Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка веб-застосунку для оцінки співробітників»

Виконав: студент групи ІПЗ20-2
Спеціальність 121 «Інженерія програмного забезпечення»
Ковтун Н.А.
(прізвище та ініціали)
Керівник к.ф.-м.н., доц. Фірсов О.Д.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
Доцент кафедри кібербезпеки та інформаційних технологій
(посада)
к.т.н., доцент Флоров С.В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Ковтун Н.А. Розробка веб-застосунку для оцінки співробітників.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавра за спеціальністю 121 «Інженерія програмного забезпечення» – Університет митної справи та фінансів, Дніпро, 2024.

Кваліфікаційна робота присвячена розробці веб-застосунку для оцінки співробітників. Актуальність роботи зумовлена необхідністю автоматизації процесів оцінювання персоналу для підвищення продуктивності праці та об'єктивності оцінок. Веб-застосунок дозволяє керівництву компанії оперативно та точно аналізувати результати роботи співробітників, що має велике значення для управління персоналом у різних галузях, включаючи корпоративний сектор, освіту та державне управління.

У процесі дослідження було проведено аналіз сучасних методів розробки веб-додатків. Для реалізації системи було обрано технології ASP.NET, WCF, LINQ та інші інструменти платформи .NET, що забезпечують високу продуктивність та масштабованість. Вибір цих технологій обґрунтовано їхньою високою продуктивністю, безпекою та гнучкістю.Система включає клієнтську та серверну частини, інтеграцію з базою даних та механізми безпеки, що забезпечує спрощення розробки, підтримки та масштабування системи.

Розроблена система демонструє високу ефективність та продуктивність, а також забезпечує гнучкість у використанні та подальшому вдосконаленні. Результати роботи можуть бути використані для подальшого розвитку подібних систем та автоматизації процесів управління персоналом в інших сферах.

Ключові слова: оцінка співробітників, веб-застосунок, ASP.NET, WCF, .NET.

ABSTRACT

Kovtun N.A. Development of a Web Application for Employee Evaluation.

Bachelor's thesis for obtaining a degree in Software Engineering, specialty 121. – University of Customs and Finance, Dnipro, 2024.

This bachelor's qualification work is dedicated to the development of a web application for employee evaluation. The relevance of this work is driven by the necessity to automate the processes of personnel evaluation to increase productivity and objectivity of assessments. The web application allows the company's management to promptly and accurately analyze the results of employees' work, which is crucial for personnel management in various sectors, including corporate, education, and public administration.

During the research, an analysis of modern methods for developing web applications was conducted. For the implementation of the system, technologies such as ASP.NET, WCF, LINQ, and other tools of the .NET platform were chosen, providing high performance and scalability. The choice of these technologies is justified by their high productivity, security, and flexibility. The system includes client and server parts, integration with the database, and security mechanisms, ensuring simplified development, maintenance, and scaling of the system.

The developed system demonstrates high efficiency and productivity, as well as flexibility in use and further improvement. The results of this work can be utilized for the further development of similar systems and automation of personnel management processes in other fields.

Keywords: employee evaluation, web application, ASP.NET, WCF, .NET.

ЗМІСТ

ВСТУП

Актуальність проблеми. У сучасному бізнес-середовищі ефективне управління людськими ресурсами стає все більш важливим фактором для забезпечення конкурентоспроможності підприємств. Системи оцінювання працівників є ключовим інструментом, який дозволяє керівництву об'єктивно оцінювати внесок кожного співробітника в загальний успіх компанії, ідентифікувати таланти та визначати області, що потребують покращення. Це сприяє не лише підвищенню продуктивності праці, але й покращенню мотивації та залученості співробітників.

Зі стрімким розвитком технологій та цифровізацією бізнес-процесів виникає необхідність у створенні спеціалізованих програмних рішень, які б автоматизували процес оцінювання працівників. Використання таких додатків дозволяє суттєво знизити витрати часу та ресурсів на проведення оцінювання, забезпечити прозорість та об'єктивність цього процесу, а також надавати керівництву актуальні та точні дані для прийняття управлінських рішень.

Крім того, тенденції до віддаленої роботи та гнучких робочих графіків роблять додаток для оцінювання працівників надзвичайно важливим для підтримки ефективної комунікації та координації між членами команди. Це особливо актуально для великих корпорацій та міжнародних компаній, де працівники можуть бути розподілені по різних офісах та навіть країнах. Додаток дозволяє оперативно збирати та аналізувати дані з різних джерел, забезпечуючи комплексний підхід до оцінювання.

Інтуїтивно зрозумілий та функціонально насичений додаток для оцінювання працівників відповідає потребам сучасного ринку праці, де дедалі більшу роль відіграють молоді спеціалісти, звиклі до використання новітніх технологій та цифрових інструментів. Такий додаток може стати значною конкурентною перевагою для компаній у залученні та утриманні талановитих

працівників, забезпечуючи їм зручний та ефективний інструмент для розвитку кар'єри.

Метою роботи є автоматизація роботи оцінки співробітників.

Методи дослідження: обробка та аналіз інформації, методи проектування та розробки веб-додатків.

У відповідності до поставленої мети в кваліфікаційній роботі поставлені наступні завдання дослідження:

1. Проаналізувати технічні засоби, що застосовуються для розробки веб-додатків додатків.

2. Розробити архітектуру та функціональні можливості веб-додатку.

3. Розробити веб- додаток з застосуванням технології ASP.Net.

4. Провести тестування.

Об'єктом дослідження є розробка програмного забезпечення в сфері надання управлінських послуг.

Предметом дослідження є апаратно-програмне забезпечення для розробки веб-додатку.

Структура роботи:

- Розділ 1 Аналіз існуючих рішень.
- Розділ 2 Аналіз засобів розробки веб-застосунку оцінки співробітників
- Розділ 3 Розробка веб-застосунку оцінки співробітників

Робота складається зі вступу, 3-х розділів, висновків, списку використаних джерел з 15 найменувань. Обсяг роботи 60 сторінок кваліфікаційної роботи, 42 рисунків, 1 таблиця.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Поняття веб-додатку

Веб-додатки стали невід'ємною частиною сучасного цифрового світу, пропонуючи широкий спектр функціональних можливостей та послуг користувачам. Їхня гнучкість та доступність роблять їх цінними інструментами для різних цілей, від інформаційних ресурсів до електронної комерції та соціальних мереж.

Веб-сервіс – це інформаційна система що доступна через мережу Інтернет та може надавати простий або складний функціонал для користувачів [8].

Він може інтегруватись з різними додатками та мережевими протоколами, такими як SOAP, XML-RPC, або REST. Веб-сервіси мають інтерфейс, описаний у машинно-обчислювальному форматі (WSDL) і зазвичай використовують HTTP/ HTTPS протоколи. Поширені приклади веб-сервісів включають:

- пошукові системи,
- веб-хостинги,
- веб-пошта,
- веб-архіви.

Головною перевагою веб-сервісів є можливість доступу до даних з будь-якої точки світу, де є глобальна мережа.

Веб-додаток - це програмне забезпечення, що працює в браузері та доступне через Інтернет. Він складається з взаємопов'язаних веб-сторінок, які містять текст, зображення, аудіо, відео та інші дані. На відміну від статичних веб-сайтів, веб-додатки динамічно генерують контент та реагують на дії користувачів, забезпечуючи інтерактивний досвід.

Структура веб-додатку описує організацію його контенту та навігаційні шляхи, що впливають на загальний досвід користувача. Ретельне планування структури є ключовим для забезпечення зручного та інтуїтивно зрозумілого інтерфейсу.

Типи структур веб-додатків

1) Стандартна структура

Використовує головну сторінку як центральний вузол, з якого відгалужуються інші сторінки. Вона забезпечує просту та чітку навігацію.

2) Каскадна структура

Організовує контент у одній послідовності, подібно до книги або інструкції. Користувач може переміщатися вперед або назад, зберігаючи логіку подання інформації.

3) Структура «Хмарочос»

Ця модель використовує ієрархічний підхід, де доступ до наступного рівня контенту стає можливим лише після перегляду попереднього. Вона нагадує переміщення між поверхами хмарочосу.

Вибір структури

Оптимальний вибір структури веб-додатку залежить від його мети, цільової аудиторії та типу контенту. Важливо враховувати такі фактори:

- Має бути інтуїтивно зрозумілою, щоб користувачі могли легко знаходити потрібну інформацію.
- Пповинна відповідати послідовності подання інформації та відповідати цілям веб-додатку.
- Різні елементи контенту повинні бути чітко пов'язані між собою, щоб забезпечити логічний перехід між сторінками [6].

1.2. Аналіз методів та підходів розробки веб-додатків

Створення веб-додатків – це комплексний процес, що складається з чітко окреслених етапів:

- Аналіз вимог
- Проектування
- Реалізація
- Тестування

- Впровадження
- Підтримка

Кожен етап має свої завдання, які в сукупності гарантують створення якісного та функціонального продукту.

Розробка веб-застосунків охоплює три основні аспекти:

- Клієнтська сторона – відповідає за інтерфейс та взаємодію з користувачем. Реалізується зазвичай за допомогою HTML, CSS та JavaScript. Фреймворки, такі як Angular, React або Vue.js, полегшують розробку складних інтерфейсів.
- Серверна сторона – обробляє запити від клієнта, керує даними та виконує бізнес-логіку. Популярні серверні технології: Node.js, Ruby on Rails, Django, ASP.NET. Вибір технології залежить від вимог проекту, обсягу даних та необхідної продуктивності.
- Бази даних – забезпечує зберігання, управління та доступ до даних. Використовуються реляційні (MySQL, PostgreSQL) та NoSQL (MongoDB, Cassandra) бази даних. Вибір технології баз даних залежить від характеру даних, обсягів зберігання та вимог до масштабованості.

Існує декілька моделей розробки програмного забезпечення, кожна з яких має свої особливості:

- Водоспадна модель

Передбачає послідовне виконання етапів без можливості повернення до попередніх. Ця модель підходить для проектів з чітко визначеними вимогами, але може бути неефективною для складних та динамічних проектів [5].

- Ітеративні моделі

Розбивають процес на ітерації, де кожна з них включає всі основні етапи життєвого циклу та орієнтована на розробку окремого компоненту системи. Цей підхід більш гнучкий, адже дозволяє поступово розширювати функціональність, додавати нові компоненти та вносити зміни на кожному етапі.

Архітектурні патерни – це типові рішення для організації структури програмного забезпечення. Деякі з популярних патернів:

- Model-View-Controller (MVC) – розділяє додаток на три компоненти: Model (модель), View (подання) і Controller (контролер). Цей патерн полегшує підтримку та розширення додатку, але може ускладнювати управління залежностями між компонентами.
- Model-View-ViewModel (MVVM) – поділяє додаток на Model (модель), View (подання) і ViewModel (модель подання). ViewModel виступає посередником між моделлю та поданням, забезпечуючи зв'язок даних та логіку поведінки. MVVM дає можливість повторного використання коду та полегшує автоматизоване тестування, але може ускладнювати підтримку великих проектів [14].

1.3. Висновок до першого розділу

У першому розділі роботи було проведено всебічний аналіз предметної області, що охоплює поняття щодо розробки веб-додатків та методів їх створення. Веб-додаток - це програмне забезпечення, що працює в браузері та доступне через Інтернет. Він складається з взаємопов'язаних веб-сторінок, які містять текст, зображення, аудіо, відео та інші дані. На відміну від статичних веб-сайтів, веб-додатки динамічно генерують контент та реагують на дії користувачів, забезпечуючи інтерактивний досвід.

Проведений аналіз дозволив визначити основні тенденції та підходи в розробці веб-додатків, що використовуються в сучасних умовах.

Встановлено, що веб-додатки є невід'ємною частиною сучасного цифрового світу, пропонуючи широкий спектр функціональних можливостей та послуг користувачам. Їх гнучкість та доступність роблять їх цінними інструментами для різних цілей, від інформаційних ресурсів до електронної комерції та соціальних мереж.

Розглянуто три основні аспекти, які охоплює розробка веб-застосунків: клієнтська сторона, яка відповідає за інтерфейс та взаємодію з користувачем, серверна сторона, яка обробляє запити від клієнта, керує даними та виконує

бізнес-логіку, бази даних, які забезпечуюють зберігання, управління та доступ до даних.

Було розглянуто різні методи та підходи до розробки веб-додатків, включаючи використання таких фреймворків, як Angular, React, Vue.js та інших. Виявлено, що вибір фреймворку залежить від специфічних вимог проекту, таких як продуктивність, зручність розробки та доступність ресурсів.

Загалом, проведений аналіз підтверджує, що розвиток веб-технологій є важливим напрямком у сучасній розробці програмного забезпечення, який дозволяє значно підвищити ефективність розробки та забезпечити високу якість кінцевого продукту.

РОЗДІЛ 2. АНАЛІЗ ЗАСОБІВ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ОЦІНКИ СПІВРОБІТНИКІВ

2.1. Вибір програмних засобів для реалізації проекту

При розробці веб-додатку для системи оцінки співробітників важливо обрати оптимальний технологічний стек, що забезпечить ефективність, масштабованість і безпеку. Розглянемо три варіанти стеку, які активно використовуються у сучасній веб-розробці

MERN – складається з MongoDB, Express.js, React та Node.js. Цей стек дозволяє створювати повноцінні додатки, використовуючи лише JavaScript. MongoDB забезпечує зберігання даних у гнучкому форматі JSON, а React дозволяє створювати інтерактивні інтерфейси користувача. Express.js і Node.js забезпечують швидкість і ефективність роботи на сервері [11].

LAMP – входять Linux, Apache, MySQL і PHP. Це один з найстаріших і найбільш перевірених стеків для веб-розробки. Він добре підходить для створення динамічних веб-додатків, забезпечуючи надійність і безпеку. Linux та Apache забезпечують стабільне серверне середовище, MySQL — реляційне управління даними, а PHP – серверну логіку додатка [12].

MEAN, що включає MongoDB, Express.js, Angular та Node.js. Як і MERN, цей стек використовує JavaScript на всіх рівнях архітектури. Angular, розроблений Google, дозволяє створювати потужні односторінкові додатки (SPA), а поєднання з MongoDB, Express.js і Node.js забезпечує високопродуктивний серверний і клієнтський код [13].

Стека на основі технологій Microsoft, зокрема:

- ASP.NET дозволяє створювати надійні та масштабовані веб-додатки з використанням мови C#.
- LINQ (Language Integrated Query) забезпечує потужні можливості для роботи з даними, дозволяючи писати запити до баз даних безпосередньо в коді C#, що підвищує його читабельність і зручність.

- WCF (Windows Communication Foundation) забезпечує створення сервіс-орієнтованої архітектури, що дозволяє ефективно управляти взаємодією між компонентами системи, забезпечуючи безпеку та надійність передачі даних [14].

Такий стек технологій забезпечує високу продуктивність, безпеку та гнучкість у створенні сучасних веб-додатків для управління проектами.

2.2. Програмні засоби для розробки веб-додатку

.NET 6 є важливою віхою у розвитку платформи .NET, об'єднуючи різні технології та інструменти в єдину кросплатформену екосистему. Цей реліз був створений для забезпечення високої продуктивності, зручності використання та гнучкості для розробників, дозволяючи створювати різноманітні типи додатків, включаючи веб-додатки, десктопні програми, мобільні додатки, ігри та хмарні рішення.

Однією з основних характеристик .NET 6 є підтримка кросплатформеності. Ця платформа дозволяє розробникам писати код, який може виконуватись на різних операційних системах, таких як Windows, macOS та Linux, що робить її дуже зручною для розробки сучасних додатків. Розробники можуть використовувати єдину базу коду для різних платформ, зменшуючи час та витрати на розробку.

.NET 6 інтегрує декілька важливих компонентів, включаючи ASP.NET Core, Entity Framework Core, та Xamarin, що дозволяє створювати високопродуктивні веб-додатки, працювати з базами даних та створювати мобільні додатки. ASP.NET Core, наприклад, є потужним фреймворком для створення веб-додатків і API, який забезпечує високу продуктивність і масштабованість. Entity Framework Core надає зручні інструменти для роботи з базами даних, дозволяючи розробникам легко виконувати CRUD-операції (створення, читання, оновлення, видалення) [1].

Окрім цього, .NET 6 включає в себе значні покращення в продуктивності. Компанія Microsoft постійно працює над оптимізацією платформи, щоб забезпечити швидке виконання коду та зменшити використання ресурсів. Це робить .NET 6 ідеальним вибором для створення високонавантажених систем, які потребують максимальної продуктивності.

Розширені можливості інструментів розробки також є важливою частиною .NET 6. Інтеграція з Visual Studio забезпечує зручне середовище для розробників, з потужними засобами налагодження, автодоповненням коду та іншими функціями, що покращують продуктивність роботи. Також .NET 6 підтримує нові функції мови програмування C#, включаючи покращення у роботі з шаблонами, спрощення синтаксису та нові можливості для асинхронного програмування.

Безпека є ще одним ключовим аспектом .NET 6. Платформа включає в себе нові механізми захисту даних, забезпечуючи надійний захист додатків від різних загроз. Розробники можуть використовувати вбудовані функції для аутентифікації та авторизації, шифрування даних та інших заходів безпеки.

ASP.NET (Active Server Pages для .NET) – це безкоштовна платформа від Microsoft для розробки веб-сайтів та веб-застосунків. Вона використовує HTML, CSS та JavaScript, а також надає можливість створювати веб-API та використовувати технології реального часу, такі як веб-сокеты.

Основні можливості ASP.NET:

- Створення динамічних веб-сторінок
- Підтримка MVC
- Висока продуктивність та масштабованість
- Безпека

Переваги використання ASP.NET:

- Безкоштовно та з відкритим кодом
- Велике співтовариство
- Широкий спектр інструментів та компонентів
- Інтеграція з іншими технологіями Microsoft [3]

Windows Communication Foundation (WCF) — це платформа для створення та використання сервісів з використанням .NET Framework, яка дозволяє розробникам створювати надійні, масштабовані та безпечні додатки для зв'язку між клієнтами та серверами.

Компоненти WCF

1. Service (Сервіс) – основна одиниця виконання логіки в WCF. Серверна програма, яка приймає запити і відправляє відповіді.

2. Endpoint – точка доступу до сервісу, яка складається з адресита контракту.

3. Contract – описує операції, які може виконувати сервіс. Включає в себе:

- Service Contract – для опису основних методів сервісу,
- Data Contract – для опису даних, що передаються),
- Message Contract – для контролю над SOAP-повідомленнями
- Fault Contract – для обробки помилок.

4. Binding – визначає, як сервіс і клієнт будуть комунікувати. Включає налаштування протоколів, механізмів безпеки та інших параметрів.

5. Behavior – дозволяє налаштовувати додаткові аспекти сервісу, такі як тайм-аути, потоки, інстанції та інші властивості.

Переваги WCF

- Уніфікація
- Масштабованість та продуктивність
- Інтеграція з іншими технологіями Microsoft

Недоліки WCF

- Складність
- Microsoft більше не розвиває WCF

WCF залишається потужним інструментом для розробки сервісів в середовищі .NET, особливо для внутрішніх корпоративних додатків та сценаріїв, що вимагають високого рівня безпеки та надійності [7].

LINQ (Language Integrated Query) — це технологія в .NET, яка дозволяє виконувати запити до різних типів даних безпосередньо у межах мови програмування C#. Вона забезпечує інтеграцію запитів на рівні синтаксису мови, що робить код більш читабельним і зручним для написання. Основна перевага LINQ полягає в його універсальності: його можна використовувати з різними джерелами даних, такими як колекції об'єктів, бази даних, XML-документи тощо.

Запити LINQ складаються з трьох основних частин: джерела даних, операторів запиту та виконання запиту. Джерело даних – це колекція або інше джерело, до якого застосовуються запити. Оператори запиту включають стандартні методи, такі як Where, Select, OrderBy, які дозволяють фільтрувати, проєктувати, сортувати та об'єднувати дані. Запит LINQ не виконується до тих пір, поки не відбудеться ітерація по його результатах (відкладене виконання) або не буде викликаний метод, що примусово виконує запит, наприклад, ToList.

Основні переваги LINQ включають його здатність покращувати продуктивність та читабельність коду, дозволяючи писати менше коду для виконання тих самих завдань. Завдяки інтеграції у мову C#, запити LINQ отримують переваги компіляції та автодоповнення. Сильна типізація забезпечує перевірку коректності запитів на етапі компіляції, що знижує кількість помилок [2].

Visual Studio Community 2022 пропонує розробникам потужний набір інструментів для відладки та тестування додатків. Завдяки відладчику, ви можете виконувати покрокову відладку, аналізувати значення змінних у реальному часі, встановлювати точки зупинки та досліджувати стек викликів, що значно спрощує процес діагностики та виправлення помилок. Платформа також підтримує модульне тестування і інтеграцію з популярними тестовими фреймворками, такими як NUnit, MSTest і xUnit, надаючи всі необхідні засоби для автоматизованого тестування та забезпечення високої якості програмних продуктів.

Інтеграція з Git та GitHub дозволяє легко працювати з системами контролю версій прямо з Visual Studio. Ви можете створювати, клонувати, комітити та пушити репозиторії, вирішувати конфлікти та працювати з гілками коду, що робить процес розробки більш ефективним та організованим. Крім того, Visual Studio Community 2022 інтегрується з хмарними платформами, такими як Microsoft Azure, що значно спрощує створення, розгортання та управління хмарними додатками.

Visual Studio підтримує велику кількість розширень, що дозволяє додавати нові функціональні можливості та налаштовувати IDE під конкретні потреби розробника. Marketplace Visual Studio пропонує тисячі розширень для різних мов програмування, фреймворків та інструментів. Інтеграція з Xamarin дозволяє розробляти кросплатформні мобільні додатки для iOS та Android з використанням єдиного коду на C#, що значно спрощує процес розробки та підтримки мобільних додатків.

Visual Studio Community 2022 також підтримує розробку ігор за допомогою Unity та Unreal Engine, що дозволяє створювати високоякісні 2D та 3D ігри. Крім того, IDE підтримує сучасні технології та фреймворки, такі як .NET 6, C# 10, ASP.NET Core, Blazor та інші, що дозволяє використовувати найновіші інструменти та методики для створення інноваційних додатків.

Однією з ключових переваг Visual Studio Community 2022 є те, що вона безкоштовна для індивідуальних розробників, студентів, відкритих проектів та навчальних закладів. Це робить її доступною для широкого кола користувачів. Інтеграція з іншими продуктами Microsoft, такими як Azure, Office 365 та GitHub, розширює можливості розробки, дозволяючи використовувати широкий спектр сервісів та інструментів в єдиному середовищі, що підвищує продуктивність та спрощує процес розробки. Велика та активна спільнота розробників забезпечує доступ до численних ресурсів, включаючи документацію, форуми, відеоуроки та блоги, що полегшує процес навчання та вирішення проблем.

Visual Studio Community 2022 є потужним і універсальним середовищем розробки, яке надає всі необхідні інструменти для створення високоякісних

додатків для різних платформ. Її безкоштовність, багатий набір функцій, підтримка сучасних технологій та інтеграція з хмарними сервісами роблять її ідеальним вибором як для новачків, так і для досвідчених розробників, які прагнуть використовувати найновіші інструменти та методи для створення передових програмних рішень.

2.3. Архітектура додатку

Монолітна архітектура - це підхід до розробки програмного забезпечення, де всі компоненти програми тісно пов'язані та функціонують як єдине ціле. Цей підхід ідеально підходить для невеликих проектів з обмеженим функціоналом, оскільки він забезпечує простоту реалізації, тестування та розгортання.

Однак, з розширенням проекту та додаванням нового функціоналу, монолітна архітектура може стати проблематичною. Збільшення складності коду, кількості фреймворків та таблиць баз даних ускладнює навігацію та підтримку проекту. Кожна зміна вимагає перебудови та повторного тестування всього проекту, що збільшує ризик виникнення помилок та ускладнює перехід на нові технології.

Переваги монолітної архітектури:

- Простота реалізації та розгортання
- Порівняно низька вартість розгортання
- Легкий процес тестування

Недоліки монолітної архітектури:

- Збільшення складності підтримки з ростом проекту
- Необхідність перебудови всього проекту при внесенні змін
- Підвищений ризик виникнення помилок
- Складність переходу на нові технології

Мікросервісна архітектура - це сучасний підхід до розробки програмного забезпечення, який спрощує традиційну сервісоорієнтовану архітектуру (SOA).

Вона передбачає розбиття застосунку на невеликі, незалежні сервіси, кожен з яких виконує одну конкретну функцію.

Ці мікросервіси можуть бути написані на різних мовах програмування та використовувати різні технології зберігання даних, що забезпечує високу гнучкість та адаптивність. Кожен мікросервіс має власну базу даних та працює в окремому процесі, що дозволяє розгортати та оновлювати їх незалежно один від одного.

Ця архітектура забезпечує ряд переваг, включаючи легкість внесення змін, високу відмовостійкість, простоту підтримки та відсутність прив'язки до конкретних технологій. Завдяки незалежності мікросервісів, вихід з ладу одного з них не впливає на роботу всього застосунку.

Однак, мікросервісна архітектура має і свої недоліки. Тестування такої системи є складнішим завданням, оскільки необхідно забезпечити коректну взаємодію між численними сервісами. Крім того, реалізація надійної системи координації між мікросервісами може бути складною, а фінансові витрати на підтримку такої архітектури зростають зі збільшенням кількості сервісів.

Ключові характеристики мікросервісів:

- Функціональна спрямованість

Кожен мікросервіс відповідає за одну конкретну функцію.

- Незалежне розгортання

Мікросервіси можна розгортати та оновлювати незалежно один від одного.

- Ізоляція процесів

Кожен мікросервіс працює в окремому процесі або групі процесів.

- Власне сховище даних

Кожен мікросервіс має власну базу даних.

- Легка заміна

Мікросервіси можна легко замінити або оновити без впливу на інші компоненти системи.

- Простота підтримки

Невелика команда розробників може ефективно підтримувати декілька мікросервісів.

Мікросервісна архітектура набуває все більшої популярності завдяки своїм перевагам у сфері масштабованості, гнучкості та надійності. Вона особливо добре підходить для великих та складних застосунків, де важлива можливість швидкого внесення змін та адаптації до нових вимог [5].

Сервіс-орієнтована архітектура (SOA) - це модульний підхід до розробки програмного забезпечення, який виник наприкінці 1980-х років. Він базується на використанні окремих сервісів (служб) зі стандартизованими інтерфейсами, що дозволяє уникнути дублювання функціональності та забезпечити повторне використання компонентів.

SOA прагне до централізації процесів та уніфікації операцій, сприяючи переходу компанії до функціональної організації на основі промислової платформи інтеграції. Однак, на відміну від мікросервісної архітектури, SOA передбачає тісну зв'язаність між сервісами, що може ускладнювати внесення змін та розширення системи.

Основні відмінності між SOA та мікросервісною архітектурою:

- Зв'язаність

SOA базується на зв'язаності сервісів, тоді як мікросервісна архітектура прагне до їх незалежності, дозволяючи навіть дублювання логіки для забезпечення автономності кожного мікросервісу.

- Внесення змін

У SOA внесення змін до одного сервісу може вимагати змін у всіх пов'язаних сервісах, що ускладнює процес розробки та підтримки. У мікросервісній архітектурі зміни в одному мікросервісі не впливають на інші.

- Тестування

Тестування SOA є складнішим процесом, оскільки вимагає перевірки не лише окремого сервісу, але й усіх залежних від нього сервісів. У мікросервісній архітектурі тестування кожного мікросервісу можна проводити незалежно.

Переваги SOA:

- Повторне використання

Сервіси можна використовувати в різних застосунках, що зменшує дублювання коду та прискорює розробку.

- Стандартизація

Використання стандартизованих інтерфейсів спрощує інтеграцію сервісів.

- Централізація

SOA сприяє централізації бізнес-процесів та уніфікації операцій.

Недоліки SOA:

- Складність внесення змін

Зміни в одному сервісі можуть вимагати змін у всіх пов'язаних сервісах.

- Складність тестування

Тестування вимагає перевірки не лише окремого сервісу, але й усіх залежних від нього сервісів.

- Зв'язаність

Тісна зв'язаність сервісів може ускладнювати масштабування та модернізацію системи.

При виборі архітектури для веб-застосунку оцінки співробітників важливо враховувати різні фактори, такі як розмір проекту, вимоги до масштабованості, гнучкості та бюджет. Монолітна та мікросервісна архітектури пропонують різні підходи до розробки, кожен з яких має свої переваги та недоліки [4].

Таблиця 2.1

Порівння сучасних архітектур веб-додатків

Фактор	Монолітна архітектура	Мікросервісна архітектура
Розмір проекту	Підходить для малих та середніх проектів	Підходить для середніх та великих проектів

Складність розробки	Менша складність на початковому етапі	Більша складність через необхідність розробки окремих сервісів та їх взаємодії
Масштабування	Складне та потребує перебудови всього проекту	Легке та здійснюється масштабуванням окремих мікросервісів
Гнучкість	Менш гнучка, зміни в одній частині впливають на інші	Більш гнучка, зміни в одному мікросервісі не впливають на інші
Розгортання	Розгортається весь проект цілком	Можливе незалежне розгортання окремих мікросервісів
Тестування	Простіше тестувати весь проект	Складніше тестувати через необхідність тестування взаємодії мікросервісів
Підтримка	Менш складна, адмініструється один проект	Більш складна, адмініструється багато окремих сервісів
Технології	Обмежена одним набором технологій	Можливість використання різних технологій для різних мікросервісів
Відмовостійкість	Вихід з ладу однієї частини може вплинути на весь проект	Вихід з ладу одного мікросервісу може не вплинути на інші

Вартість	Менша початкова вартість розробки	Може бути більшою початковою вартістю через складність розробки

2.3. Висновок до другого розділу

У розділі два було проведено аналіз різних підходів до реалізації веб-застосунку оцінки співробітників. Були розглянуті різні технологічні стеки, такі як MERN, LAMP, MEAN та стек Microsoft, кожен з яких має свої переваги та недоліки залежно від вимог проекту.

Особливу увагу було приділено аналізу двох основних архітектурних підходів: монолітної та мікросервісної. Було виявлено, що монолітна архітектура краще підходить для невеликих проектів з обмеженим функціоналом, тоді як мікросервісна архітектура забезпечує більшу гнучкість, масштабованість та відмовостійкість для великих і складних систем.

Також було розглянуто різні програмні засоби для розробки кросплатформних додатків, такі як .NET 6, ASP.NET, WCF, LINQ та Visual Studio Community 2022. Ці інструменти пропонують широкий спектр можливостей для створення ефективних та зручних у використанні додатків.

Вибір конкретного технологічного стеку та архітектури залежить від специфіки проекту, його вимог та обмежень. Важливо враховувати всі фактори, щоб зробити обґрунтований вибір, який забезпечить успішну реалізацію та подальший розвиток веб-застосунку оцінки співробітників.

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ОЦІНКИ СПІВРОБІТНИКІВ

3.1 Актуальність розробки додатку

У сучасному бізнес-середовищі ефективне управління людськими ресурсами стає все більш важливим фактором для забезпечення конкурентоспроможності підприємств. Зокрема, системи оцінювання працівників є ключовим інструментом, який дозволяє керівництву об'єктивно оцінювати внесок кожного співробітника в загальний успіх компанії, ідентифікувати таланти, а також визначати області, що потребують покращення. Це сприяє не лише підвищенню продуктивності праці, але й покращенню мотивації та залученість співробітників.

У зв'язку з стрімким розвитком технологій та цифровізацією бізнес-процесів, виникає необхідність у створенні спеціалізованих програмних рішень, які б автоматизували процес оцінювання працівників. Використання таких додатків дозволяє суттєво знизити витрати часу та ресурсів на проведення оцінювання, забезпечити прозорість та об'єктивність цього процесу, а також надавати керівництву актуальні та точні дані для прийняття управлінських рішень.

Крім того, враховуючи тенденції до віддаленої роботи та гнучких робочих графіків, додаток для оцінювання працівників стає надзвичайно важливим для підтримки ефективної комунікації та координації між членами команди. Це особливо актуально для великих корпорацій та міжнародних компаній, де працівники можуть бути розподілені по різних офісах та навіть країнах. Додаток дозволяє оперативно збирати та аналізувати дані з різних джерел, забезпечуючи тим самим комплексний підхід до оцінювання [9].

На додаток до цього, розробка інтуїтивно зрозумілого та функціонально насиченого додатку для оцінювання працівників відповідає потребам сучасного ринку праці, де дедалі більшу роль відіграють молоді спеціалісти, звиклі до використання новітніх технологій та цифрових інструментів. Такий додаток

може стати значною конкурентною перевагою для компаній у залученні та утриманні талановитих працівників, забезпечуючи їм зручний та ефективний інструмент для розвитку кар'єри.

Отже, розробка додатку системи оцінювання працівників є надзвичайно актуальною та відповідає сучасним тенденціям розвитку бізнесу та управління персоналом. Вона сприяє підвищенню ефективності, прозорості та об'єктивності процесу оцінювання, що в кінцевому результаті позитивно впливає на загальну продуктивність та успіх компанії.

3.2 Структура проекту

Загальна структура проекту, передбачає взаємодію представлень ASP.net та використання в якості бізнес логіки серверну архітектуру WCF, який взаємодіє з базою даних в якості моделей:

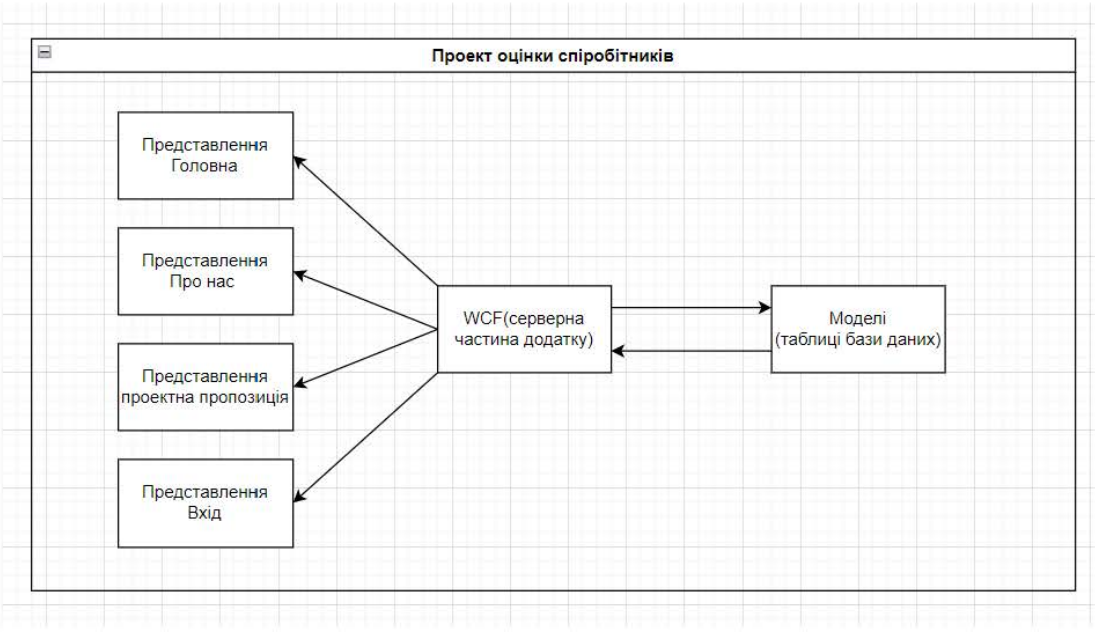


Рисунок 3.1 – Структура проекту

Таким чином серверна частина може представляти собою, як перехідна служба між моделями тобто таблицями. Дана система дозволяє вирішити

проблему зберігання даних при переході на наступне представлення, якщо постійно працює серверна частина додатку.

3.3 Технології розробки додатку

Windows Communication Foundation (WCF) є потужною платформою для створення розподілених і взаємодіючих систем, яка була розроблена компанією Microsoft і входить до складу .NET Framework. WCF надає інструментарій для побудови сервіс-орієнтованих архітектур (SOA), які дозволяють різноманітним додаткам взаємодіяти між собою незалежно від платформи чи протоколу зв'язку.

Рисунок 3.2 – Технологія WCF

WCF підтримує різноманітні протоколи зв'язку, включаючи HTTP, HTTPS, TCP, IPC, MSMQ, що дозволяє створювати сервіси, які можуть працювати як у межах одного комп'ютера, так і через Інтернет. авдяки підтримці стандартів веб-служб, таких як SOAP, WSDL, і WS-* протоколи, WCF забезпечує високу ступінь сумісності з іншими платформами і технологіями, що робить його ідеальним для інтеграційних рішень.

WCF пропонує розширені можливості для забезпечення безпеки, включаючи аутентифікацію, авторизацію, конфіденційність і цілісність даних. Це дозволяє захищати дані на всіх етапах їх передачі та обробки.

WCF складається з декількох основних компонентів (рис. 3.3):

- Сервісні контракти (Service Contracts): Визначають інтерфейси, які описують методи, що надаються сервісом. Ці контракти встановлюють угоди між сервісом і його клієнтами щодо структури повідомлень і форматів даних.
- Контракти даних (Data Contracts): Описують структури даних, що використовуються для обміну інформацією між сервісом і клієнтами. Контракти даних гарантують, що обидві сторони мають однаковий формат даних.
- Контракти повідомлень (Message Contracts): Дозволяють налаштовувати формат повідомлень на низькому рівні, що важливо для забезпечення сумісності з іншими системами.
- Політики та зв'язки (Policies and Bindings): Визначають, як сервіс буде обмінюватися повідомленнями, включаючи використані транспортні протоколи, механізми безпеки і інші характеристики комунікації.

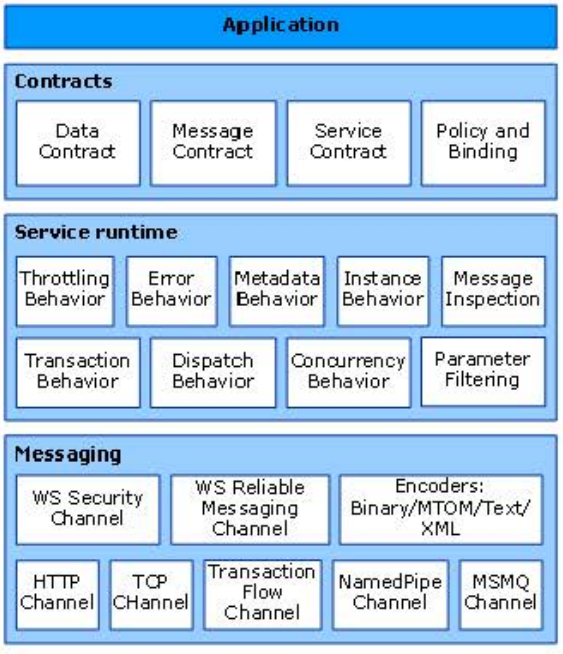


Рисунок 3.3 – Основні компоненти WCF

Використання WCF у поєднанні з ASP.NET забезпечує створення масштабованих рішень, які можуть обслуговувати велику кількість користувачів і обробляти значні обсяги даних.

ASP.NET є потужною веб-платформою, розробленою теж компанією Microsoft, яка дозволяє створювати динамічні веб-додатки. Будучи частиною .NET Framework, ASP.NET забезпечує розробникам широкий спектр інструментів і бібліотек для створення високопродуктивних і безпечних веб-додатків.

ASP.NET використовує компіляцію, кешування, управління станом і інші технології для підвищення продуктивності веб-додатків. Це забезпечує швидкий відгук і високу продуктивність навіть при значному навантаженні.

Дана платформа доволі гнучка в розвитку різноманітних архітектур за бажанням користувача, для цього ASP.net надає Web Forms, MVC (Model-View-Controller) і Web API, що значно покращує процес розробки різноманітних проектів.

Як вже зазначалося до цього, дану технологію розробила компанія Microsoft разом з .NET Framework та WCF, тобто ASP.NET має повну інтеграцію з цими технологіями, що надає доступ до широкого спектра бібліотек і компонентів для виконання різних завдань, від роботи з базами даних до управління безпекою.

ASP.NET підтримує використання HTML5, CSS3, JavaScript і інших сучасних веб-технологій, що дозволяє створювати інтерактивні та привабливі веб-додатки.

Серед цих технологій потрібно зазначити використання Bootstrap. При створенні нового проекту ASP.NET, технологія вже автоматично підключена як стартовий компонент з яким відразу можна працювати без додаткового завантаження.

Bootstrap – це популярний фреймворк для розробки адаптивних та мобільних веб-інтерфейсів, який надає розробникам набір інструментів для створення стильних і функціональних веб-сторінок. Використання Bootstrap у проектах ASP.NET значно спрощує створення інтерфейсів користувача завдяки готовим CSS і JavaScript компонентам.

Bootstrap забезпечує адаптивність веб-сторінок, автоматично підлаштовуючи їх вигляд під різні розміри екрану, включаючи настільні комп'ютери, планшети і мобільні пристрої (рис. 3.4) [15].

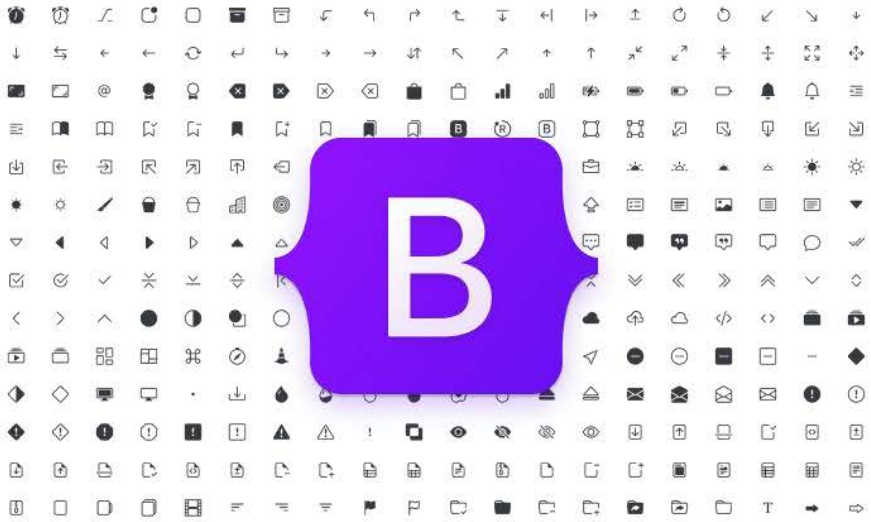


Рисунок 3.4 – Bootstrap

Звісно, що варто згадати і саму платформу .NET Framework, без якої перелічені вище технології не працювали.

.NET Framework (рис. 3.5) є комплексною і потужною програмною платформою, яка забезпечує всебічну підтримку для створення, розгортання і виконання додатків. .NET Framework об'єднує різноманітні технології і бібліотеки, що дозволяють розробникам створювати додатки для різних середовищ, включаючи настільні комп'ютери, сервери, мобільні пристрої та веб-додатки.

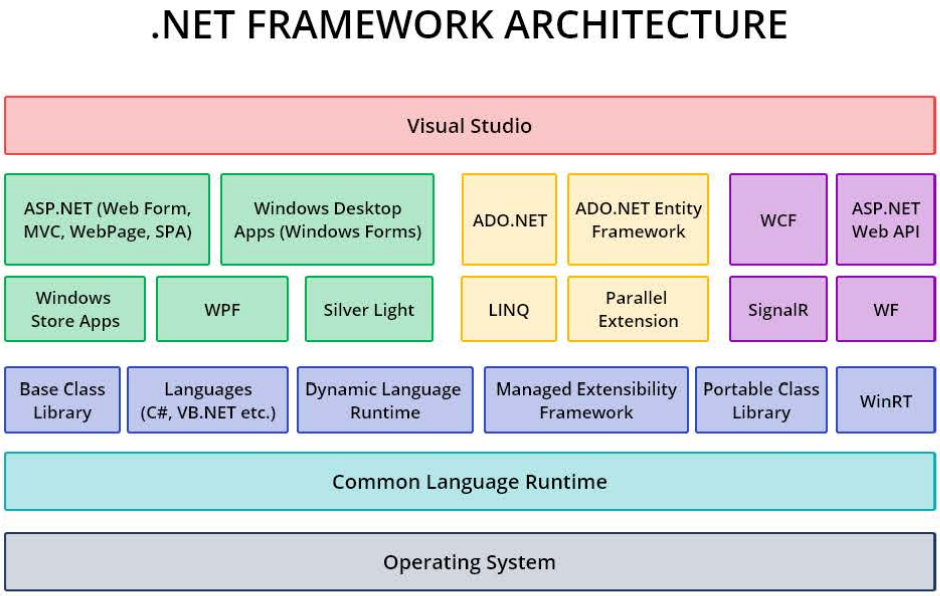


Рисунок 3.5 – Структура платформи .NET Framework

Загалом можна узагальнити, що .NET Framework, є важливим компонентом, який дає можливість створювати комбіновані проекти не турбуючись про сумісність даних чи бібліотек.

Для розробки таких проектів зі складною архітектурою потрібно використовувати середовище розробки, яке б дало б одночастно розробляти сумісні проекти в одному рішенні. Для створення проекту використовується Visual Studio.

Visual Studio (рис. 3.6), є інтегрованим середовищем розробки (IDE), що забезпечує розробникам широкий спектр інструментів для створення, налагодження та розгортання програмного забезпечення. Visual Studio підтримує різноманітні мови програмування і платформи, включаючи .NET, C#, VB.NET, C++, Python, JavaScript, TypeScript та інші. Воно є одним із найпотужніших і найбільш використовуваних IDE у світі, що забезпечує ефективний процес розробки для різних типів додатків.

Дане середовище розробки забезпечує розширені можливості для налагодження додатків, включаючи точкові зупинки (breakpoints), спостереження за змінними (watch variables), перегляд стеку викликів (call stack)

та інші інструменти для діагностики і виправлення помилок у коді. Також підтримує інтеграцію з популярними системами контролю версій, такими як Git, Azure DevOps, SVN та інші. Це дозволяє розробникам ефективно управляти змінами в коді, співпрацювати з командою і слідкувати за історією змін.

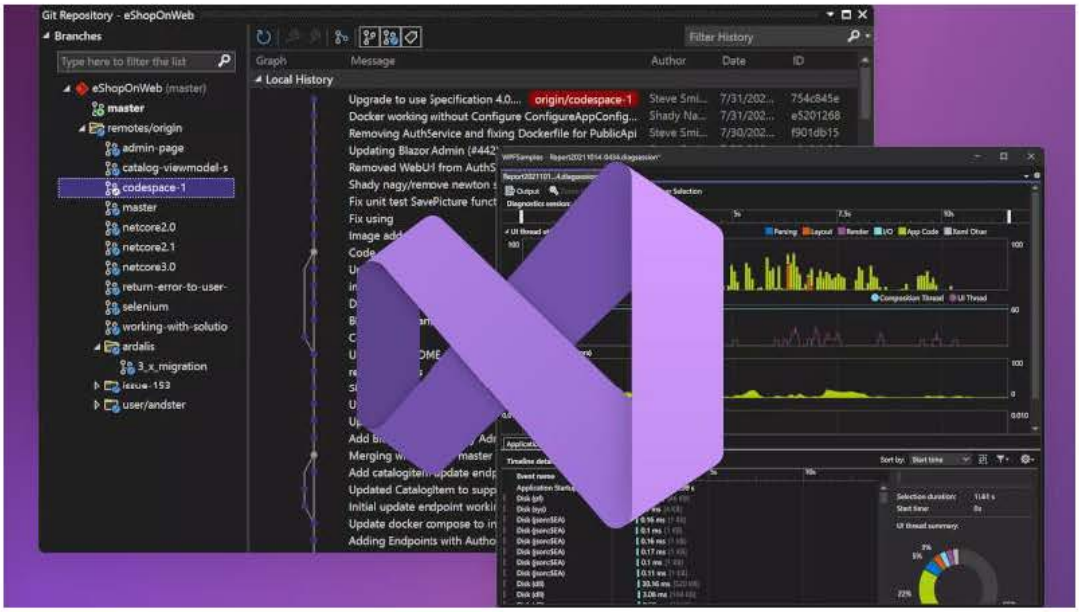


Рисунок 3.6 – Visual Studio

Архітектура Visual Studio дозволяє розширювати його функціональність за допомогою додатків (extensions), які можна завантажити з Visual Studio NuGet. Це дозволяє розробникам налаштовувати середовище під свої потреби і використовувати додаткові інструменти для підвищення продуктивності (рис. 3.7).

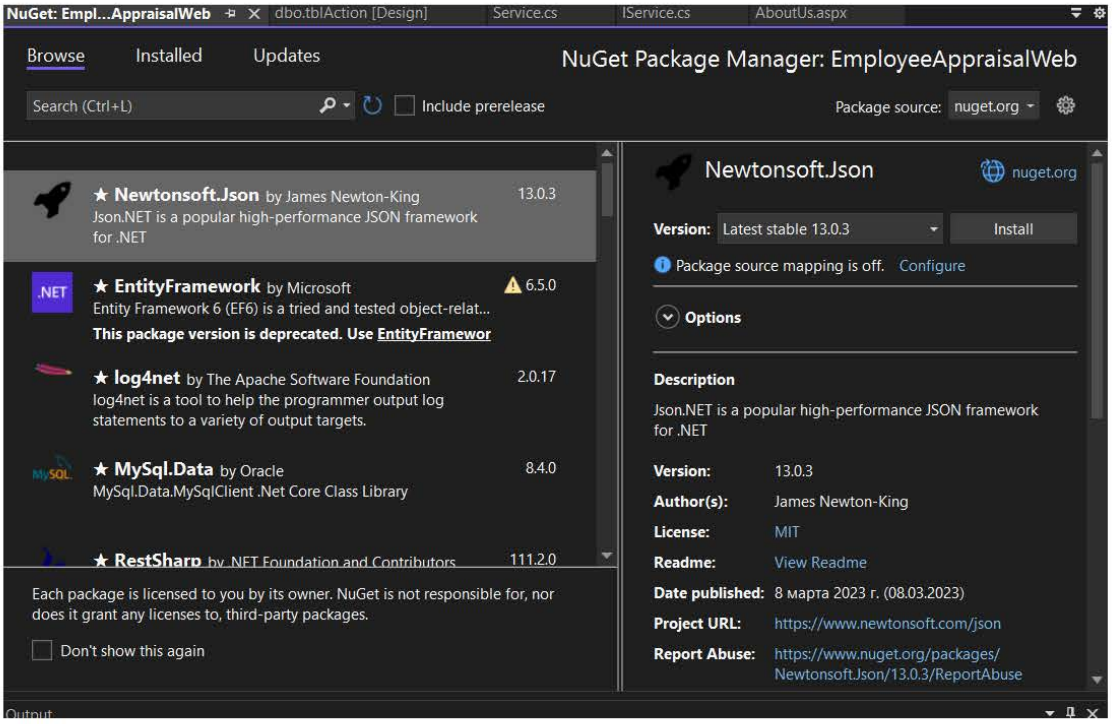


Рисунок 3.7 – Менеджер пакетів NuGet

3.4 Розробка додатку

Відповідно до діаграми класів, весь бізнес-функціонал виконує інтерфейс та клас який наслідує даний інтерфейс (рис. 3.8) [10].

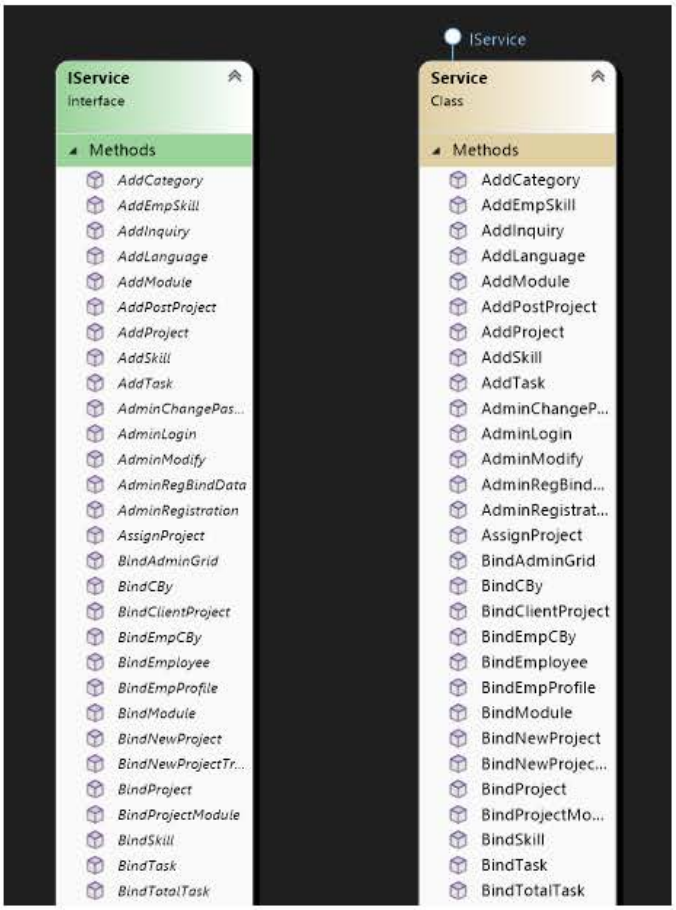


Рисунок 3.8 – Діаграма класів

Далі розглянемо клас Service, та розглянемо його найважливіші функції (рис. 3.9).

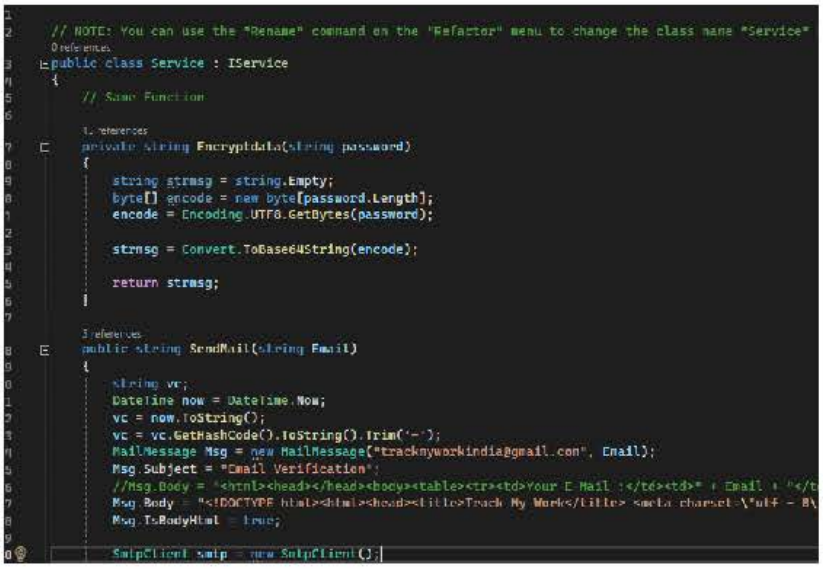


Рисунок 3.9 – Клас Service

Перша функція Encryptdata виконує базове шифрування рядка пароля за допомогою кодування в формат Base64. Спочатку вона оголошує змінну strmsg для зберігання зашифрованого рядка та масив байтів encode для тимчасового зберігання байтів пароля. Потім вона перетворює вхідний пароль у масив байтів за допомогою кодування UTF-8. Після цього байтовий масив перетворюється на рядок у форматі Base64, який присвоюється змінній strmsg і повертається як результат роботи функції (рис. 3.10).

```
15 references
private string Encryptdata(string password)
{
    string strmsg = string.Empty;
    byte[] encode = new byte[password.Length];
    encode = Encoding.UTF8.GetBytes(password);

    strmsg = Convert.ToBase64String(encode);

    return strmsg;
}
```

Рисунок 3.10 – Функція Encryptdata

Наступна функція SendMail відправляє електронний лист для верифікації користувача на вказану адресу електронної пошти. Спочатку вона генерує унікальний код верифікації, використовуючи поточний час і його хеш-код. Потім створюється об'єкт MailMessage, який налаштовується з відправником, одержувачем, темою та тілом повідомлення. Тіло листа містить HTML-код з вбудованим стилями для відображення верифікаційного коду і електронної адреси користувача (рис. 3.11).

```
3 references
public string SendMail(string Email)
{
    string vc;
    DateTime now = DateTime.Now;
    vc = now.ToString();
    vc = vc.GetHashCode().ToString().Trim('-');
    MailMessage Msg = new MailMessage("trackmyworkindia@gmail.com", Email);
    Msg.Subject = "Email Verification";
    //Msg.Body = "<html><head></head><body><table><tr><td>Your E-Mail :</td><td>" + Email + "</td></tr><tr><td>You
    Msg.Body = "<!DOCTYPE html><html><head><title>Track My Work</title> <meta charset=\"utf - 8\"><meta name=\"vie
    Msg.IsBodyHtml = true;

    SmtpClient smtp = new SmtpClient();
    smtp.Host = "smtp.gmail.com";
    smtp.Port = 587;
    smtp.UseDefaultCredentials = false;
    smtp.EnableSsl = true;
    smtp.DeliveryMethod = SmtpDeliveryMethod.Network;

    NetworkCredential MyCredentials = new NetworkCredential("trackmyworkindia@gmail.com", "TMW2016open");
    smtp.Credentials = MyCredentials;
    smtp.Send(Msg);
    return vc;
}
```

Рисунок 3.11 – Функція SendMail

Метод ViewSkill (рис. 3.12) використовується для отримання всіх записів таблиці tblSkill з бази даних, використовуючи LINQ to SQL. Він створює новий контекст даних DataClassesDataContext, який представляє з'єднання з базою даних, і виконує запит для вибірки всіх об'єктів tblSkill. Результат запиту повертається у вигляді IQueryable<tblSkill>, що дозволяє виконувати додаткові операції на даних перед їх фактичним виконанням у базі даних.

```
1 reference
public IQueryable<tblSkill> ViewSkill()
{
    var DC = new DataClassesDataContext();
    IQueryable<tblSkill> strskill = from obj in DC.tblSkills select obj;
    return strskill;
}
```

Рисунок 3.12 – Метод ViewSkill

Метод AddSkill приймає три параметри: назву навички (SkillName), ідентифікатор користувача, що створив запис (CBY), та вміння (SSkill). Він створює новий об'єкт tblSkill, заповнює його значеннями, встановлює дату і час створення, перетворює значення SSkill на ціле число, якщо воно не є null, та додає новий запис до таблиці tblSkill в базі даних, зберігаючи зміни за допомогою методу SubmitChanges (рис. 3.13).

```
1 reference
public void AddSkill(string SkillName, int CBY, string SSkill)
{
    var DC = new DataClassesDataContext();

    tblSkill skill = new tblSkill();
    skill.SkillName = SkillName;
    skill.CreatedBy = CBY;
    skill.CreatedOn = DateTime.Now;
    if (SSkill == null)
    {
        skill.SuperSkill = null;
    }
    else
    {
        skill.SuperSkill = Convert.ToInt32(SSkill);
    }
    DC.tblSkills.InsertOnSubmit(skill);
    DC.SubmitChanges();
}
```

Рисунок 3.13 – Метод AddSkill

Для роботи цієї функції використовується наступна таблиця бази даних (рис. 3.14):

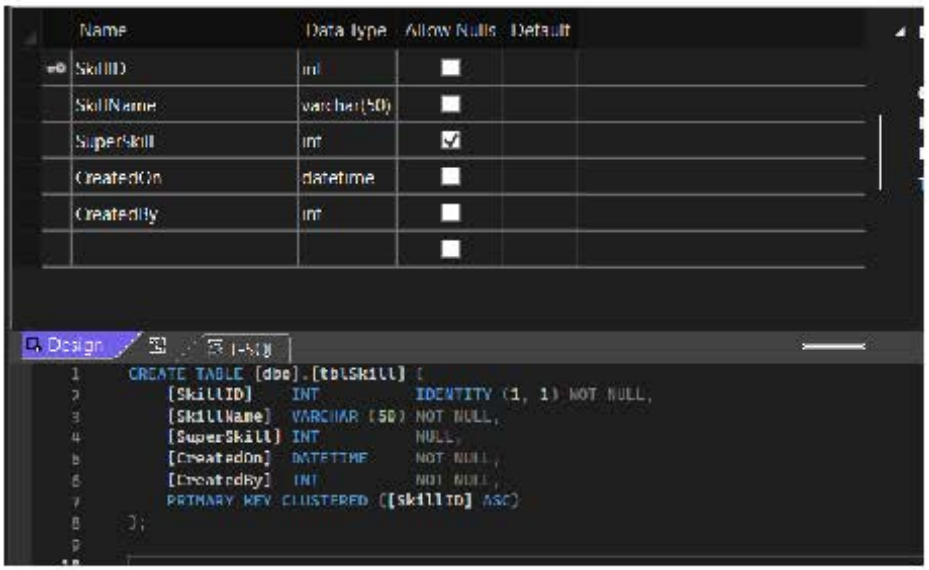


Рисунок 3.14 – Таблиця tblSkill

Наступний ResSubAdmin метод використовується для перевірки чи є адміністратор з заданим ідентифікатором (AdminID) суперадміністратором. Для цього використовується LINQ to SQL для запиту до бази даних і отримання інформації про адміністратора (рис. 3.15).

```
1 reference
public bool ResSubAdmin(int AdminID)
{
    var DC = new DataClassesDataContext();
    tblAdmin result = (from u in DC.tblAdmins
                       where u.AdminID == AdminID
                       select u).SingleOrDefault();
    return Convert.ToBoolean(result.IsSuper);
}
```

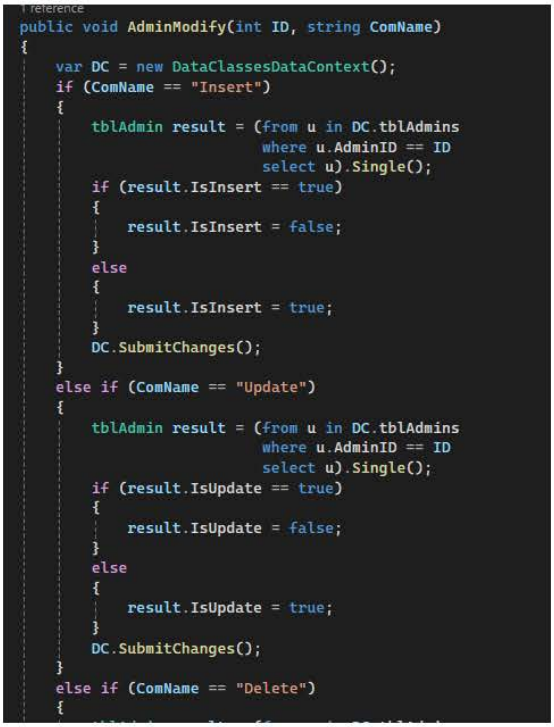
Рисунок 3.15 – Метод ResSubAdmin

Для отримання таких даних використовується таблиця tblAdmin, з параметром IsSuper (рис. 3.16):

	LastName	varchar(50)	☐
	EmailID	varchar(250)	☐
	Password	varchar(MAX)	☐
	ContactNo	varchar(15)	☐
	Image	varchar(MAX)	☑
	IsInsert	bit	☑
	IsUpdate	bit	☑
	IsDelete	bit	☑
	IsActive	bit	☐
	CreatedBy	int	☑
	CreatedOn	date	☐
	IsSuper	bit	☑

Рисунок 3.16 – Таблиця tblAdmin

AdminModify дозволяє змінювати певні атрибути запису адміністратора в базі даних на основі значення параметра ComName. Якщо ComName дорівнює "Insert", "Update" або "Delete", відповідне поле (IsInsert, IsUpdate або IsDelete) перемикається на протилежне значення (рис. 3.17).



```
1 reference
public void AdminModify(int ID, string ComName)
{
    var DC = new DataClassesDataContext();
    if (ComName == "Insert")
    {
        tblAdmin result = (from u in DC.tblAdmins
                           where u.AdminID == ID
                           select u).Single();
        if (result.IsInsert == true)
        {
            result.IsInsert = false;
        }
        else
        {
            result.IsInsert = true;
        }
        DC.SubmitChanges();
    }
    else if (ComName == "Update")
    {
        tblAdmin result = (from u in DC.tblAdmins
                           where u.AdminID == ID
                           select u).Single();
        if (result.IsUpdate == true)
        {
            result.IsUpdate = false;
        }
        else
        {
            result.IsUpdate = true;
        }
        DC.SubmitChanges();
    }
    else if (ComName == "Delete")
    {
```

Рисунок 3.17 – Метод AdminModify

Метод CMSInsert призначений для додавання нового запису до таблиці tblCM в базі даних. Він приймає три параметри: Title (заголовок контенту), content (вміст), та CBy (ідентифікатор користувача, який створив запис). Метод створює новий об'єкт tblCM, заповнює його властивостями, і зберігає новий запис в базі даних (рис. 3.18).

```
1 reference
public void CMSInsert(string Title, string content, int CBy)
{
    var DC = new DataClassesDataContext();
    tblCM tblCMS = new tblCM();
    tblCMS.Title = Title;
    tblCMS.Content = content;
    tblCMS.CreatedBY = CBy;
    tblCMS.CreatedOn = DateTime.Now;
    tblCMS.IsActive = true;
    DC.tblCMs.InsertOnSubmit(tblCMS);
    DC.SubmitChanges();
}
```

Рисунок 3.18 – Метод CMSInsert

Далі розглянемо метод, що додає новий проект в базу даних – AddProject. Він приймає сім параметрів: Title (назва проекту), ClientID (ідентифікатор клієнта), CatID (ідентифікатор категорії), LanID (ідентифікатор мови), Desc (опис проекту), AdminID (ідентифікатор адміністратора, який створив проект), і DeadLine (дата завершення проекту). Метод створює новий об'єкт tblProject, заповнює його властивостями, і зберігає новий запис в базі даних (рис. 3.19).

```
public void AddProject(string Title, int ClientID, int
{
    var DC = new DataClassesDataContext();

    tblProject Project = new tblProject();
    Project.Title = Title;
    Project.ClientID = ClientID;
    Project.CategoryID = CatID;
    Project.LanguageID = LanID;
    Project.DeadlineDate = DeadLine;
    Project.Description = Desc;
    Project.CreatedOn = DateTime.Now;
    Project.CreatedBy = AdminID;
    Project.IsActive = false;
    DC.tblProjects.InsertOnSubmit(Project);
    DC.SubmitChanges();
```

Рисунок 3.19 – Метод AddProject

Результатом даної функції, це додавання запису до бази даних в дану таблицю (рис. 3.20):

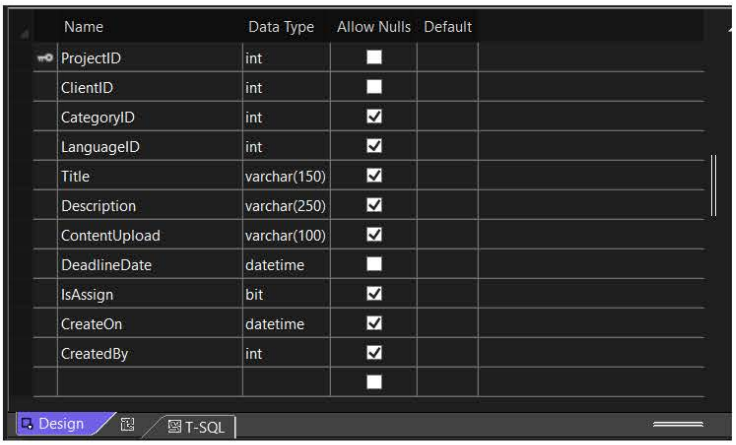


Рисунок 3.20 – Таблиця tblProjects

Для автентифікації в веб-додатку використовується функція AdminLogin. Функція перевіряє, чи існує запис адміністратора з заданими обліковими даними в базі даних, і повертає список, який містить кількість відповідних записів та ідентифікатор адміністратора (рис. 3.21).

```
1 reference
public IList<int> AdminLogin(string Email, string Pwd)
{

    var DC = new DataClassesDataContext();
    var st = (from ob in DC.tblAdmins
              where (ob.EmailID == Email.ToLower() && ob.Password == Encryptdata(Pwd))
              select ob).Count();

    var st1 = (from ob in DC.tblAdmins
               where (ob.EmailID == Email.ToLower() && ob.Password == Encryptdata(Pwd))
               select ob).FirstOrDefault();
    IList<int> Login = new int[] { st, st1.AdminID };

    return Login;
}
```

Рисунок 3.21 – Функція AdminLogin

Для зміни пароля передбачена наступна функція – ChangePwd. Дана функція перевіряє наявність адміністратора з заданою електронною поштою в базі даних, змінює пароль на нове значення, зашифроване за допомогою методу

Encryptdata, і повертає ідентифікатор адміністратора у вигляді рядка або null, якщо такого адміністратора не знайдено (рис. 3.22).

```
1 reference
public string ChangePwd(string Email, string Pwd)
{
    var DC = new DataClassesDataContext();
    var check = (from user in DC.tblAdmins
                 where user.EmailID == Email.ToLower()
                 select user.AdminID).Count();
    var AdminData = (from user in DC.tblAdmins
                     where user.EmailID == Email.ToLower()
                     select user).FirstOrDefault();
    if (check > 0)
    {
        tblAdmin result = (from u in DC.tblAdmins
                           where u.AdminID == AdminData.AdminID
                           select u).Single();
        result.Password = Encryptdata(Pwd);
        DC.SubmitChanges();
        return AdminData.AdminID.ToString();
    }
    else
    {
        return null;
    }
}
```

Рисунок 3.22 – Функція ChangePwd

Звісно, що крім авторизації адміністратора також передбачена функція для режстрації. Метод EmployeeRegistration створює новий об'єкт tblEmployee, заповнює його значеннями з параметрів і зберігає новий запис в базі даних. Метод повертає true у разі успішної реєстрації та false у разі невдачі (рис. 3.23).

```
1 reference
public Boolean EmployeeRegistration(string FirstName, string LastName, string EmailID, st
{
    var DC = new DataClassesDataContext();
    if (CBy != "")
    {

        tblEmployee objEmp = new tblEmployee();
        objEmp.FirstName = FirstName;
        objEmp.LastName = LastName;
        objEmp.EmailID = EmailID.ToLower();
        objEmp.Password = Encryptdata(Password);
        if (Gender != 0)
        {
            objEmp.Gender = Gender;
        }

        objEmp.ContactNo = ContactNo;

        objEmp.CreatedOn = DateTime.Now;
        if (CBy != null)
        {
            objEmp.CreatedBy = Convert.ToInt32(CBy);
            objEmp.IsActive = true;
            objEmp.IsVerify = true;
            objEmp.IsVerifyByAdmin = true;
        }
        else
        {
            objEmp.IsActive = false;
            objEmp.IsVerify = false;
        }

        DC.tblEmployees.InsertOnSubmit(objEmp);
        DC.SubmitChanges();
        return true;
    }
```

Рисунок 3.23 – Метод EmployeeRegistration

Аналогічна функція, використовується для реєстрації вже адміністратора (рис. 3.24):

```
1 reference
public bool AdminRegistration(string FirstName, string LastName,
{
    var DC = new DataClassesDataContext();
    tblAdmin objAddAdmin = new tblAdmin();
    objAddAdmin.FirstName = FirstName;
    objAddAdmin.LastName = LastName;
    objAddAdmin.EmailID = EmailID;
    objAddAdmin.Password = Encryptdata(Password);
    objAddAdmin.ContactNo = ContactNo;
    objAddAdmin.Image = Image;
    objAddAdmin.IsInsert = IsInsert;
    objAddAdmin.IsUpdate = IsUpdate;
    objAddAdmin.IsDelete = IsDelete;
    objAddAdmin.IsActive = true;
    objAddAdmin.CreatedBy = CBY;
    objAddAdmin.CreatedOn = DateTime.Now;
    objAddAdmin.IsSuper = IsSuper;
    DC.tblAdmins.InsertOnSubmit(objAddAdmin);
    DC.SubmitChanges();
    return true;
}
// AdminReg.aspx
```

Рисунок 3.24 – Функція AdminRegistration

Відповдно для реєстрації клієнтів використовується функція ClientRegistration, яка теж додає запис та повертає булеву змінну (рис. 3.25).

```
1 reference
public bool ClientRegistration(string Name, string E
{
    var DC = new DataClassesDataContext();
    tblClient objAddClient = new tblClient();
    objAddClient.ClientName = Name;
    objAddClient.ContactNo = ContactNo;
    objAddClient.EmailID = EmailID;
    objAddClient.Password = Encryptdata(Password);
    if (ComName != null)
    {
        objAddClient.CompanyName = ComName;
        objAddClient.WebsiteURL = WebURL;
        objAddClient.Address = Address;
        objAddClient.Landmark = Landmark;
        objAddClient.CityID = City;
        objAddClient.Zipcode = ZipCode;
        objAddClient.CreatedBy = CBY;
        objAddClient.IsActive = true;
    }
    else
    {
        objAddClient.IsActive = false;
    }
    objAddClient.CreatedOn = DateTime.Now;
    DC.tblClients.InsertOnSubmit(objAddClient);
    DC.SubmitChanges();
    return true;
}
```

Рисунок 3.25 – Функція ClientRegistration

Для створення представлення в ASP.net використовується файл типу chtml – спеціальний формат html розмітки з використанням бібліотеки C#. Стартова сторінка Home, має відповідний клас та функціонал з власними функціями та наслідує System.Web.UI.Page (рис. 3.26):

```
2 references
public partial class Home : System.Web.UI.Page
{
    ServiceClient ViewServiceObject = new ServiceClient();
    7 references
    public static string GetMacAddress()
    {
        foreach (NetworkInterface nic in NetworkInterface.GetAllNetworkInterfaces())
        {
            // Only consider Ethernet network interfaces
            if (nic.NetworkInterfaceType == NetworkInterfaceType.Ethernet)
            {
                return nic.GetPhysicalAddress().ToString();
            }
        }
        return null;
    }
    7 references
    public void AddErrorLog(ref Exception strException, string PageName, string UserType, in
    {
        var DC = new DataClassesDataContext();
        //Insert record in ErrorLog
        tblError objError = new tblError();
        objError.PageName = PageName;
        objError.Description = strException.Message.ToString();
        objError.CreatedOn = Convert.ToDateTime(DateTime.Now);
        objError.UserType = UserType;
        if (UserID != 0)
```

Рисунок 3.26 – Клас Home

Розглянемо функції даного класу. Метод GetMacAddress призначений для отримання MAC-адреси Ethernet мережевого інтерфейсу. Він проходить по всіх мережевих інтерфейсах на машині і повертає MAC-адресу першого знайденого Ethernet інтерфейсу. Якщо таких інтерфейсів немає, метод повертає null (рис. 3.27).

```
ServiceClient ViewServiceObject = new ServiceClient();
7 references
public static string GetMacAddress()
{
    foreach (NetworkInterface nic in NetworkInterface.GetAllNetworkInterfaces())
    {
        // Only consider Ethernet network interfaces
        if (nic.NetworkInterfaceType == NetworkInterfaceType.Ethernet)
        {
            return nic.GetPhysicalAddress().ToString();
        }
    }
    return null;
}
```

Рисунок 3.27 – Метод GetMacAddress

Наступний метод просто контролює на наявність помилок при завантаженні даної сторінки, якщо такі помилки існують, то він записує в базу даних (рис. 3.28):

```
7 references
public void AddErrorLog(ref Exception strException, s
{
    var DC = new DataClassesDataContext();
    //Insert record in ErrorLog
    tblError objError = new tblError();
    objError.PageName = PageName;
    objError.Description = strException.Message.ToStr
    objError.CreatedOn = Convert.ToDateTime(DateTime.
    objError.UserType = UserType;
    if (UserID != 0)
    {
        objError.UserID = UserID;
    }
    else
    {
        objError.UserID = null;
    }
    if (AdminID != 0)
    {
        objError.AdminID = AdminID;
    }
    else
    {
        objError.AdminID = null;
    }
    if (MACAddress != null)
    {
        objError.MacAddress = MACAddress;
    }
    else
    {
        objError.MacAddress = null;
```

Рисунок 3.28 – Метод контролю помилок AddErrorLog

Наступна функція працює, при спрацюванні Кнопки Subscribe, після натискання якої користувач ввівши власний електронний адрес може отримувати повідомлення від цього сервісу (рис. 3.29):

```
0 references
protected void btnSubscribe_Click(object sender, EventArgs e)
{
    try
    {
        bool CheckEmail = ViewServiceObject.SubscribeEmailCheck(txtSubEmail.Text);
        if (CheckEmail == false)
        {
            errorSubscribe.Text = "Email already Subscribed";
            errorSubscribe.Visible = true;
            PanelUnSubscribe.Visible = true;
            PanelSubscribe.Visible = false;
        }
        else
        {
            ViewServiceObject.Subscribe(txtSubEmail.Text);
            errorSubscribe.Text = "Your Email Subscribed Successfully..";
            errorSubscribe.Visible = true;
            txtSubEmail.Text = "";
        }
    }
    catch (Exception ex)
    {
        int session = Convert.ToInt32(Session["ClientID"].ToString());
        string PageName = System.IO.Path.GetFileName(Request.Url.AbsolutePath);
        string MACAddress = GetMacAddress();
        AddErrorLog(ref ex, PageName, "Client", session, 0, MACAddress);
        ClientScript.RegisterStartupScript(GetType(), "abc", "alert('Something went wrong! Try again');", true);
    }
}
```

Рисунок 3.29 – Функція btnSubscribe_Click

Даний функціонал визначається в chtml розмітці (рис. 3.30):

```
<!-- our services -->
<div id="services" class="services-list">
    <div class="container">
        <h3 class="head">Наші послуги</h3>
        <p class="urna">Оновіть свою технічну базу, обравши з нашого широкого спектру послуг.</p
        <div class="services-gds">
            <asp:Repeater ID="rptViewOurServices" runat="server">
                <ItemTemplate>
                    <asp:LinkButton ID="LinkButton1" runat="server" OnClick="lnkService_Click">
                    <div class="col-md-4 Services-grid">
                        <center><span class="glyphicon glyphicon-collapse-down" aria-hidden="tru
                        <h2><%# Eval("CategoryName") %></h2></center>
                    </div>
                    </asp:LinkButton>
                </ItemTemplate>
            </asp:Repeater>
            <div class="clearfix"></div>
        </div>
    </div>
</div>

<!-- newsletter -->
<div class="newsletter">
    <div class="container">
        <div class="col-md-6 w3agile_newsletter_left">
            <h3>Підпишіться на розсилку новин</h3>
            <p>Підпишіться на нашу розсилку і будьте в курсі останніх подій та особливих днів. I
        </div>
        <div class="col-md-6 w3agile_newsletter_right">
            <form action="#" method="post">
                <asp:TextBox ID="txtSubEmail" runat="server" type="email" OnTextChanged="txtSubE
                <asp:Panel ID="PanelSubscribe" runat="server">
                    <asp:Button ID="btnSubscribe" runat="server" Text="Subscribe" OnClick="btnSu
                </asp:Panel>
                <asp:Panel ID="PanelUnSubscribe" runat="server" Visible="false">
```

Рисунок 3.30 – Chtml розмітка головної сторінки

Наступна сторінка «Про нас». Для завантаження використовується відповідний клас AboutUs. Даний клас забезпечує роботу даної сторінки, та відлагодження помилок, адже даний клас має ідентичний функціонал, що й попередня сторінка (рис. 3.31):

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using EmployeeAppraisalServiceReference;
using System.Net.NetworkInformation;

2 references
public partial class AboutUs : System.Web.UI.Page
{
    ServiceClient ObjectAboutUS = new ServiceClient();
    2 references
    public static string GetMacAddress()
    {
        foreach (NetworkInterface nic in NetworkInterface.GetAllNetworkInterfaces())
        {
            // Only consider Ethernet network interfaces
            if (nic.NetworkInterfaceType == NetworkInterfaceType.Ethernet)
            {
                return nic.GetPhysicalAddress().ToString();
            }
        }
        return null;
    }
    2 references
    public void AddErrorLog(ref Exception strException, string PageName, string UserT
    {
        var DC = new DataClassesDataContext();
        //Insert record in ErrorLog
        tblError objError = new tblError();
        objError.PageName = PageName;
        objError.Description = strException.Message.ToString();
        objError.CreatedOn = Convert.ToDateTime(DateTime.Now);
        objError.UserType = UserType;
        if (UserID != 0)
```

Рисунок 3.31 – Клас AboutUs

Розмітка даної веб-сторінки виглядає наступним чином (рис. 3.32):

```
<%@ Page Title="" Language="C#" MasterPageFile="~/ClientHeadMaster.master" AutoEventWireup="true" CodeFile="AboutUs.aspx.cs" Inherits="AboutUs" L
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="Server">
    <script src="bootstrap/js/bootstrap.min.js"></script>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="Server">
    <!-- banner1 -->
    <div class="banner1">
        <div class="container">
            <h1>About Us</h1>
        </div>
    </div>
    <!-- //banner1 -->
    <!-- about -->
    <div class="about">
        <div class="container">
            <div class="col-md-6 w3_about_grid">
                <h3>Короткий опис проекту <span>ВідслідкуйМоюРоботу</span></h3>
                <p>
                    <i>"ВідслідкуйМоюРоботу" - це зручний веб додаток, в якому клієнти можуть надавати різні типи вимог до проектів. Клієнти можу
                    </i>
                    Організація може отримувати вимоги під клієнта та керувати роботою над проектом онлайн. Керівник проекту призначає команду пр
Члени команди входять в систему і вказують свої завдання. Після завершення вони заповнюють звіт про виконання та статус виконання завдання онлай
Клієнт може увійти в систему і відстежувати статус проекту. Система показуватиме статус проекту в режимі реального часу, що допоможе менеджеру пр
Додаток працюватиме в Інтернеті, що дозволить географічно віддаленим офісам організації одночасно працювати над одним і тим же проектом в режимі
Зрештою, веб-сайт дасть можливість користуватися сервісом і допоможе компанії оцінити роботу співробітників за допомогою відгуків та рейтингів.
                </p>
            </div>
            <div class="col-md-6 w3_about_grid">
                <img src="Client_Temp/images/16.jpg" class="img-responsive" style="display: block; width: 100%; height: auto; margin-top: 120px;"
            </div>
            <div class="clearfix"></div>
        </div>
    </div>
    <!-- //about -->
    <!-- team -->
    <div class="team">
```

Рисунок 3.32 – Chtml розмітка сторінки «Про нас»

Наступна розмітка передбачає налаштування верхньої панелі для навігації та нижньої для контактних даних (рис. 3.33):

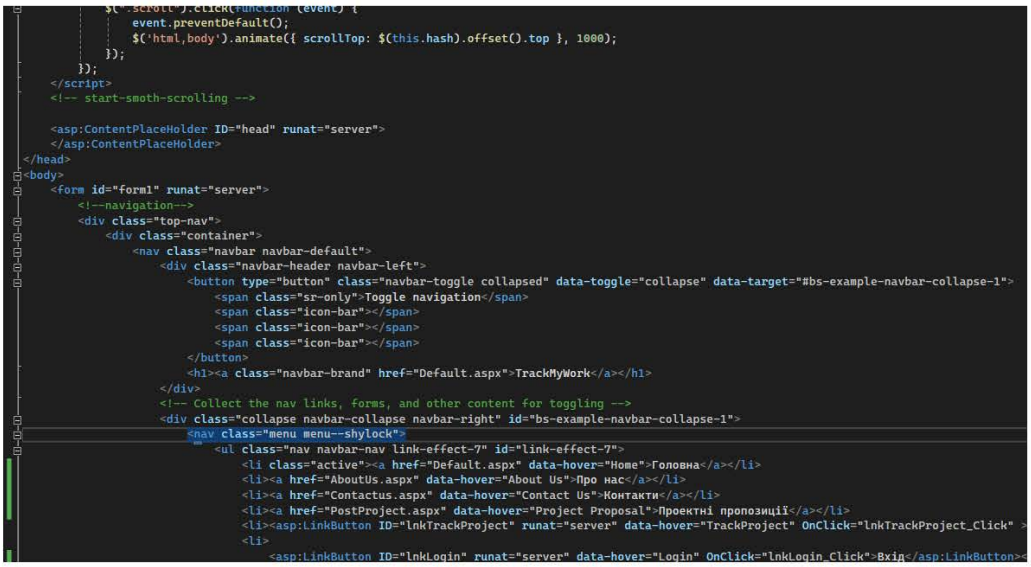


Рисунок 3.33 – Розмітка верхньої панелі навігації

Для навігації за посиланнями використовується тег: Про нас, даний тег завантажує наступну сторінку.

Для оформлення нижньої панелі використовується тег , footer – це клас який стилізований за допомогою стилів bootstrap (рис. 3.35).

```
<div class="footer">
    <div class="container">
        <div class="agileinfo_footer_grids">
            <div class="col-md-4 agileinfo_footer_grid">
                <h2 style="color: #008BFF;">TrackMyWork</h2>
                <p>
                     "TrackMyWork" is a user-friendly web application in witch clients
                    Clients can have the facility of project tracking and see the statu

                </p>
                <%--<ul class="social-icons">
                    <li><a href="#" class="icon-button twitter"><i class="icon-twitter"
                    <li><a href="#" class="icon-button google"><i class="icon-google"><
                    <li><a href="#" class="icon-button v"><i class="icon-v"></i><span><
                    <li><a href="#" class="icon-button pinterest"><i class="pinterest">
                </ul>--%>
            </div>
            <div class="col-md-4 agileinfo_footer_grid">
                <asp:Panel ID="PanelContactus" runat="server">
                    <h3>Contact Info</h3>
                    <ul class="agileinfo_footer_grid_list">
                        <li><i class="glyphicon glyphicon-map-marker" aria-hidden="true
                        <li><i class="glyphicon glyphicon-envelope" aria-hidden="true">
                        <li><i class="glyphicon glyphicon-earphone" aria-hidden="true">
                    </ul>
                </asp:Panel>
            </div>
            <div class="col-md-4 agileinfo_footer_grid">
                <h3>Navigation</h3>
                <ul class="agileinfo_footer_grid_nav">
                    <li><span class="glyphicon glyphicon-chevron-right" aria-hidden="tr
                    <li><span class="glyphicon glyphicon-chevron-right" aria-hidden="tr
                    <li><span class="glyphicon glyphicon-chevron-right" aria-hidden="tr
```

Рисунок 3.34 – Розмітка нижньої панелі сайту

3.6 Тестування веб-додатку

Для перевірки роботи функціоналу додатку потрібно перевірити роботу веб-додатку та впевнитися в завантаженні всіх сторінок.

Спочатку потрібно завантажити сторінку перейшовши на відповідну URL-адресу (рис. 3.35): http://localhost:12337/Default.aspx

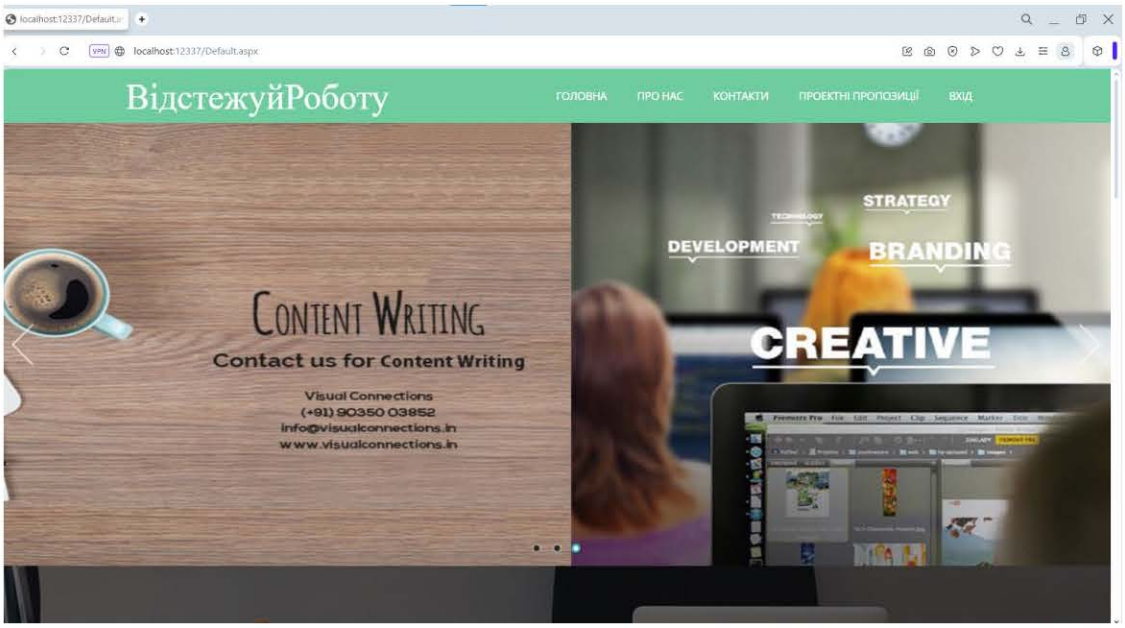


Рисунок 3.35 – Відкриття сайту.

Користувач може спостерігати слайдер з картинками які він може перемикати в обраному напрямку (рис. 3.36):

Рисунок 3.36 – Слайдер фотографій

У верхній стороні сторінки є навігаційне меню, яке дає можливість перейти на інші сторінки веб-додатку (рис. 3.37).

Рисунок 3.37 – Навігаційне меню

Опустившись в низ користувач, може побачити, над якими проектами працювала компанія та ознайомитися з послугами (рис. 3.38):

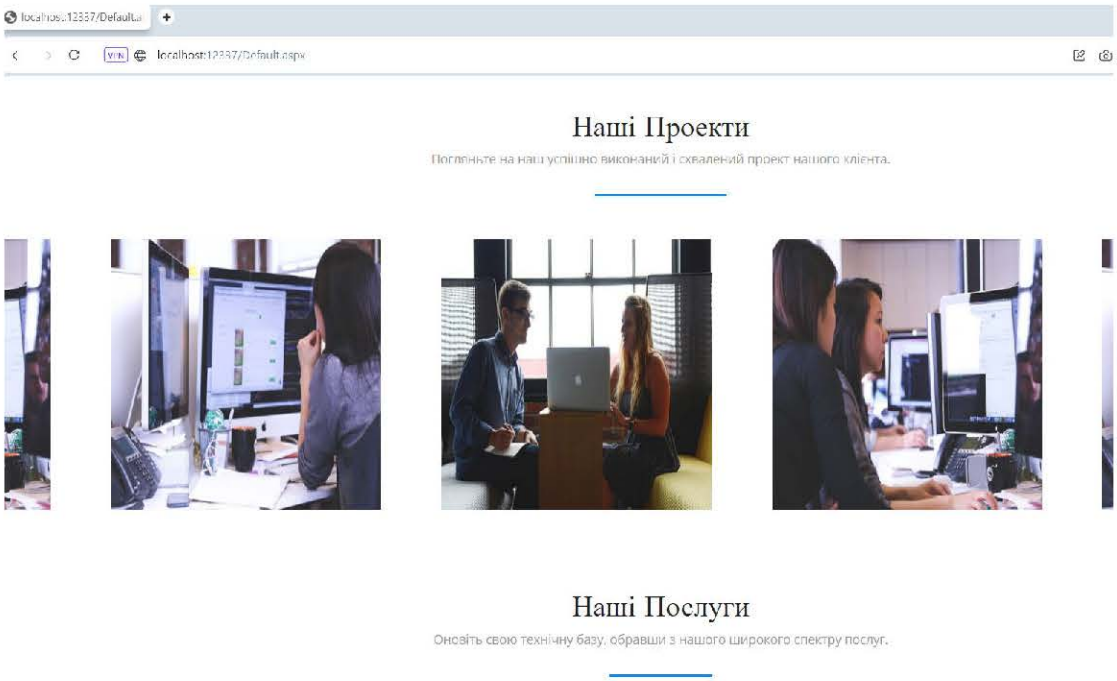


Рисунок 3.38 – Частина першої сторінки «Наші проекти»

Крім цього користувач може підписатися на розсилання новин на власту пошту (рис. 3.39):

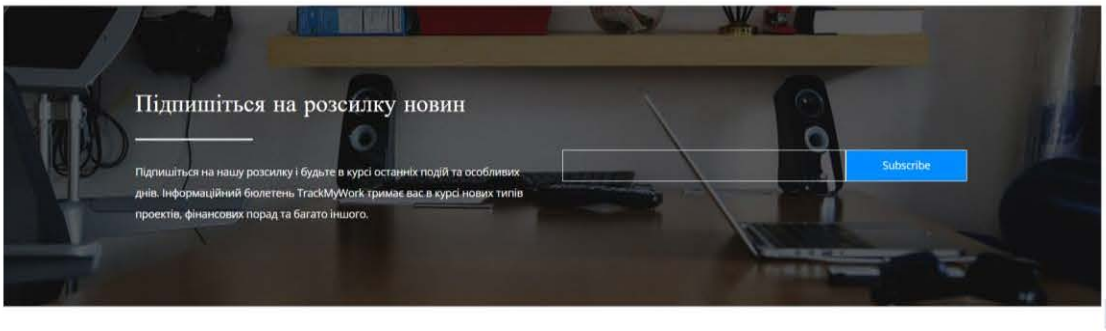


Рисунок 3.39 – Блок з розсиланням новин

В кінці сторінки користувач може отримати доступну контактну інформацію про компанію та невелике меню, з якого зручно можна перейти на наступну сторінку (рис. 3.40):

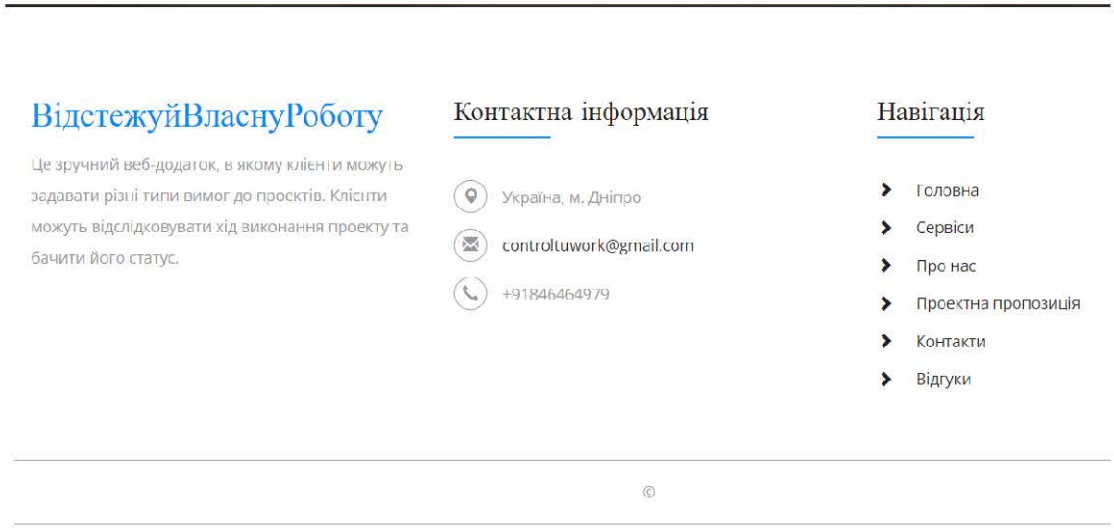


Рисунок 3.40 – Нижча частина головної сторінки

Далі користувач може перейти на наступну сторінку «Про нас», натиснувши на відповідну кнопку (рис. 3.41):

Рисунок 3.41 – Сторінка «Про нас»

На даній сторінці користувач може отримати коротку інформацію про компанію та про проект для оцінки роботи працівників. Якщо користувач матиме бажання співпрацювати з компанією, або повідомити про помилку, він може перейти на сторінку «Контакти», та ввести відповідні поля, щоб відправити компанії повідомлення (рис. 3.42).

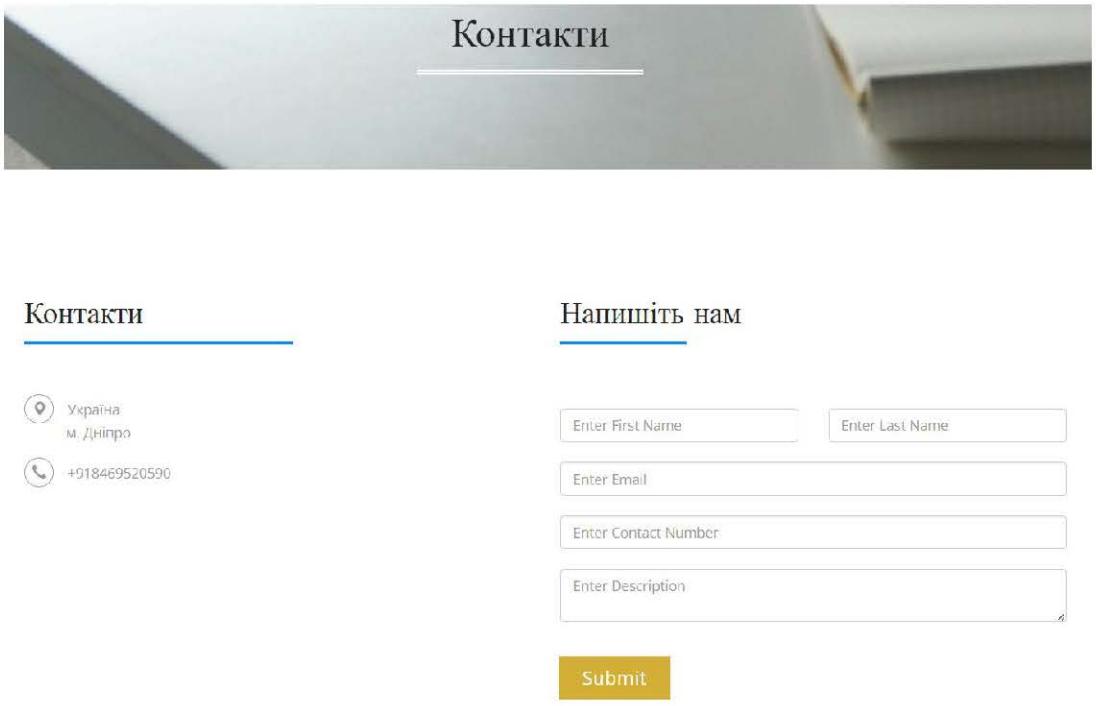


Рисунок 3.42 – Сторінка з контактною інформацією

3.7 Висновок до третього розділу

Результатом даної практичної частини є створення веб-додатку який дає можливість компанії залучати нових людей до власного проекту для оцінки роботи власних працівників. В даному проекті використано сучасні технології, щодо створення клієнт-серверного додатку, використовуючи такі технології як ASP.net та WCF, що дозволило легко масштабувати проект та розробити його таким чином, щоб він був корисним для майбутніх користувачів.

Окремо варто зауважити, що проект не позбавлений недоліків та може бути покращений за рахунок впровадження додаткових дизайнерських рішень, щодо зовнішнього вигляду додатку, так і налаштування додаткових анімацій, використовуючи її не тільки на головній сторінці проекту а й на інших сторінках також.

Загалом даний проект допоможе полегшити роботу для менеджерів в можливості управління та аналізу продуктивності персоналу, що значно краще вплине на роботу всієї компанії.

ВИСНОВОК

Кваліфікаційна робота присвячена розробці веб-застосунку для оцінки співробітників, що є актуальним завданням у контексті сучасного розвитку цифрових технологій та автоматизації бізнес-процесів. Метою роботи було створення ефективного інструменту для об'єктивного оцінювання працівників, що забезпечить керівництву компанії швидкий та точний аналіз результатів роботи персоналу.

У першому розділі роботи було проведено детальний аналіз предметної області, зокрема дослідження сучасних методів та підходів до розробки веб-додатків. Особливу увагу було приділено аналізу існуючих рішень та вибору оптимальних технологічних стеків. Встановлено, що використання сучасних фреймворків, таких як ASP.NET та WCF, є найбільш ефективним для створення масштабованих та продуктивних веб-застосунків.

У другому розділі роботи було обрано оптимальні програмні засоби для реалізації системи. Серед розглянутих технологій були ASP.NET, WCF, LINQ та інші інструменти платформи .NET. Для реалізації проекту було обрано саме ці технології завдяки їх високій продуктивності, безпеці та гнучкості. Було визначено архітектуру системи, яка включає клієнтську та серверну частини, інтеграцію з базою даних та механізми безпеки.

У третьому розділі було детально описано процес розробки веб-застосунку. Реалізовано основні компоненти системи, такі як інтерфейси користувача, логіка бізнес-процесів та інтеграція з базою даних. Було розроблено та протестовано ключові елементи системи, включаючи функції реєстрації та аутентифікації користувачів, введення та збереження даних про оцінку співробітників, генерацію звітів та аналіз результатів. Проведене тестування показало, що система ефективно обробляє дані та забезпечує високу точність оцінювання.

Розроблений веб-застосунок демонструє високу ефективність та продуктивність, а також відповідає сучасним вимогам у галузі управління

персоналом та автоматизації бізнес-процесів. Використання сучасних технологій дозволило досягти високої продуктивності та масштабованості системи, що підвищує її цінність для практичного застосування. Результати роботи можуть бути корисними для інших дослідників та розробників у сфері інформаційних технологій, які прагнуть створювати інноваційні продукти для управління персоналом.

Розробка цього веб-застосунку підкреслює важливість використання сучасних методів розробки програмного забезпечення для вирішення складних задач управління персоналом, забезпечуючи високу продуктивність і точність аналізу даних. Подальші дослідження та вдосконалення системи можуть ще більше підвищити її ефективність і точність, розширюючи її застосування у різних галузях, таких як корпоративний сектор, освіта, державне управління та інші.

В цілому, результати кваліфікаційної роботи підтверджують, що використання сучасних технологій та архітектурних рішень є перспективним напрямком у розробці веб-застосунків для оцінки співробітників і може значно покращити якість та швидкість обробки інформації, що є важливим для розвитку сучасних інформаційних технологій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Офіційний документація C# [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview

2. Основи LINQ [Електронний ресурс]. – Режим доступу: https://krypton.com.ua/rozdil-17-linq/

3. ASP.NET Core [Електронний ресурс]. – Режим доступу: https://dotnet.microsoft.com/ru-ru/apps/aspnet

4. SOA vs. microservices: Which is best for your business [Електронний ресурс]. – Режим доступу: https://www.atlassian.com/microservices/microservices-architecture/soa-vs-microservices

5. Microservice Architecture: Aligning Principles, Practices, and Culture / I. Nadareishvili, R. Mitra, M. McLarty, M. Amundsen – O'Reilly Media, 2016 – 146p.

6. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions / Gregor Hohpe, Bobby Woolf – Addison-Wesley, 2004 – 736p.

7. Develop Service-Oriented Applications with WCF [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/framework/wcf/

8. Поняття веб-сервіс. [Електронний ресурс]. – Режим доступу: https://semantica.in/blog/chto-takoe-veb-servis.html

9. Системи управління проектами. Кращі системи управління проектами: стаття [Електронний ресурс]. – Режим доступу: https://www.fewskills.com/workflow-project-app/

10. UML Class Diagram Tutorial with Examples. Guru99. [Електронний ресурс]. – Режим доступу:https://www.guru99.com/uml-class-diagram.html

11. MERN Stack (MongoDB, Express.js, React, Node.js) [Електронний ресурс]. – Режим доступу: https://appmaster.io/glossary/mern-stack-mongodb-express-js-react-node-js

12. Що таке LAMP? Переваги і недоліки [Електронний ресурс]. – Режим доступу: https://hyperhost.ua/info/uk/shcho-take-lamp-perevagi-i-nedoliki

13. ОСОБЛИВОСТІ СТВОРЕННЯ WEB-ОРІЄНТОВАНИХ СИСТЕМ З ДОПОМОГОЮ MEAN СТЕК [Електронний ресурс]. – Режим доступу: https://otakoyi.ua/osoblyvosti-stvorennya-web-system-z-dopomohoyu-mean-stek

14. Barker, R. (2014). High Performance Responsive Design: Building Faster Sites Across Devices. Apress.

15. Get started with Bootstrap [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.3/getting-started/introduction/