливо важливо для рекрутингової системи, де дані можуть мати різну структуру, наприклад, дані про вакансії, кандидатів, резюме тощо.

Крім того, система використовує передові технології серверної архітектури для забезпечення високої швидкодії та масштабованості. Серверна частину системи реалізована за допомогою фреймворку ASP.NET Core, який забезпечує високу продуктивність, а також можливість розгортання на різних платформах. Система також використовує передові методики управління даними та безпеки. В проекті використовуються механізми аутентифікації та авторизації користувачів на основі JWT токенів, що забезпечує безпеку та конфіденційність даних.

В цілому, дана інноваційна система рекрутингу вирізняється комплексним підходом до вирішення проблем в сфері підбору персоналу, використанням передових технологій управління даними та безпеки, а також забезпеченням високої ефективності, масштабованості та швидкодії. Ця система допоможе підприємствам оптимізувати процеси підбору персоналу та забезпечить їм конкурентну перевагу на ринку праці.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. RECRUITEE [Електронний ресурс] – URL: https://recruitee.com (дата звернення: 10.05.2024).

2. MANATAL [Електронний ресурс] – URL: https://www.manatal.com (дата звернення: 10.05.2024).

3. Google AdSense [Електронний ресурс] – URL: https://adsense.google.com (дата звернення: 11.05.2024).

4. CPA-мережі [Електронний ресурс] – URL: https://finance.ua (дата звернення: 11.05.2024).

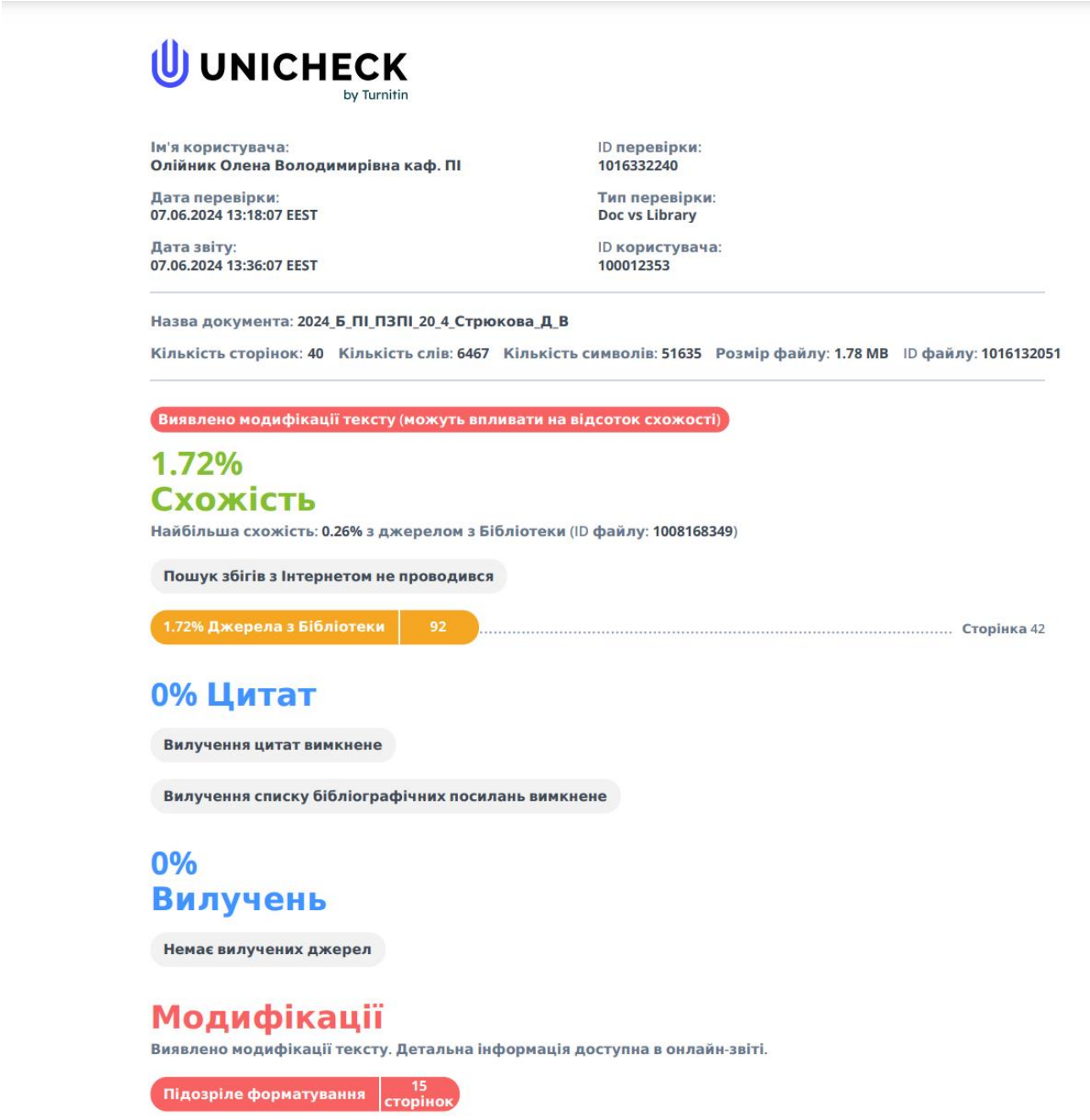
5. Multitier [Електронний ресурс] – URL: https://medium.com/ayokoding/system-design-multi-tier-architecture (дата звернення: 13.05.2024).

6. Docker [Електронний ресурс] – URL: https://www.freecodecamp.org (дата звернення: 13.05.2024).

7. MongoDB [Електронний ресурс] – URL: https://dev.to/manthanank (дата звернення: 13.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016332240

Дата перевірки:
07.06.2024 13:18:07 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
07.06.2024 13:36:07 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ_20_4_Стрюкова_Д_В

Кількість сторінок: 40 Кількість слів: 6467 Кількість символів: 51635 Розмір файлу: 1.78 MB ID файлу: 1016132051

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

1.72%
Схожість

Найбільша схожість: 0.26% з джерелом з Бібліотеки (ID файлу: 1008168349)

Пошук збігів з Інтернетом не проводився

1.72% Джерела з Бібліотеки 92 ..... Сторінка 42

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

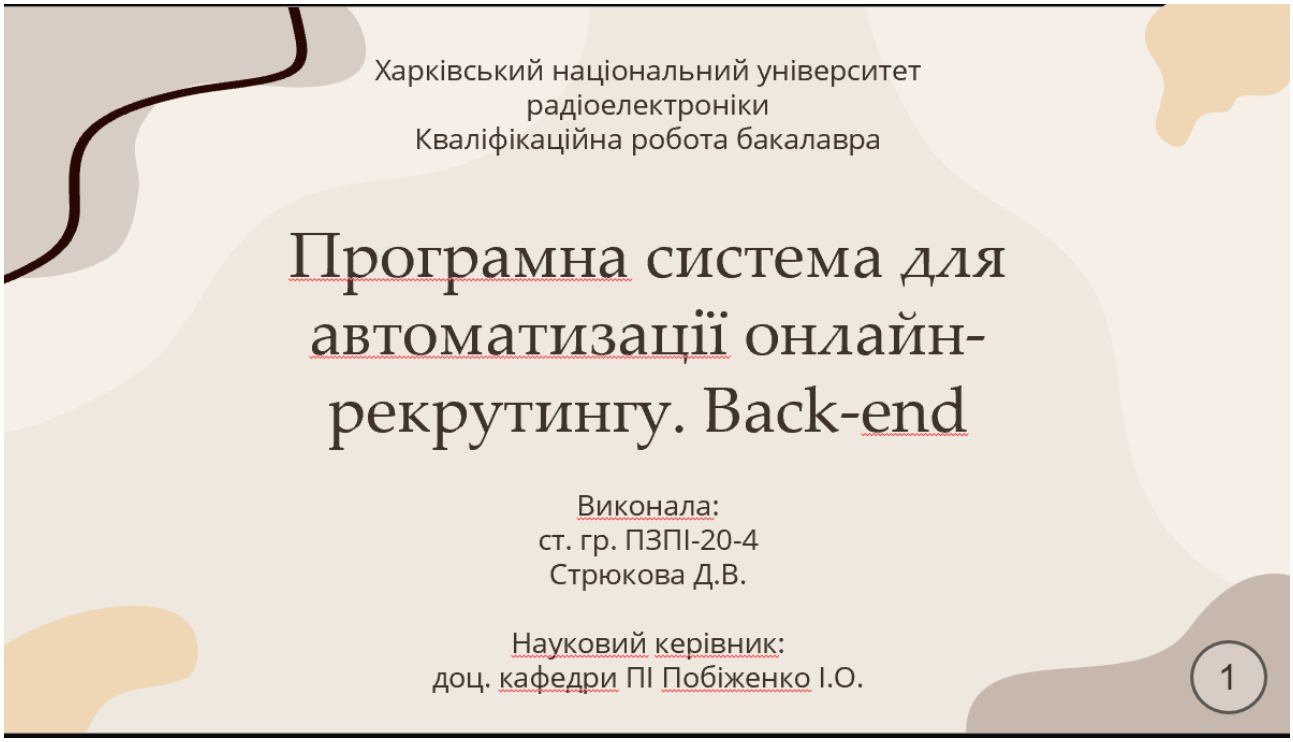
Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 15 сторінок

ДОДАТОК Б

Слайди презентації



АКТУАЛЬНІСТЬ

01 — Автоматизація дозволяє значно зменшити час, необхідний для обробки заявок та відбору кандидатів

02 — Система забезпечує точність та об'єктивність в оцінюванні кандидатів за ключовими параметрами, такими як досвід та навички.

03 — Автоматизація на серверному рівні полегшує управління великим обсягом даних, забезпечуючи їхню систематизацію та доступність.

3

ПОСТАНОВКА ЗАДАЧІ

01 — Створення багатошарового веб-додатку, який дозволяє розділити функціональність на різні рівні для незалежного виконання.

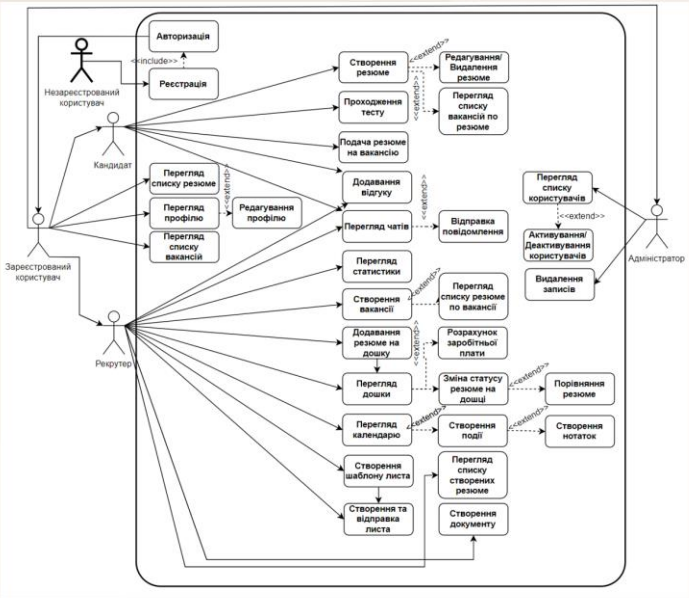
02 — Забезпечення ефективної роботи з різними типами даних та вимогами проекту завдяки використанню двох різних баз даних.

03 — Можливість взаємодії з сервером за допомогою контролерів та захищеної комунікації за протоколом HTTPS, забезпечуючи безпеку передачі даних.

04 — Забезпечення безпеки даних та конфіденційності інформації, а також швидкої та ефективної роботи

4

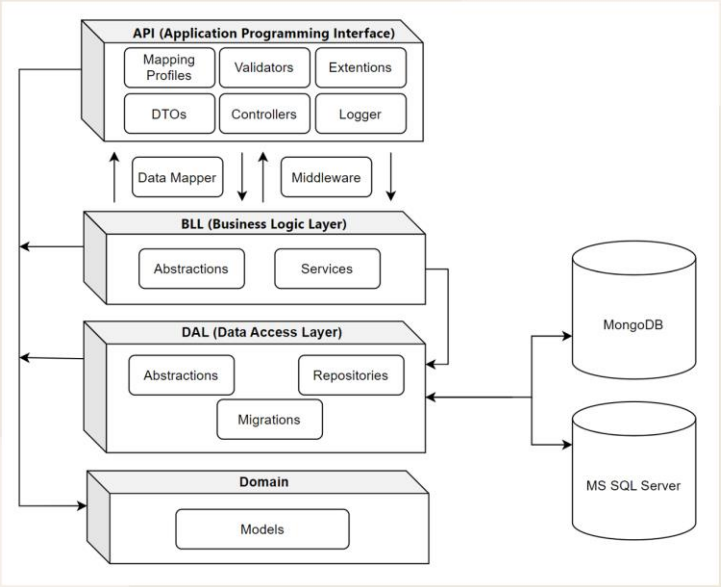
АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Діаграма Use Case

5

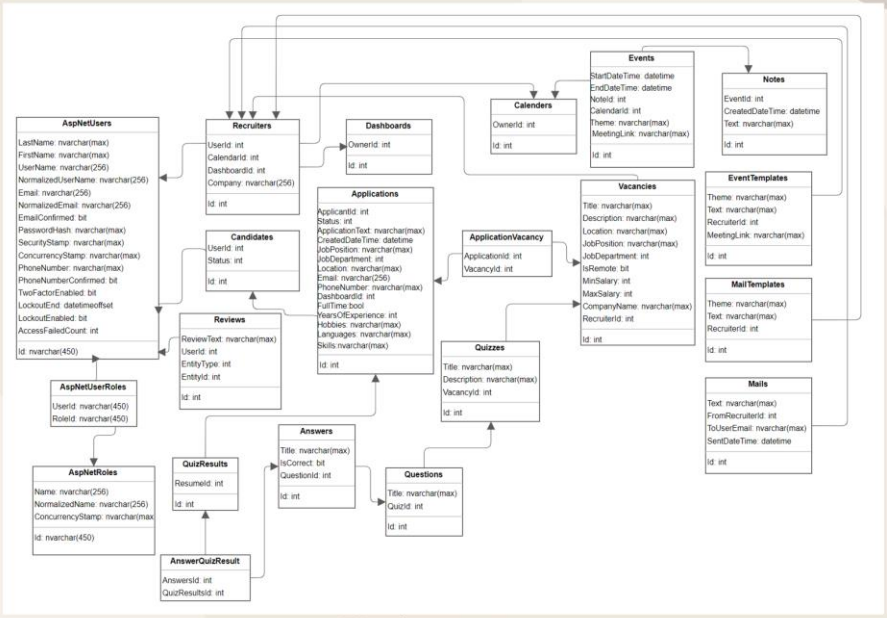
АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Архітектура серверної частини додатку

6

АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



ER діаграма

СТЕК ТЕХНОЛОГІЙ

mongoDB

Microsoft® SQL Server®

Entity Framework

Swagger

БАЗИ ДАНИХ

MS SQL

MS SQL є потужною та надійною реляційною системою керування базами даних, яка широко використовується для побудови реляційних баз даних

MONGO DB

MongoDB є популярною NoSQL базою даних, яка забезпечує гнучку та масштабовану зберігання даних у вигляді документів. Вона особливо корисна для зберігання даних зі змінною структурою або великим обсягом даних

9

ENTITY FRAMEWORK

У проекті для роботи з SQL базою даних використовується Entity Framework, Identity Framework та підхід Code First разом із міграціями. Цей вибір зроблений з урахуванням кількох факторів, які визначають ефективність, зручність та безпеку роботи з даними в програмній системі для автоматизації онлайн-рекрутингу.

Entity Framework

01 використовується для реалізації аутентифікації та авторизації користувачів у системі

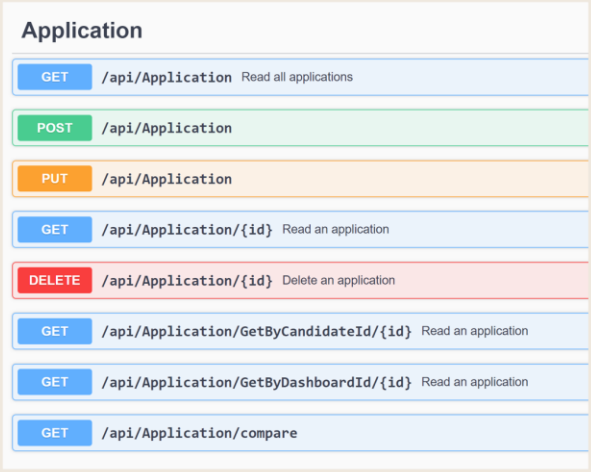
02 надає різноманітні можливості для взаємодії з даними, включаючи створення моделей даних, виконання запитів та керування зв'язками між об'єктами

03 дозволяє спростити розробку, забезпечуючи високий рівень абстракції над базою даних

10

API ендпоінти

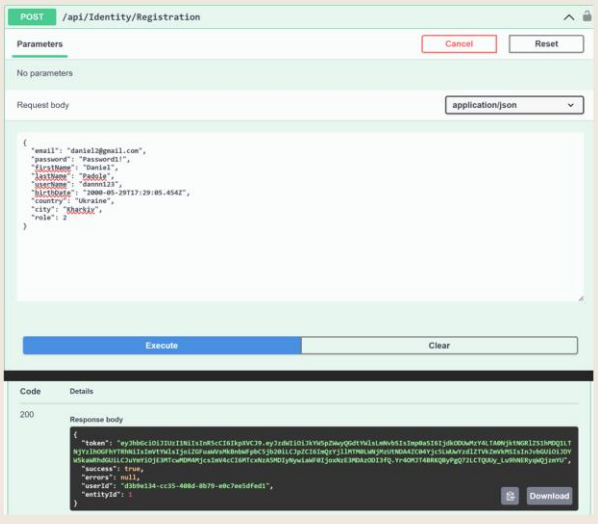
- ApplicationController
- CalendarController
- ChatController
- DashboardController
- EmailTemplateController
- EventController
- EventTemplateController
- IdentityController
- MailController
- RecruiterController
- ReviewController
- StatisticsController
- UserController
- CandidateController
- VacancyController
- NoteController
- QuizController



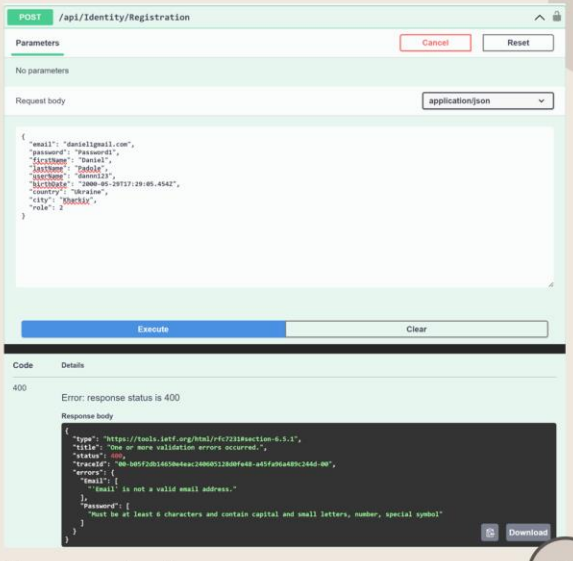
Приклад контролеру

11

ТЕСТУВАННЯ API



Позитивний результат



Помилка валідації

12

АПРОБАЦІЯ

Сертифікат учасника конференції «Print, Multimedia and Web» за доповідь «Аналіз трендів у сфері рекрутингу та їх відображення в функціональних можливостях додатків»

13

ВИСНОВКИ

01 Розроблено багатошарову архітектуру на платформі .NET з використанням C#: Це забезпечило чіткий поділ функціональності, підвищивши продуктивність та спрощенням обслуговування системи.

02 Інтегровано дві бази даних, MS SQL та MongoDB: Це дозволило оптимізувати роботу з різними типами даних, забезпечуючи ефективність і гнучкість зберігання та обробки інформації.

03 Забезпечено безпечну комунікацію через HTTPS та дотримання принципів SOLID: Це підвищило рівень захисту даних користувачів і забезпечило створення добре структурованого, розширюваного та підтримуваного коду, що сприяє довготривалій підтримці та масштабованості системи.

14

ДОДАТОК В

Специфікація вимог до програмного продукту

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Факультет комп’ютерних наук
Кафедра програмної інженерії

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмна система для автоматизації онлайн-рекрутингу

Студент гр. ПЗПІ-20-5 Клименюк Едуард Сергійович
Студент гр. ПЗПІ-20-4 Харченко Поліна Дмитрівна
Студент гр. ПЗПІ-20-4 Стрюкова Дар’я В’ячеславівна

Харків
2024 р.

ЗМІСТ

1 ВСТУП 3
1.1 Огляд продукту 3
1.2 Мета 3
1.3 Межі 4
1.4 Означення та абревіатури 5
2 ЗАГАЛЬНИЙ ОПИС 7
2.1 Перспективи продукту 7
2.2 Функції продукту 7
2.3 Характеристики користувачів 8
2.4 Загальні обмеження 9
2.5 Припущення й залежності 9
3 КОНКРЕТНІ ВИМОГИ 11
3.1 Вимоги до зовнішніх інтерфейсів 11
3.1.1 Інтерфейс користувача 11
3.1.2 Апаратний інтерфейс 11
3.1.3 Програмний інтерфейс 12
3.1.4 Комунікаційний протокол 13
3.1.5 Обмеження пам’яті 13
3.2 Атрибути програмного продукту 14
3.2.1 Надійність 14
3.2.2 Доступність 14
3.2.3 Безпека 15
3.2.4 Супроводжуваність 15
3.2.5 Переносність 16
3.2.6 Продуктивність 16
3.3 Вимоги до бази даних 17

1 ВСТУП

1.1 Огляд продукту

Програмна система для автоматизації онлайн-рекрутингу є потужним інструментом для компаній, які займаються наймом персоналу. Цей продукт надає комплекс функцій, які сприяють ефективному керуванню процесом рекрутингу та спрощують взаємодію з кандидатами.

Система забезпечує безпеку та контроль доступу за допомогою системи аутентифікації та авторизації. Це гарантує, що доступ до системи мають лише авторизовані користувачі, такі як рекрутери та адміністратори, забезпечуючи конфіденційність даних.

Адміністратори можуть легко керувати користувачами, створювати нові облікові записи, надавати різні рівні доступу та редагувати існуючі профілі. Це дозволяє ефективно організовувати робочий процес та розподіляти обов'язки серед персоналу.

Система дозволяє створювати та редагувати вакансії, розміщувати оголошення про вакансії та отримувати аплікації від кандидатів. Користувачі можуть переглядати деталі вакансій, такі як опис посади, вимоги до кандидатів та умови праці, а також подавати свої резюме. Це спрощує процес рекрутингу та дозволяє рекрутерам ефективно відбирати потенційних кандидатів.

Управління кандидатами є ще одним важливим аспектом продукту. Рекрутери можуть вести базу даних кандидатів, відстежувати їх статуси, проводити співбесіди та приймати рішення щодо найняття.

Ця програмна система сприяє покращенню ефективності процесу рекрутингу, зменшенню часу на пошук та відбір кандидатів, а також підвищенню якості найму.

1.2 Мета

Система для автоматизації онлайн-рекрутингу спрямована на створення зручного та централізованого середовища для організацій, які здійснюють найм

персоналу. Головною метою цієї системи є надання керівникам зручного і ефективного інструменту для планування та координації найму персоналу, забезпечуючи їх можливістю організовано проводити відбір кандидатів та вести облік рекрутингових процесів.

Крім того, система спрощує комунікацію між рекрутерами та кандидатами, надаючи зручний механізм обміну інформацією. Вона автоматично повідомляє про нові вакансії та статуси рекрутингових процесів, що в свою чергу дозволяє кандидату швидко реагувати на зміни та подаватися на нові вакансії.

Загалом, ця програмна система спрямована на поліпшення ефективності та організації процесу найму персоналу, надаючи зручні інструменти для планування, координації, відстеження та звітності щодо вакансій та кандидатів.

1.3 Межі

Програмна система спрямована на забезпечення ефективності та швидкості процесів рекрутингу, зокрема шляхом автоматизації рутинних завдань та зменшення людського втручання.

Цілі програмного продукту включають зменшення часу, потрібного для створення вакансій і подачі заявок, підвищити ефективність процесу найму.

Вона має надавати зручний API для взаємодії з клієнтською частиною додатку та має бути здатна обробляти запити від багатьох користувачів одночасно.

Однак, у системі є обмеження, такі як ресурси сервера (пам'ять та потужність процесора), які впливають на її продуктивність та масштабованість. Це може уповільнити обробку запитів та знизити доступність для користувачів. Також існують обмеження на кількість користувачів та ролей, що можуть бути створені. Додатково, система повинна забезпечувати безпеку даних, використовуючи шифрування та інші методи захисту, щоб убезпечити інформацію від несанкціонованого доступу.

1.4 Означення та абревіатури

Frontend – це частина програмної системи, що відповідає за відображення інтерфейсу користувача та взаємодію з ним через браузер або інші засоби. Вона включає в себе розробку та реалізацію веб-сторінок, компонентів і елементів керування, які користувач може бачити та взаємодіяти з ними.

Backend – це складова частина програмного забезпечення, яка відповідає за обробку даних та логічні операції на серверному рівні. Вона забезпечує взаємодію з базою даних, виконує процеси автентифікації, авторизації та управління бізнес-логікою системи.

API – це скорочення від Application Programming Interface. Це набір правил і протоколів, що визначає, як програми або компоненти системи можуть взаємодіяти між собою. API визначає доступні функції та методи комунікації між різними складовими програмної системи.

ORM – це абревіатура від Object-Relational Mapping, що означає технологію, яка дозволяє взаємодіяти з базою даних у вигляді об'єктів програми. Вона перетворює дані з реляційної бази даних у об'єкти, що можуть бути використані в програмному коді, та надає зручний спосіб управління даними.

HTTP – це протокол передачі гіпертекстових даних через мережу, що використовується для комунікації між клієнтами та серверами в Інтернеті.

REST – підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів.

CRUD – це абревіатура, що складається зі слів Create, Read, Update, Delete (створити, прочитати, оновити, видалити). Це основні операції, які можна виконати з даними в базі даних, включаючи створення нових записів, отримання даних, їх оновлення та видалення.

Entity Framework – це технологія ORM (Object-Relational Mapping), яка дозволяє розробникам працювати з базами даних у формі об'єктів програми. Вона забезпечує зручний спосіб доступу до даних з бази даних, перетворюючи

їх на об'єкти, з якими можна взаємодіяти в програмному коді.

Identity – це система аутентифікації та авторизації користувачів в програмній системі. Вона дозволяє контролювати доступ до різних частин системи, перевіряючи ідентифікаційні дані користувача та надаючи йому відповідні права доступу.

HR – це скорочення від Human Resources (кадрові ресурси). Це відділ або функціональна область в організації, що відповідає за управління персоналом. Відділ HR займається наймом та звільненням працівників, адмініструванням персональних даних, розвитком персоналу, а також забезпечує вирішення конфліктів та питань стосовно заробітної плати та соціальних пакетів.

JSON (JavaScript Object Notation) – це легкий, текстовий формат обміну даними, що базується на синтаксисі JavaScript. Він використовується для передачі структурованих даних між програмами, особливо у веб-розробці. Формат JSON складається з пар ключ-значення, де ключі є рядками, а значення можуть бути будь-якого типу даних: рядки, числа, масиви, об'єкти тощо.

CRM (Customer Relationship Management) – програмне забезпечення для організацій, призначене для автоматизації взаємодії із замовниками (клієнтами).

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Останнім часом спостерігається значний інтерес до автоматизації процесу найму та підвищення ефективності управління персоналом. Наш продукт виявляє значний потенціал застосування у різних галузях, включаючи компанії будь-якого масштабу, агентства з кадрового рекрутингу та інші сфери.

Прогнозовані напрямки розвитку нашого продукту включають розширення функціоналу та впровадження нових модулів для відповіді на конкретні потреби користувачів. Наприклад, можливості планування та розподілу ресурсів, система оцінки кандидатів, інтеграція з іншими

платформами або соціальними мережами, розширена аналітика та звітність.

Крім того, перспективи включають вихід на міжнародний ранок та масштабування продукту для задоволення потреб різних компаній та організацій у різних країнах. Адаптація продукту до міжнародних стандартів та норм у сфері рекрутингу відкриє нові можливості для його впровадження на глобальному рівні.

Загалом, наш продукт має значний потенціал для розвитку та відповіді на зростаючі потреби в управлінні процесом найму персоналу.

2.2 Функції продукту

Основні функції цього продукту включають:

FE-1: Дошка відображення процесу найму (аналог Jira)

FE-2: Календар з відображенням онлайн зустрічей та інших подій

FE-3: Google meets, тобто створення онлайн зустрічі з кандидатом

FE-4: Імейли, відправка з системи на пошту, шаблони

FE-5: Розрахунок заробітної плати

FE-6: Резюме кандидата з можливістю прикріпити файли

FE-7: Тести для вакансії

FE-8: Чат з рекрутером

FE-9: Статистика за вакансіями, компаніями, кандидатами, загальна, стан заявки на вакансію

FE-10: Звіт по вакансії

FE-11: Порівняння кандидатів та резюме

FE-12: Генерація документів, наприклад контракти та листи пропозиції на основі даних кандидата

FE-14: Відгуки

FE-15: Підписка на вакансію та розсилка імейлів

FE-16: Релевантність вакансії вимогам

2.3 Характеристики користувачів

Характеристики користувачів програмної системи для керування процесом набору персоналу з back-end реалізацією можуть бути різноманітними, оскільки система взаємодіє з різними типами користувачів. Основні характеристики користувачів включають:

– кандидати на роботу є ключовими користувачами системи. Вони можуть мати різний досвід роботи, освіту та навички, що впливають на їхню придатність для конкретної вакансії;
– HR менеджери відповідають за керування процесом набору персоналу. Вони мають розширені права доступу та можуть створювати, редагувати та видаляти вакансії, керувати кандидатами, аналізувати звітність та виконувати інші адміністративні функції;
– адміністратори системи відповідають за налаштування та підтримку програмного забезпечення. Вони забезпечують безперебійну роботу системи, вирішують технічні проблеми, які можуть виникнути.

Загалом, різноманітність користувачів враховується у розробці системи, щоб кожна категорія могла ефективно використовувати її функціонал для досягнення своїх цілей у процесі найму персоналу.

2.4 Загальні обмеження

У програмній системі для управління рекрутингом з back-end реалізацією існують обмеження, які можуть вплинути на її ефективність та можливості використання.

Продуктивність та масштабованість системи обумовлені фізичними параметрами серверів, такими як кількість пам'яті, потужність обчислювального процесора, а також пропускна здатність мережі. Ці обмеження можуть суттєво вплинути на швидкість та завантаження системи під час обробки великої кількості запитів одночасно.

Забезпечення безпеки є важливою складовою для захисту

конфіденційності даних користувачів та запобігання несанкціонованому доступу. Обмеження в цьому плані можуть включати криптографічні вимоги, захист від потенційних атак, встановлення правил доступу до інформації та контроль над правами користувачів.

Система повинна мати можливість масштабуватися для відповіді на зростаючий обсяг користувачів, проектів та завдань. Обмеження масштабованості можуть впливати на продуктивність та швидкодію системи під час інтенсивного навантаження.

2.5 Припущення й залежності

AS-1: Доступ до Інтернету. Передбачається, що всі користувачі мають доступ до Інтернету для використання системи онлайн-рекрутингу.

AS-2: Відповідність інформації в резюме. Передбачається, що інформація, надана кандидатами у їхніх резюме, є точною і відповідає їхньому досвіду та навичкам.

AS-3**:** Стабільність технічних систем. Передбачається, що технічні системи, необхідні для роботи програмної платформи, будуть працювати стабільно без значних перебоїв.

DP-1: Доступ до бази даних кандидатів. Система залежить від наявності доступу до бази даних з профілями кандидатів для ефективного пошуку та відбору претендентів на вакансії.

DP-2: Інтеграція з існуючими системами рекрутингу. Може бути необхідною інтеграція з існуючими системами рекрутингу або обліку кандидатів, які вже використовуються компанією.

DP-3: Підтримка технічного персоналу. Для ефективної роботи системи може знадобитися технічна підтримка для моніторингу.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Система повинна мати інтуїтивно зрозумілий та ергономічний інтерфейс, що дозволяє користувачам легко взаємодіяти з програмним забезпеченням. Інтерфейс користувача має бути зручним у використанні та забезпечувати швидкий доступ до всіх основних функцій системи.

Крім того, інтерфейс повинен бути адаптивним для різних типів пристроїв, включаючи комп'ютери, планшети та мобільні девайси.

Інтерфейс користувача представлений у вигляді веб-додатку, залежно від потреб користувачів та доступних платформ. Додаток має послідовну організацію, зрозумілу навігаційну систему та інтуїтивно зрозумілі елементи управління, що спрощують користувачам взаємодію та виконання різноманітних завдань.

3.1.2 Апаратний інтерфейс

Апаратне забезпечення програмної системи для управління рекрутингом з back-end реалізацією складається з необхідних компонентів та засобів, які гарантують безперебійну роботу системи. Оскільки програма зазвичай працює в онлайн-середовищі та доступна через веб-браузер, апаратне забезпечення включає:

– сервери для зберігання та обробки даних. Це можуть бути фізичні сервери, віртуальні сервери або хмарні платформи, які забезпечують необхідні обчислювальні та зберігальні ресурси для ефективної роботи системи;

– мережеві пристрої, такі як маршрутизатори, комутатори та мережеві кабелі, для забезпечення безперервного зв'язку між користувачами та серверами системи через мережу.

3.1.3 Програмний інтерфейс

Програмна система для автоматизації онлайн-рекрутингу впроваджує програмний інтерфейс (API) як ключовий механізм для спільної роботи з іншими додатками та системами. Цей API визначає набір правил та протоколів

для взаємодії між різними компонентами програмного забезпечення, що дозволяє зовнішнім системам або розробникам використовувати функціональні можливості та отримувати доступ до даних програмної системи.

У контексті онлайн-рекрутингу, API може включати в себе різноманітні функції, такі як створення та оновлення вакансій, перегляд кандидатів, управління процесом найму та багато іншого. Реалізація такого API у формі RESTful API є популярним підходом, оскільки він базується на використанні HTTP-протоколу, що дозволяє здійснювати просту та стандартизовану взаємодію між клієнтами та сервером.

Однією з переваг RESTful API є його простота та легкість використання. Використання стандартних HTTP-методів, таких як GET, POST, PUT та DELETE, спрощує процес комунікації з системою. Крім того, обмін даними у форматі JSON дозволяє ефективно передавати структуровану інформацію між клієнтом та сервером.

Застосування API у програмній системі для онлайн-рекрутингу відкриває безліч можливостей для інтеграції з іншими системами та сервісами.

Використання API у програмній системі для рекрутингу також дозволяє забезпечити гнучкість та розширюваність системи. Розробники можуть легко додавати нові функції та можливості, не змінюючи основну архітектуру програми, що робить систему більш адаптивною до змін потреб бізнесу та ринкових умов.

У цілому, використання програмного інтерфейсу в програмній системі для автоматизації онлайн-рекрутингу є ключовим фактором для забезпечення її ефективності, гнучкості та інтегрованості з іншими системами.

3.1.4 Комунікаційний протокол

Система повинна підтримувати різні інтерфейси комунікації для забезпечення обміну даними з іншими системами та сервісами. Це може включати в себе стандартні протоколи передачі даних, такі як HTTP/ HTTPS

для взаємодії з веб-серверами, REST для взаємодії з іншими веб-службами, а також можливості інтеграції з поштовими клієнтами. Крім того, система повинна забезпечувати захист даних та аутентифікацію при здійсненні комунікації з іншими системами, щоб забезпечити конфіденційність та цілісність інформації.

3.1.5 Обмеження пам’яті

Обмеження пам'яті є критичним аспектом у розробці програмного забезпечення, особливо в сучасних системах, де ефективне використання ресурсів має вирішальне значення для забезпечення продуктивності та надійності програм.

Під час розробки програмного забезпечення важливо усвідомлювати обмеження пам'яті, які встановлені для різних типів пристроїв та платформ. Наприклад, вбудовані системи можуть мати обмежену кількість доступної пам'яті, тоді як сервери або персональні комп'ютери можуть мати більші ресурси.

Одним з методів управління обмеженням пам'яті є ефективне використання алгоритмів та структур даних. Наприклад, використання компактних представлень даних, які займають менше пам'яті, може допомогти знизити використання ресурсів. Крім того, уникання зайвого копіювання даних та оптимізація роботи з пам'яттю також може покращити продуктивність програми.

Для ефективного управління пам'яттю важливо також враховувати процеси видалення та звільнення пам'яті. Наприклад, у системах, де використання пам'яті є критичним, важливо уникати витоку пам'яті шляхом правильного вивільнення непотрібних ресурсів та використанням механізмів автоматичного управління пам'яттю.

3.2 Атрибути програмного продукту

3.2.1 Надійність

Надійність програмної системи для управління онлайн-рекрутингом є ключовою складовою успішного функціонування. Система має забезпечувати безперебійну та стабільну роботу, що є критично важливим для ефективного управління процесом рекрутингу та задоволення потреб користувачів.

Надійність означає, що система працює так, як очікується, та обробляє запити користувачів, запобігаючи можливим збоям чи відмовам. Для досягнення цієї мети система має бути забезпечена вбудованим механізмом обробки помилок та винятків, що можуть виникати під час роботи. Це допомагає виявляти та вирішувати проблеми, що впливають на роботу системи, забезпечуючи її надійну та стабільну функціональність.

Окрім цього, важливо мати механізми резервного копіювання, які забезпечують збереження та захист даних системи. Це може включати регулярне резервне копіювання даних, створення резервних копій баз даних та інших важливих ресурсів. Такий підхід дозволяє підтримувати надійність та безпеку даних у будь-який час, що є важливим аспектом для успішної роботи програмної системи з управління рекрутингом.

3.2.2 Доступність

Для забезпечення доступності системи можна використовувати різні технології та стратегії. Одним з ключових методів є використання дублювання та резервування. Це означає створення дублюючих екземплярів системи або компонентів, які автоматично вступають в дію у випадку відмови основного обладнання або програмного забезпечення.

Крім того, для забезпечення доступності можна використовувати системи моніторингу та управління. Такі системи постійно контролюють стан обладнання та програмного забезпечення і сповіщають адміністраторів у випадку виявлення проблем. Додатково, вони можуть автоматично виконувати заходи для відновлення роботи системи або виконувати переведення

навантаження на резервні сервери.

Одним з важливих аспектів є розробка системи з урахуванням масштабованості горизонтальної або вертикальної. Горизонтальна масштабованість передбачає розширення системи за рахунок додавання нових серверів або вузлів у мережу, щоб розділити навантаження та забезпечити стабільну роботу при зростанні обсягу даних або користувачів. Вертикальна масштабованість включає у підвищення продуктивності і потужності окремих компонентів системи, наприклад, шляхом оновлення обладнання або програмного забезпечення.

3.2.3 Безпека

Система має забезпечувати високий рівень безпеки для захисту конфіденційності, цілісності та доступності даних. Це охоплює застосування сучасних методів шифрування для конфіденційної інформації, механізми автентифікації для запобігання несанкціонованому доступу та механізми контролю доступу для обмеження прав доступу.

Система повинна використовувати параметризовані запити і виключити можливість SQL-ін'єкцій та інших атак, для забезпечення безпеки запитів до бази даних через API. Також важливо реалізувати аутентифікацію і авторизацію на рівні API, щоб контролювати доступ до даних та забезпечити їх конфіденційність.

3.2.4 Супроводжуваність

Забезпечення документацією відповідає за збереження знань та інформації про систему, що дозволяє новим членам команди ефективно розібратися в її структурі та функціональності. Моніторинг та логування відіграють ключову роль у виявленні проблем та відслідковуванні причин виникнення помилок, що дозволяє оперативно реагувати на них та уникати критичних збоїв у роботі системи.

Регулярні оновлення та тестування забезпечують актуальність та надійність програмного забезпечення, а також виявлення та виправлення помилок ще до їх впливу на користувачів. Підтримка користувачів є важливою для забезпечення задоволення та впевненості користувачів у якості та ефективності системи.

Автоматизація процесів супроводження дозволяє зменшити час та зусилля, витрачені на ручні операції, а також забезпечує більшу стабільність та ефективність управління системою. Включення цих механізмів у пункт про супроводжуваність дозволяє забезпечити готовність та надійність системи для подальшого функціонування після її впровадження.

3.2.5 Переносність

Система повинна бути легко переносною між різними середовищами та платформами. Це включає здатність запускати систему на різних операційних системах, таких як Windows, macOS та Linux, а також на різних типах апаратного забезпечення. Переносність також означає, що система повинна бути здатною працювати в різних мережевих середовищах із різними конфігураціями мережі та доступу до Інтернету.

Крім того, важливо, щоб система могла легко інтегруватися з іншими додатками та сервісами, що використовуються у сфері рекрутингу, такими як CRM-системи, платформи соціальних мереж та рекрутингові портали, незалежно від їхніх технологій чи платформ.

3.2.6 Продуктивність

У програмній системі для управління онлайн-рекрутингом, важливим аспектом є продуктивність, яка включає в себе оптимізацію коду, масштабування та ефективне використання ресурсів. Написання оптимізованого коду, що забезпечує швидку обробку завдань. Масштабування

системи для того, щоб ефективно розподіляти навантаження та збалансовано використовувати ресурси, що сприяє більшій продуктивності та підвищення досвіду користувача при користуванні продуктом.

3.3 Вимоги до бази даних

Забезпечення вимог до бази даних є важливим етапом у процесі розробки програмного забезпечення, оскільки це визначає основні принципи організації, зберігання та доступу до даних, необхідних для ефективної роботи системи.

Перш за все, важливо визначити структуру даних, яка відповідає потребам програми. Це включає в себе створення таблиць, визначення полів та їх типів, а також встановлення зв'язків між таблицями для забезпечення консистентності даних.

Далі, важливо нормалізувати дані, щоб уникнути дублювання та забезпечити їх цілісність. Це допомагає забезпечити ефективність роботи з базою даних та уникнути виникнення проблем при маніпулюванні даними.

Однією з ключових вимог є забезпечення безпеки даних. Це включає в себе встановлення прав доступу до даних, що забезпечує конфіденційність інформації, а також застосування методів шифрування для захисту даних від несанкціонованого доступу.

Крім того, важливо забезпечити швидкодію бази даних шляхом оптимізації запитів та використання індексів. Це допомагає забезпечити ефективну обробку даних та швидку відповідь на запити користувачів.

Не менш важливою є можливість резервного копіювання та відновлення даних. Це дозволяє відновити інформацію в разі втрати або пошкодження даних, що забезпечує безпеку та надійність роботи системи.

ДОДАТОК Г

Сертифікат учасника конференції «Print, Multimedia and Web» за доповідь «Аналіз трендів у сфері рекрутингу та їх відображення в функціональних можливостях додатків»