МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Львівський національний університет імені Івана Франка
Факультет електроніки та комп'ютерних технологій
Кафедра системного проектування

Допустити до захисту
Завідувач кафедри
доц. Шувар Роман Ярославович
«___» _____2023 р.

Кваліфікаційна робота

Бакалавр

Використання мікросервісної архітектури для розробки сервісу двосторонньої комунікації з користувачами веб порталу

Виконала:
Студентка групи ФЕП – 41
спеціальності 121 – Інженерія програмного забезпечення
Омелянюк Ольга Філімонівна

Науковий керівник:
доц. Шувар Роман Ярославович
«____» _____2023 р.

Рецензент:
Катеринчук Іван Миколайович

Львів 2023

ЗМІСТ

АНОТАЦІЯ

У сучасному світі, де інтернет є невід'ємною частиною нашого повсякденного життя, сайти стали складовою цифрового простору. Вони надають можливість людям отримувати інформацію, спілкуватися, здійснювати покупки, знаходити розваги та виконувати безліч інших дій. Мій дипломний проект, має велику актуальність в сучасному світі. Він надає можливість своєчасно та зручно отримувати необхідну інформацію для викладачів.

Окрім надання інформації, проект також сприяє покращенню комунікації та співпраці між викладачами. Він допомагає злагоджувати роботу, обмінюватися досвідом та знаннями, а також забезпечує зручні інструменти для ведення активного діалогу та вирішення завдань. Він також надає інноваційний підхід до спілкування та обміну інформацією. Завдяки цьому, викладачі можуть легко та швидко знаходити необхідну інформацію, спілкуватися з колегами, обмінюватися досвідом та вирішувати професійні завдання.

Сайт відповідає усім сучасним вимогам, що дозволяє користувачам зручно та ефективно використовувати його функціонал. У процесі розробки свого проекту я використовувала передові технології програмування та дизайну веб-сайтів. Мій проект має сучасний та естетичний дизайн, інтуїтивно зрозумілий інтерфейс та високий рівень функціональності. Я також забезпечила безпеку даних та конфіденційність користувачів, використовуючи передові методи шифрування та захисту інформації.

Посилання на Github https://github.com/olhaomelianiuk/olha_omelianiuk_FEP-41_code/blob/b0cf66f553de7598d35350ceab0478360a971b60/%D0%9E%D0%BC%D0%B5%D0%BB%D1%8F%D0%BD%D1%8E%D0%BA_%D0%9E%D0%BB%D1%8C%D0%B3%D0%B0_%D0%A4%D0%95%D0%9F-41_%D0%BA%D0%BE%D0%B4.rar

ABSTRACT

In today's world, where the Internet is an integral part of our daily lives, websites have become an integral part of the digital space. They enable people to get information, communicate, shop, find entertainment, and perform many other activities. My graduation project is of great relevance in the modern world. It provides an opportunity to receive the necessary information for teachers in a timely and convenient manner.

In addition to providing information, the project also helps to improve communication and cooperation between teachers. It helps to coordinate work, share experiences and knowledge, and provides convenient tools for active dialog and problem solving. It also provides an innovative approach to communication and information sharing. Thanks to this, teachers can easily and quickly find the necessary information, communicate with colleagues, share experiences and solve professional problems.

The site meets all modern requirements, which allows users to use its functionality conveniently and efficiently. In the process of developing my project, I used advanced programming and website design technologies. My project has a modern and aesthetic design, an intuitive interface and a high level of functionality. I have also ensured data security and user privacy by using advanced encryption and data protection methods.

Here is a link to Github https://github.com/olhaomelianiuk/olha_omelianiuk_FEP-41_code/blob/b0cf66f553de7598d35350ceab0478360a971b60/%D0%9E%D0%BC%D0%B5%D0%BB%D1%8F%D0%BD%D1%8E%D0%BA_%D0%9E%D0%BB%D1%8C%D0%B3%D0%B0_%D0%A4%D0%95%D0%9F-41_%D0%BA%D0%BE%D0%B4.rar

ВСТУП

Мета проекту: Метою проекту є полегшення спілкування та обміну інформацією між викладачами шляхом створення зручної та сучасної платформи. Основними цілями проекту є забезпечення зручного доступу до необхідної інформації для викладачів, покращення комунікації та співпраці між ними, а також надання інструментів для активного діалогу та вирішення завдань.

У сучасному світі, де інформаційні технології проникають у всі сфери нашого життя, освітній процес не залишається осторонь. Викладачі мають потребу у швидкому доступі до актуальної інформації, спілкуванні з колегами, обміні досвідом та веденні ефективного діалогу. Часто існуючі засоби комунікації та інформаційні системи не відповідають сучасним вимогам, що призводить до затримок, незручностей та недоцільного використання ресурсів.

Методи дослідження: Методи дослідження, застосовані під час розробки цього проекту, базуються на сучасних практиках та інструментах програмної розробки. Використання популярних програмних мов, таких як Java, Python або JavaScript, дозволяє створювати ефективні та функціональні веб-додатки.

Для поліпшення якості розробки та забезпечення надійності інформації, що обмінюється між викладачами, були використані методи шифрування та захисту даних. Застосування сучасних алгоритмів шифрування дозволяє забезпечити конфіденційність та цілісність інформації, а також знизити ризик несанкціонованого доступу до даних.

Для розробки веб-додатків були використані популярні фреймворки. Використання фреймворків спрощує створення модульних та масштабованих додатків, що дозволяє легко додавати новий функціонал та забезпечує зручний досвід користувачів.

Новизна проекту: Дипломний проект має новизну у контексті інноваційного підходу до спілкування та обміну інформацією між викладачами. Шляхом створення зручної платформи, яка надає своєчасну та необхідну інформацію, ви пропонуєте новий спосіб покращити комунікацію та співпрацю в освітній сфері.

Однією з ключових особливостей цього проекту є забезпечення своєчасного та необхідного доступу до інформації для викладачів. За допомогою цієї платформи викладачі можуть швидко знаходити необхідні матеріали, ресурси та оновлення, що дозволяє зекономити їхній час та зосередитися на якісному наданні освіти. Цей підхід допомагає викладачам бути в курсі останніх змін, трендів та нових методик, що впливає на покращення якості освіти.

Крім того, платформа сприяє покращенню комунікації та співпраці між викладачами. Вона надає зручні інструменти для активного діалогу, обміну досвідом та знаннями. Викладачі можуть легко обговорювати питання, розробляти нові ідеї, спільно працювати над проектами та вирішувати завдання. Це сприяє зростанню професійного розвитку викладачів, створенню колективного досвіду та підвищенню якості освітнього процесу.

Іншим аспектом новизни проекту є його потенціал для розширення та адаптації до різних освітніх сфер. Платформа може бути успішно застосована в різних освітніх установах, включаючи вищі навчальні заклади, школи та корпоративні навчальні програми.

Сфера застосування: Проект має широкі можливості застосування в освітній сфері. Викладачі вищих навчальних закладів, шкіл або корпоративних навчальних програм можуть скористатися вашою платформою для полегшення спілкування, обміну досвідом та знаннями, забезпечення взаємодії та координації роботи.

Прогнозований розвиток: Розглядаючи потенційний розвиток додатку, можна зробити кілька припущень. Можна розширити функціонал платформи, додавши нові інструменти та можливості, щоб задовольняти різноманітні потреби викладачів. Також можна розглянути можливість інтеграції з іншими освітніми платформами або системами, що дозволить розширити коло користувачів та поліпшити їх досвід використання. Партнерство з освітніми установами може допомогти впровадженню та поширенню вашого додатку.

Основна мета проекту полягає в розробці і реалізації зручної та сучасної платформи, яка надасть викладачам усі необхідні інструменти для успішної роботи та взаємодії. Платформа буде забезпечувати легкий доступ до актуальних матеріалів, навчальних програм, методичних рекомендацій та інших ресурсів, які викладачі використовують у своїй роботі. Вона також надасть можливість викладачам обмінюватися досвідом, задавати питання, спілкуватися з колегами, створювати групи для спільної роботи та вирішення завдань.

Розділ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Сайт (від англ. Website : web - «павутина, мережа» і site - «місце», буквально «місце, сегмент, частина в мережі») - сукупність електронних документів (файлів) приватної особи або організації в комп'ютерній мережі, об'єднаних під одним адресою (доменним ім'ям або IP - адресою). Іншими словами сайт - це приватний або організаційний комп'ютерний ресурс, що складається з електронних документів, об'єднаних під одним доменним ім'ям або IP-адресою.

Загалом, всесвітня павутина складається з безлічі сайтів, які утворюють мережу інформації, доступної для всіх користувачів, які можуть отримати до неї доступ через свої комп'ютери. Ця "мережа" складається з декількох сегментів, які об'єднані в єдине ціле, надаючи можливість для комунікації та доступу до баз даних з різних кутків нашої планети. HTTP - це протокол, який був спеціально розроблений для того, щоб забезпечити прямий доступ клієнтів до сайтів на серверах. Це зробило Інтернет більш доступним та зручним для використання в різних галузях, від розваг до науки та бізнесу.

В 1990 році з'явився перший у світі сайт info.cern.ch, створений Тімом Бернерс-Лі. Цей сайт був призначений для публікації нової технології World Wide Web, яка була заснована на протоколі передачі даних HTTP, системі адресації URI і мові гіпертекстової розмітки HTML. На цьому сайті було опубліковано опис принципів установки і роботи серверів і браузерів. Також розміщено список посилань на інші сайти, що зробило цей сайт першим в світі інтернет-каталогом.

Для створення першого веб-сайту Тім Бернерс-Лі ще в кінці 1990 року розробив всі необхідні інструменти, зокрема перший гіпертекстовий браузер WorldWideWeb з функціоналом веб-редактора та перший сервер на базі NeXTcube. Таким чином веб-сторінки були запроваджені на початку 1991 року.

Для розробки веб-сайтів використовуються сторінки, які складаються

з текстових файлів, що містять мову HTML. Ці файли завантажуються відвідувачем на його комп'ютер. Завантажені файли обробляються браузером, після чого виводяться на засіб відображення користувача, такий як монітор, екран КПК, принтер або синтезатор мови. Мова HTML дозволяє форматувати текст та розрізняти в ньому функціональні елементи. Зокрема, вона дозволяє створювати гіпертекстові посилання (гіперпосилання), вставляти в сторінку зображення, звукозаписи та інші мультимедійні елементи. Ці елементи роблять сайт більш привабливим та корисним для відвідувачів, а також надають можливість посилання на різні ресурси в Інтернеті та звернення до відповідних даних.

Сторінки сайтів можна додатково форматувати, додаючи стилі на мові CSS. Це дозволяє централізувати всі елементи форматування в одному файлі. Зокрема, можна визначити розмір та колір заголовків 2-го рівня, розмір та вид блоку вставки тощо. Можна також використовувати сценарії на мові JavaScript, щоб додавати події або дії на сторінці.

Створення сторінок сайтів може здійснюватися у двох форматах: як простий статичний набір файлів або за допомогою спеціальної комп'ютерної програми на сервері. Остання може бути розроблена на замовлення для окремого сайту або мати готовий продукт, призначений для певного класу сайтів. Для більшої гнучкості власника сайту, деякі програми забезпечують можливість настроювати структуру та відображення інформації на веб-сайті. Такі програми називаються системами управління вмістом, або CMS. Вони роблять розробку та управління сайтом більш зручним та простим.

У Інтернеті є різні типи сайтів. Деякі з них називають «інтернет-представництвом», які належать людині чи організації. Інші сайти є повнофункціональними порталами, на яких може бути розміщена сторінка-візитка. Це може бути коментар до посилання на сайт. Також є можливість мати свою сторінку в Інтернеті, що може бути або цілим сайтом, або особистою сторінкою, що входить до складу іншого сайту або порталу. Крім

звичайних сайтів та порталів, в Інтернеті також існують WAP-сайти, які створені для мобільних телефонів та інших мобільних пристроїв.

Початково сайти були простими статичними документами, таким як сайти-візитки. Однак з розвитком технологій кількість внутрішніх та зовнішніх посилань збільшувалася, що дозволяло сайтам стати не тільки довідником, але й функціональним офісом, новинним або медійним центром. У сучасному світі більшість сайтів є динамічними та інтерактивними. В таких випадках фахівці використовують термін «веб-додаток» - готовий програмний комплекс для вирішення завдань сайту.

Беб-додаток – невід'ємний елемент сайту, але він не має ніякої значущості без даних сайту. Щоб веб-додаток був корисним, потрібно заповнити його відповідним контентом та активувати. Зараз реклама сайтів (просування) стала однією з головних індустрій в Інтернеті.

Зазвичай для ідентифікації сайту в Інтернеті використовується лише одне доменне ім'я, за яким його знають в глобальній мережі. Звісно, є інші варіанти, - один сайт може бути доступний за кількома доменними іменами або кілька сайтів можуть бути підпорядковані єдиному домену. Великі сайти використовують кілька доменів, щоб логічно згрупувати різні види послуг, наприклад mail.google.com, news.google.com, maps.google.com. Інші випадки використання окремих доменів для різних країн або мов також не рідкість. Наприклад, google.ru та google.fr - це один і той же сайт Google на різних мовах, але технічно це різні сайти. Безкоштовні хостинги, як правило, збирають кілька сайтів під одним доменом.

Веб-сервери - це апаратні засоби, які призначені для зберігання сайтів. Послуга зберігання сайтів на веб-сервері отримала назву "хостинг". Раніше кожен сайт мав свій власний фізичний сервер, однак з розвитком технологій та зростанням обсягу Інтернет-ресурсів, на одному комп'ютері стало можливо розмістити безліч сайтів, використовуючи віртуальний хостинг. Сьогодні сервери, на яких розміщується тільки один сайт, називаються виділеними (англ. dedicated).

Веб-сайт може бути доступний за різними адресами та зберігатися на різних серверах. Якщо створена копія оригінального сайту, то її можна назвати дзеркалом. Також існує концепція офлайнової версії сайту - це копія, яку можна переглянути на будь-якому комп'ютері без підключення до Інтернету та без використання серверного програмного забезпечення (ПО).

По завершенні розробки сайту важливо провести тестування та налагодження саме офлайнової версії, щоб уникнути демонстрації помилок співробітникам та користувачам. У цілях прискорення процесу тестування та налагодження великих проектів, досвідчені тестери запрошуються в корпоративну мережу або на захищений за допомогою пароля захист. Це допомагає покращити проект та підготувати його для публіки.

Отже, копії сайту та їх використання є важливим кроком у процесі розробки та тестування в Інтернеті. Налагодження проектів у безпечному середовищі допомагає підготувати сайт для оптимальної роботи та забезпечити максимально можливий рівень користувацького досвіду.

Адміністратори в інтернет-середовищі також відомі як "адміни", грають важливу роль у розробці та обслуговуванні сайту або порталу. Якщо проект займається розробкою форми або дизайну, завдання виконує експерт або висококваліфікована група фахівців, такі як програмісти, веб-дизайнери та системні адміністратори (інакше відомі як "сисадміни"). Однак, обслуговування та наповнення сайту інформацією повинно орієнтуватися на виконання стратегічних завдань та потребує неперервної участі команди під управлінням адміністратора проекту. Зараз існує багато програм на PHP, але це підвищило вимоги до кваліфікації учасників проекту з урахуванням багатопрофільності завдань, що розв'язуються.

1.1 Основні функції сайту

Сайт надає доступ до наступних розділів: графік роботи, склад методичного об'єднання, плани засідань, звіти про працю викладачів та об'єднання в цілому, історія та результати проведення обласного туру

Всеукраїнської олімпіади з інформатики та комп'ютерної техніки, а також фотографії з олімпіади. Більш того, на сайті є можливість додавати новини. Цю функцію може використовувати лише адміністратор, який має доступ до адмін-панелі, до якої потрібно перейти за відповідним лінком у рядку адреси. Відвідувачі сайту можуть задавати будь-які запитання через форму зворотного зв'язку, в якій слід вказати контактні дані та питання. Адміністратор зможе дати відповідь на запитання у найближчий час.

1.2 Класифікація сайтів

Веб-ресурси можуть бути виділені за такими характеристиками, як схема подання інформації, її обсяг та категорія вирішуваних завдань. Серед них можна виділити наступні типи:

1. Інтернет-портали - це багатокомпонентні структури, що складаються з функціонально самодостатніх сайтів, належать самостійним організаціям або підрозділам корпоративних структур.

2. Інформаційні ресурси:

- Тематичні сайти - це сайти, які надають специфічні вузькотематичні інформацію з різних тем.

- Тематичні портали - це великі веб-ресурси, що забезпечують вичерпну інформацію з певної тематики. Портали схожі на тематичні сайти, але також містять засоби взаємодії з користувачами і дають можливість спілкуватися на сайті у форматі форумів та чатів. Це середовище існування користувача, що дає можливість для продуктивних обмінів та співпраці.

3. Інтернет-представництва власників бізнесу також можуть бути відрізнені за мірою зв'язку з Інтернетом та призначенням. Серед них можна виділити такі типи:

1. Сайт-візитка - містить загальні дані про власників сайту - фізичних або юридичних осіб. Основна інформація про діяльність, історію, ціни, контактні дані, реквізити та інше. Спеціалісти також можуть розміщувати свої резюме на таких сайтах.

2. Корпоративний сайт - містить повну інформацію про компанію та її послуги / продукцію. Також включає події в житті компанії та як підвид - навчальний сайт.

3. Каталог продукції - містить детальний опис товарів / послуг, включаючи сертифікати, технічні та споживчі характеристики, відгуки експертів і т.д. Такі сайти надають інформацію про товари / послуги детальніше, ніж це можливо на прайс-листах.

4. Інтернет-магазин - це каталог продуктів у вигляді сайту, який дозволяє клієнтам замовляти потрібні їм товари. Залежно від сайту, може бути використана різна система оплати: від оплати при отриманні товару до автоматичного відправлення рахунку по факсу або оплата за допомогою пластикових карт.

5. Промо-сайт - це сайт, присвячений конкретній торговій марці або продукту, на якому знаходиться повна інформація про бренд, різні рекламні акції (конкурси, вікторини, ігри тощо).

6. Веб-сервіс - це сайт, розроблений для виконання різноманітних завдань або надання послуг в межах мережі Інтернет.

7. Дошка оголошень дозволяє публікувати оголошення про продаж або купівлю товарів та послуг, а також інші короткі повідомлення.

8. На каталозі сайтів розміщуються блоги і веб-сайти, які можуть бути платними або безкоштовними. Розміщення свого ресурсу на каталозі сайтів може допомогти просуванню в Інтернеті.

9. Пошукові сервіси, такі як Yahoo!, Google, Bing і Яндекс, дозволяють знайти інформацію в мережі шляхом введення пошукового запиту.

10. Сервіси електронної пошти, такі як Mail.ru і Gmail, надають можливість обмінюватися електронними листами та відправляти файли.

11. Веб-форуми та блоги створюють платформу для обміну інформацією між користувачами в мережі.

12. Файлообмінний пірінговий сервіс, наприклад, Bittorrent, дозволяє обмінюватися файлами між користувачами в мережі без потреби в централізованому сервері.

13. Хмарне сховище даних, наприклад, Skydrive, забезпечує можливість

зберігати та обмінюватися файлами в хмарному сервісі.

14. Фото- та відеохостинги, наприклад, Picnik, ImageShack, Panoramio, Photobucket, YouTube та Dailymotion, дають можливість користувачам зберігати і ділитися своїми фото- та відеоматеріалами в мережі Інтернет.

1.3 Програмна розробка

Для створення сайту, я обрала програму Pycharm, оскільки вона має більш розширені можливості та полегшує написання коду. Зазвичай, для розробки сайтів використовується мова гіпертекстової розмітки HTML та формальна мова описання зовнішнього виду документа CSS. Цей стандартний набір забезпечує можливість створювати сторінки сайту з яскравим та привабливим дизайном, що включає шрифти, зображення та унікальний фон сайту. CSS дозволяє створювати різні комбінації з оздобленням сторінок сайту, щоб кожен сайт мав свій власний стиль та вигляд.

1.4 Створення сайтів

Створення сайту є складним процесом, який потребує участі різних фахівців та називається веб-розробкою. Люди, які бажають створити свій власний сайт, можуть зробити це самостійно або звернутися до спеціалізованих розробників, таких як фрілансери, студії, бюро тощо. Взаємини між замовником та виконавцем можуть бути регульовані договорами, технічними завданнями, спеціальними системами, або усною домовленістю. Замовлений проект може включати в себе створення сайту з нуля, аж до вигадування назви та реєстрації домена, а також доповнення та редизайн уже існуючого сайту. Розробка та супровід сайту стає все більш важливим для підприємств та організацій. «Тестери» кінцевого продукту відіграють особливу роль в процесі розробки, так як стадія динамічного великого проекту ніколи не припиняється. Вони забезпечують якість та ефективність проекту, допомагаючи в розробці та підвищенні його функціональності.

Для успішного виконання серйозних проектів не можна обійтися без

скриптових мов програмування. Ми обрали PHP для свого проєкту, оскільки ця мова є відносно поширеною і підтримується на більшості хостингів у всьому світі. Використовуючи скрипти на PHP, можна з легкістю додавати різноманітні новини на сайт, створювати адмін-панель для їх управління та зв'язувати їх з базою даних MySQL, що значно полегшує процес роботи з різними типами даних. Ще одним плюсом PHP скриптів є можливість створювати як користувацькі, так і адміністративні модулі для сайту, які будуть доступні лише для розробників або адміністраторів сайту.

Отже, ми вирішили, що використання найкращих можливих інструментів є ключовим аспектом при створенні веб-сайту, такого як сайт методичного об'єднання викладачів інформатики.

Основна мова програмування, яку використовують для розробки, це HTML - стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються з використанням HTML (англ. HyperText Markup Language — мова розмітки гіпертекстових документів) або XHTML. Браузер може зчитати код HTML і зобразити сторінку у доступному для обізнаної людини стилі. Крім того, із використанням HTML можна поєднувати каскадні таблиці стилів та вбудовані скрипти - це дозволить зробити сайт ще більш привабливим та інтерактивним для користувачів.

Створення структурованого документу шляхом позначення структурного складу тексту - одна з важливих функцій HTML. Користувач може використовувати засоби тегування, щоб створити заголовки, абзаци, списки, таблиці, цитати та інші елементи, що роблять документ більш читабельним і зрозумілим.

Ще однією важливою функцією HTML є можливість отримання інформації з Всесвітньої мережі з допомогою гіперпосилань. Користувач може вставити гіперпосилання на інші веб-сторінки, зображення або медіа-елементи, що робить документи інтерактивними та більш інформативними.

PHP є скриптовою мовою, що використовується для генерації HTML-сторінок на стороні веб-сервера. Вона є однією з найпоширеніших мов веб-розробки поруч з Java, NET, Perl, Python та Ruby. PHP підтримується більшістю хостинг-провайдерів, і він є проектом відкритого програмного забезпечення.

Іншою перевагою PHP є те, що він інтерпретується веб-сервером в HTML-код. Браузер ніколи не бачить PHP-коду, оскільки він відправляється як готовий html-код. Це відрізняється від скриптових мов, таких як JavaScript, які мають свій код, який відправляється на виконання в браузері користувача.

PHP - це мова програмування, що забезпечує можливість вбудовувати її безпосередньо в HTML-код сторінок. Цей механізм дозволяє розширити функціональність сайту, не перезавантажуючи сторінки. Хоча є перевагою для безпеки, використання PHP може погіршити інтерактивність сторінок. Але програмісти можуть використовувати PHP разом з JavaScript-кодами, що виконуються вже на стороні клієнта, для покращення інтерактивності сайту.

Основна перевага PHP полягає в тому, що вона може бути вбудована безпосередньо в HTML-код сторінок. PHP-інтерпретатор оброблює код, розміщений між символами <? та ?>. Функції PHP забезпечують велику різноманітність функцій, що дозволяє уникнути написання складних функцій на C або Pascal. PHP була розроблена з урахуванням зручності програмістів, тому ця мова здаватиметься знайомою програмістам, що працюють в різних областях. Багато конструкцій PHP запозичені з C та Perl. Хоча PHP є досить молодою мовою програмування, її популярність серед web-програмістів швидко зростає. Зараз PHP є однією з найпопулярніших мов для створення веб-додатків, що забезпечується її вільною ліцензією та доступністю для використання.

CSS - це спеціальна мова, що використовується для відображення веб-сторінок, написаних мовами розмітки даних. Зазвичай CSS використовують для кращої візуальної презентації HTML та XHTML сторінок, але він також може бути застосований до інших видів XML-документів. CSS розроблений як засіб для розділення змісту та оформлення веб-сторінок. Це означає, що в HTML-коді можна вказати, які елементи потребують певного оформлення, а в CSS описати, як саме це оформлення повинне виглядати. CSS дозволяє розміщувати властивості стилів з централізованої точки і використовувати їх для всієї веб-сторінки, що дозволяє надати одинакового вигляду всім елементам сторінки. Із загальним застосуванням CSS, стиль всього вмісту сторінки може легко змінюватися, забезпечуючи зручність управління і підтримкою веб-сайту.

Apache HTTP-сервер - це відкритий веб-сервер, доступний на UNIX-

подібних, Microsoft Windows, Novell NetWare та інших операційних системах. Розробка та підтримка Apache здійснюється спільнотою розробників відкритого програмного забезпечення, керованої Apache Software Foundation. З 1996 року Apache є найпопулярнішим веб-сервером у світі, обійшовши NCSA HTTPd. Цей продукт не має комерційної мети і є вільно розповсюджуваним. Він має безліч можливостей, багато з яких здійснюються за допомогою скомпільованих модулів, що розширюють базові функціональні можливості веб-сервера. Apache розширюється за допомогою прикладних інтерфейсів (API), для підтримки мов програмування Perl, Python, Tcl і PHP.

Apache має декілька методів стиснення, одним з яких є модуль mod_gzip, який допомагає скоротити розмір веб-сторінок, переданих через HTTP. Віртуальний хостінг дає можливість використовувати одну установку Apache для кількох веб-сайтів, таких як www.example.com, www.test.com, server.test.com і т. д.

Apache найбільш широко використовується для передачі через HTTP статичних та динамічних веб-сторінок в Інтернеті. Багато веб-застосунків були створені з урахуванням можливостей, які надає цей веб-сервер. Apache також може працювати як кеш-проксі-сервер, що дозволяє значно підвищити продуктивність при роботі з документами, розташованими в Інтернеті для користувачів локальної мережі. Налаштування проксі-сервера включають типи файлів для кешування, максимальний обсяг дискового простору для кешу тощо. Також можливе періодичне переглядання та індексування бази даних кеша з метою вивільнення дискового простору під час видалення застарілих об'єктів.

Створення сайтів у візуальних редакторах

HTML-редактор - це комп'ютерна програма, що дозволяє створювати HTML-сторінки або змінювати наявні. Проте, хоча HTML-код можна написати в простому текстовому редакторі, використання спеціальних редакторів для HTML-коду має безліч переваг. Вони пропонують більше зручностей і функціональних можливостей, щоб зробити процес більш швидким та простим. Серед найпопулярніших HTML-редакторів можна виділити Microsoft FrontPage та Dreamweaver.

Microsoft FrontPage - це інструмент для створення веб-сторінок і сайтів, який не вимагає від користувача знання мови HTML. Завдяки цьому спрощенню використання, програма дозволяє створювати та розміщувати сайти на веб-сервері з подальшою експлуатацією. Інструмент надає два варіанти для створення окремої сторінки або сайту. Для цього можна скористатись шаблоном з колекції FrontPage або створити щось з нуля. Wide choice of available templates для окремих сторінок забезпечує широку свободу вибору дизайну вашого сайту. Por otra parte, кількість шаблонів в програмі FrontPage перевищує аналогічний показник в колекції програми Word. Для зручності користувача, програма пропонує декілька режимів, таких як режим папки, навігація, звичайний, код та перегляд. Незважаючи на наявність кількох версій програми, Microsoft FrontPage залишається одним з найпопулярніших редакторів веб-сторінок в світі.

Adobe Dreamweaver, колишня назва якої - Macromedia Dreamweaver, є HTML-редактором. Найновіша версія HTML-редактора Adobe DreamWeaver CS 5, яка відноситься до категорії WYSIWYG-редакторів, має безліч переваг. Його інтерфейс є зручним, редагування функцій настраюється, його підтримка великих проектів і ShockWave-технологій, можливість закачування файлів через FTP, підтримка SSI і багато іншого. Робота в цій програмі не потребує глибокого знання HTML, оскільки програма використовує технологію WYSIWYG. Проте, DreamWeaver перевершує редактори, що використовують технологію WYSIWYG, завдяки здатності генерувати чистий HTML-код. Завдяки DreamWeaver можна уникнути рутинної роботи при створенні сторінок.

1.5 Хостинг для сайту

Зазвичай сайти створюють з метою публікації в інтернеті з метою виконання певних задач. Після завершення розробки сайт завантажують на хостинг.

Хостинг надає дисковий простір на сервері для зберігання фізичної інформації, яка постійно доступна в мережі, наприклад, в Інтернеті. Зазвичай хостинг включає послугу розміщення файлів сайту на сервері, необхідне ПЗ для обробки запитів до цих файлів та веб-сервер. У відповідні послуги також може входити місце для поштової кореспонденції, баз даних, DNS файлової сховища тощо. Зазвичай вони надаються разом, але можуть запропонуватися і окремо. Хостинги можна поділити на безкоштовні та платні. Безкоштовні хостинги зазвичай мають рекламу на своїх сайтах, яке є основним джерелом доходів компанії-хостера.

Розділ 2. ПОСТАНОВКА ЗАВДАННЯ

2.1. Основна будова програми

Корпоративний портал дозволяє зберігати всю необхідну інформацію на одному ресурсі, що значно прискорює пошук. Знижується ризик витоку конфіденційної інформації. Нові працівники легше і швидше адаптуються до робочого процесу завдяки документації, яка є на порталі.

Співробітники можуть працювати над документами одночасно, спільно складати документи і пропонувати внесення змін, що дозволяє покращити якість та зменшити тривалість процесу погодження.

За рахунок зростання горизонтальних зв'язків в колективі, піднімається корпоративна культура, а результати роботи стають прозорими для всього колективу. Оглядовість та можливість накопичення досвіду дозволяють збільшити якість та швидкість вирішення типових завдань.

Корпоративний портал дозволяє контролювати внутрішнє листування, зайнятість та присутність співробітників, що дозволяє керувати робочим процесом більш ефективно. Покращується якість обслуговування клієнтів та зворотний зв'язок зі співробітниками, що безпосередньо призводить до підвищення ефективності роботи всієї компанії [6].

Завідувачам відділів та вищим менеджерам буде надана можливість отримати повідомлення від співробітників у відповідь на будь-яке питання. Будуть використовуватися інструменти бізнес-аналітики, які зможуть зібрати, попередньо обробити і проілюструвати комерційні показники, фінансовий стан, потребу кадрів, виконання планів та інше.

Також IT-фахівці матимуть можливість:

- показувати зображення з камер спостереження;
- надавати єдиний доступ до всіх систем компанії;
- використовувати адресну книгу з можливістю швидкого пошуку за ПІБ, посадою та телефоном;
- аналізувати статистику звернень до технічної підтримки;
- використовувати принцип "одного вікна" для зручного оброблення

запитів, інтеграцію з системами техпідтримки, статистичну обробку результатів та повідомлення користувачів про зміну статусу їх запитів.

Менеджерам з комунікацій:

1. Інформувати співробітників про заходи, в яких бере участь компанія, про нові послуги та продукти, що допомагає зорієнтувати всіх на кінцевий результат.

2. Збирати та оброблювати думки співробітників про умови, кадрові перестановки та інші аспекти, оцінюючи настрої всередині компанії.

3. Проводити внутрішні маркетингові кампанії, зокрема опитування працівників про нові товари та послуги, які будуть випущені на ринок.

4. Збирати інформацію про захоплення співробітників, їх мотивацію та внутрішні цінності.

Важливо зупинятися на цих обов'язках, оскільки вони можуть сильно впливати на ефективність та задоволення працівників.

HR-фахівцям:

1. Проведення опитувань та анкетувань співробітників та автоматичну розсилку до обраної групи користувачів.

2. Створення архіву організаційно-правових документів, включаючи посадові інструкції, положення про відділи та організаційні діаграми.

3. Інформування та адаптація нових співробітників, зокрема знайомство з компанією, зразками заяв та інформуванням про контактні особи.

4. Відстеження динаміки чисельності персоналу та візуалізація даної інформації у розрізі професії, віку та відділу.

5. Впровадження інтранет-порталу для підвищення продуктивності праці та зменшення паперової роботи. Це особливо корисно для великих компаній, де кілька відсотків економії часу може значно знизити витрати.

Важливо зазначити, що корпоративні портали та Інтернет - це лише інструменти [7]. Щоб вони були корисними, необхідно вміло ними користуватися шляхом їх впровадження, налаштування, відстеження їх ефективності та допрацювання. Застосування системного підходу та

автоматизації бізнесу допоможуть ефективно повернути здійснені інвестиції.

2.2.Попередній опис проведеної роботи

Логування часу, є основним функціоналом для контролю робочого процесу. Після додавання користувача до певної активності, в репорті, буде можливість логування часу на нього (таб. 2.2).

Timelog	View Timelog. Log time on various projects, pre-sales, departments and other activities.	All users
	View/Manage Timelog requests View Reports View User report View General Report Log time for user	PM, AM

Таблиця 2.2 – Можливості використання timelog відносно ролей

Для логування будуть доступні 3 типи логованих годин (звичайні, понаднормові, години очікування). При логуванні часу, потрібно основну увагу приділити понаднормовим годинам. Для цього використовуватимуться квитки підтвердження. При логуванні понаднормових годин, для одної людини, яка є адміністратором, буде створюватись квиток підтвердження, і залоговані години не будуть враховуватись, до моменту підтвердження (рис. 2.4). Для звичайних користувачів, логування буде доступне за сьогоднішній та вчорашній дні, для АМ-ів дозволяється логування в будь який час а для РМ-ів, лише за теперішній місяць. Так як репорт корстувачу приходитиме за ввесь місяць, потрібно врахувати можливі хибні варіанти логування часу:

- логування людиною, яка вже не знаходиться в команді;
- логування на людину, якой в те число, ще не було в команді, абобула вже видалена з неї;
- дана активність, ще не почалась або вже завершилась.

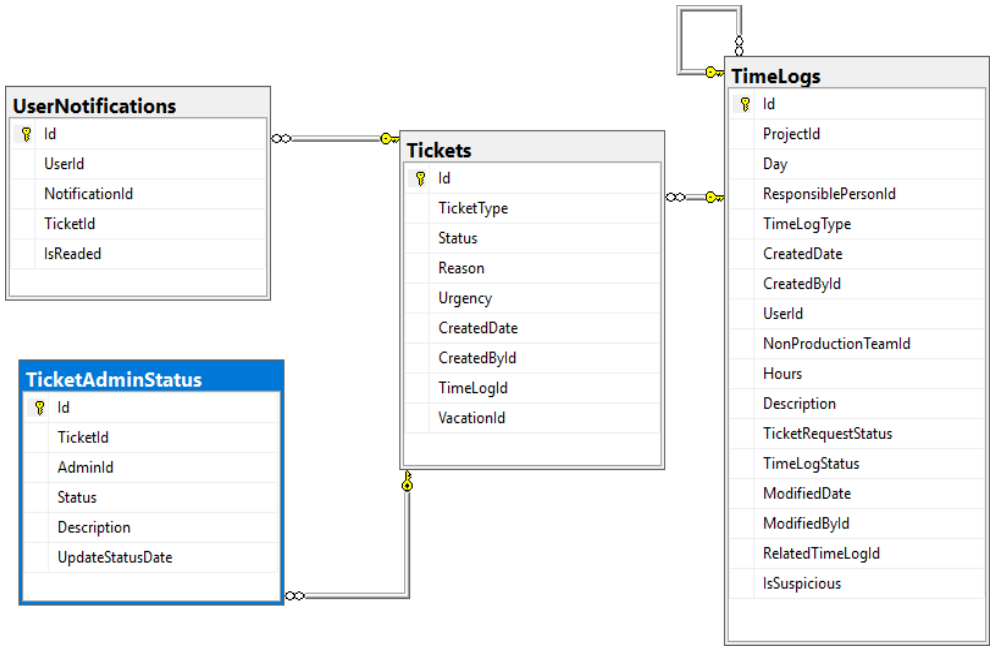

Рисунок 2.4 – Схема основних зв’язків з timelog

Для логування відпрацьованих годин, репорти будуть розділені на два типи. Перший репорт, буде по користувачу, а другий, по проектах. Для розробки вихідної моделі, спочатку, потрібно вирішити, якою формою його планується відображати. Одже, для того щоб звичайні користувачі могли логувати години, PM-и могли керувати проектом та внутрішньою роботою команди, а бухгалтери вели звітність, планується розподіл на 5 типів репортів. Орієнтуюсь розділити їх на 5 вкладок,

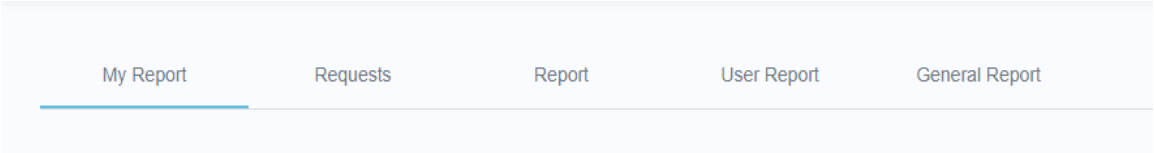

Рисунок 2.5 – Приклад поділу функціоналудля кожної, надавати доступ, в залежності від того, чи є в користувача потрібні клейми (рис. 2.5).

Для доступу до всіх вкладок, небхідною буде клейма «Timelog management». Для доступу до “Requests”, також, необхідно мати клейму “Tickets management”, а для “General Report” необхідно мати “Company Staff Report”. Для репроту по всіх проектах, необхідною буде “Company Staff Report”.

Логування часу буде доступне для наступних категорій:

- Projects: Технічні персонал. Люди, які беруть участь у проектах: PM, QA, Developers, Designers тощо;
- Pre-Sales: Технічний персонал. Люди, які беруть безпосередню участь у передпродажній діяльності: PM, дизайнер та розробник;
- Departments: Оперативний персонал. Люди, які в основному виконують непроектну діяльність: HR, Рекрутери, AM, Engagers, Marketers тощо. Технічний персонал також може зареєструвати часу відділах, якщо вони задіяні у заходах, пов'язаних із їх відділом;
- Other: Технічний та експлуатаційний персонал. Усі працівники компанії, можуть зареєструвати час для цієї категорії часових журналів.

Для ексорту в Excel документ всіх репортів, буде дотримано наступні вимоги:

- застосування формул;
- шрифт Verdana;
- колір для вихідних, повинен бути “#EDEDED”;
- колір для оплачуваних відпусток, повинен бути “#D3D4EB”, а для неоплачуваних, повинен бути “# E7DDEC”;
- колір для свята, повинен “ бути# F9F5E2”;
- нульові години, зареєстровані - нічого не відображається, не 0;
- якщо під час відпустки, були залоговані години - відображати в клітинку лише ці години;
- усього (по горизонталі) - сума відображення стовпця (план включений); понаднормові години, стосуються лише конкретного проекту;
- в стовпці відпусток, відображати лише оплачувані відпустки та години. Якщо користувач, має неоплачувані відпустки, не потрібнозмінювати стовпчик плану;

- назва файлу має бути: "Тип_Репорту _ Місяць";
- назва аркуша повинна бути Місяць;
- excel має бути адаптований до аркушів Google.

2.3. Розробка архітектури

Для розробки програмного додатка було обрано класичну трирівневу архітектуру. Існує багато різних типів та видів архітектур, але однією з найбільш популярних є класична трирівнева система, яка включає в себе три рівні.

Важливо відмітити, що для багаторівневих архітектур часто використовуються два терміни - n-layer та n-tier. Обидві ці позначення означають "рівень", а іноді "шар". Однак, рівні в кожному з цих випадків мають відмінну суттєву різницю [1]. Tier описує фізичний рівень, а отже, наприклад, для трирівневої архітектури n-tier застосунку можуть бути такі рівні: сервер бази даних, веб-додаток на веб-сервері та браузер користувача. Тож, кожен окремий рівень представляє незалежний фізичний процес, навіть якщо всі компоненти системи знаходяться на одному комп'ютері. Якщо в якості клієнта використовується мобільний додаток, то це також може бути рівнем фізичної архітектури.

У програмуванні термін "Layer" використовується для позначення логічного рівня системи. У додатку можна виділити рівень доступу до даних, бізнес-логіки, уявлення, сервісів та інші. Варто зауважити, що логічні рівні не завжди збігаються з фізичними. Наприклад, у додатку ASP.NET зовнішній рівень містить контролери для обробки введення та уявлення, яке відображається в веб-браузері. Це означає, що зовнішній рівень розділяється на два фізичні рівні.

У даному випадку, ми розглянемо логічні рівні, тобто n-layer архітектуру [2].

Класична трирівнева система має три рівні: Presentation Layer (рівень представлення), Business Layer (рівень бізнес-логіки) та Data Access Layer (рівень доступу до даних), що відповідають за відображення інформації користувачам,

обробку даних та доступ до них.

Presentation Layer - це рівень, із яким користувач безпосередньо взаємодіє. Він містить компоненти для інтерфейсу користувача, а також механізми отримання введення від користувача. Наприклад, в додатку ASP.NET MVC на цьому рівні розташовані уявлення, контролери та об'єкти контексту запиту.

Business Layer - це рівень бізнес-логіки і включає набір компонентів, які відповідають за обробку даних, отриманих з Presentation Layer. Тут реалізується всю необхідна логіка додатка, обчислення та взаємодія з базою даних. Результат обробки даних передається назад до Presentation Layer.

Data Access Layer - це рівень доступу до даних, в якому зберігаються моделі, що описують використовувані дані. Також тут розміщуються спеціальні класи для роботи з різними технологіями доступу до даних, наприклад, клас контексту даних Entity Framework. Тут знаходяться репозиторії, через які рівень бізнес-логіка взаємодіє з базою даних.

Багаторівневі додатки передбачають, що кожен рівень відповідає за свою функціональність та крайні рівні на зв'язані між собою. Так, рівень представлення, що складається з контролерів в ASP.NET MVC, не може безпосередньо звертатися до бази даних чи рівня доступу до даних, а повинен звертатися до рівня бізнес-логіки, який передає йому результати обробки даних. Незалежність рівнів передбачає, що рівень доступу до даних не залежить від інших рівнів, рівень бізнес-логіки залежить від рівня доступу до даних, а рівень представлення - від рівня бізнес-логіки [1].

При цьому, розділення рівнів на окремі проекти не є обов'язковим. Головне, щоб функціонал кожного рівня був логічно зв'язаний і відповідав за своє завдання (див. Рис. 2.1). Впровадження залежностей між компонентами також є невід'ємною ланкою багаторівневих додатків.

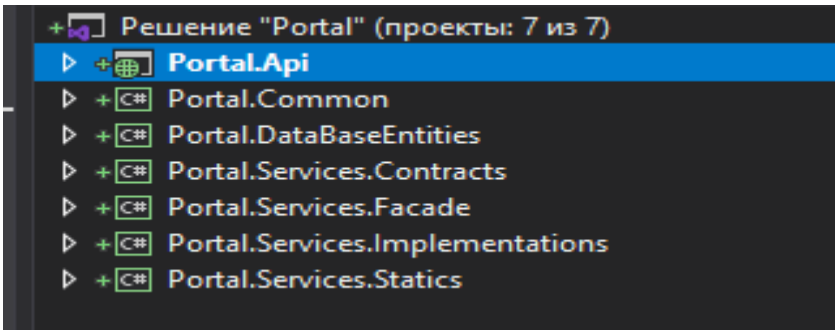

Рисунок 2.1 – Будова збірки

2.4. Вибір СУБД

Для даного типу програми ідеально підходить MS SQL Server - інтегрований пакет програм, розроблений компанією Microsoft. Ця система призначена зокрема для створення, розгортання та управління промисловими безпечними, масштабованими та надійними програмами. Використання MS SQL Server дозволяє значно збільшити продуктивність інформаційних технологій. Це досягається зменшенням складності побудови, розгортання та управління програмами, що працюють з базами даних. Окрім цього, MS SQL Server дозволяє ефективно розділяти дані між платформами, програмами та пристроями. Це зроблено для полегшення поєднання внутрішніх та зовнішніх систем і забезпечення їх взаємодії.

SQL Server - це комплексний пакет, який включає в себе низку інструментів, що спрощують розробку та управління базами даних. Ці інструменти включають в себе:

1. Реляційну базу даних, яка є масштабованою, безпечною та надійною та забезпечує підтримку різних видів даних, включаючи структуровані та неструктуровані.

2. Сервіси реплікації даних, які гарантують перенесення даних для розподілених та мобільних програм обробки даних. Система забезпечує високу доступність, масштабованість та інтеграцію з різнорідними системами, що включає бази даних Oracle.

3. Notification Services, які забезпечують можливості розповсюдження

персоналізованих та своєчасних повідомлень до великої кількості підключених та мобільних пристроїв.

4. Integration Services, які дозволяють отримувати, переводити та завантажувати інформацію в інформаційні сховища підприємства. Інструменти також забезпечують інтеграцію даних.

5. Analysis Services, що надають аналітичну обробку даних в режимі реального часу та інтелектуальний аналіз даних (Data Mining). Ці засоби дозволяють ефективно аналізувати великі та складні набори даних.

6. Кінцевий інструмент, Reporting Services, який забезпечує можливість створювати, керувати та доставляти традиційні та інтерактивні звіти за допомогою веб-технологій [5].

SQL Server надає широкі можливості з управління та розробки баз даних, включаючи наступні інструменти:

Інструменти управління: SQL Server має вбудовані засоби для управління та настроювання баз даних. Стандартні протоколи доступу до даних спрощують інтеграцію з існуючими системами, що дозволяє раціоналізувати процеси. Більш того, SQL Server має вбудовану підтримку Web-служб для забезпечення взаємодії з іншими системами та платформами.

Інструменти розробки: SQL Server надає інтегровані інструменти розробки для обробки даних, OLAP та звітності. Ці інструменти тісно інтегровані з MS Visual Studio, що дозволяє розроблювати наскрізні рішення для розробки інформаційних систем. Кожна підсистема SQL Server має власну об’єктну модель та API для розширення системи унікальним для бізнесу компанії напрямку. Крім того, ці інструменти забезпечують можливість розробки у будь-якому напрямку в залежності від потреб бізнесу.

SQL Server - ідеальна платформа для роботи з даними, яка надає безліч переваг. Зокрема:

Вона дозволяє користувачам ефективніше використовувати свої дані, завдяки вбудованим функціям управління, аналізу та звітності.

SQL Server дає змогу збільшити продуктивність роботи завдяки широкому

спектру інтелектуальних ресурсів підприємства та можливості інтеграції з популярними програмами, такими як MS Office. Інформаційна підтримка, що надається сервером, дає гнучкий та своєчасний доступ до важливої і аналітичної інформації для співробітників сфери інформаційних технологій підприємства.

SQL Server робить розробку та управління галузями промисловості й аналітичними програмами простішим, завдяки своєму гнучкому середовищу розробки й інтегрованим і автоматизованим інструментам управління адміністраторам баз даних.

Дана платформа дозволяє розраховувати на зниження загальних витрат на реалізацію та підтримку, завдяки простоті використання та інтегрованому підходу до розробки. До того ж, максимально мінімізується час на повернення інвестицій, пов'язаних з базами даних [10].

2.4.1 Вибір ORM системи

Для розробки програми була обрана ORM система Entity Framework, з використанням підходу Code First для роботи з базою даних. ORM, або Object-Relational Mapping, є технологією програмування, яка дозволяє конвертувати моделі ООП у відповідні типи даних, що використовуються в сховищах даних.

Застосовуючи ORM, розробники можуть спрощувати процес збереження та вилучення об'єктів в реляційну базу даних, перетворюючи дані між двома несумісними станами. Більшість ORM інструментів ґрунтуються на метаданих бази даних та об'єктів програмування, тому об'єктам не потрібно знати про структуру бази даних, а базі даних - не потрібно знати, як дані організовані в додатку.

ORM забезпечує повне розділення завдань у добре спроектованих додатках, де база даних та додаток можуть працювати з даними в своїх вихідних форматах. Використання Entity Framework та підходу Code First полегшує подальшу розробку програми та її підтримку. Код стає більш лаконічним, а розробники можуть легко дотримуватись кращих практик програмування [3].

Entity Framework - це спеціальна об'єктно-орієнтована технологія на базі

фреймворка .NET, яка дозволяє працювати з даними. У порівнянні з традиційними засобами ADO.NET, що дозволяють створювати підключення та команди для взаємодії з базами даних, Entity Framework надає більш високий рівень абстракції. Цей рівень дозволяє абстрагуватись від самої бази даних та працювати з даними незалежно від типу сховища.

Основна концепція Entity Framework полягає у понятті "сутність" (англ. entity). Сутність - це набір даних, що асоціюється з певним об'єктом. Отже, замість роботи з таблицями, дана технологія працює з об'єктами та їх наборами на концептуальному рівні.

За допомогою Entity Framework розробники можуть швидко та просто проектувати бази даних, виконувати їх міграції, а також працювати з даними, забезпечуючи високий рівень безпеки та надійності. Entity Framework дозволяє розробникам працювати зі структурами даних на більш абстрактному рівні, що знижує час, потрібний для написання і тестування коду. Таким чином, використання Entity Framework дозволяє прискорити процес розробки програмного забезпечення та знизити витрати на його підтримку.

Кожна сутність, як і будь-який об'єкт, має низку властивостей. Наприклад, у випадку, якщо сутність описує людину, то до її властивостей можуть відноситись ім'я, прізвище, зріст, вік та вага. Властивості не обов'язково представляють прості дані типу int, сутність може мати й більш складну структуру даних. У кожної сутності є одна або кілька властивостей, які є ключами та відрізняють її від інших. Такі властивості одноз називають ключами [19].

Асоціативні зв'язки у базі даних можуть мати різну структуру: один-до-багатьох, один-до-одного, або багато-до-багатьох. Ці зв'язки схожі на ті, що існують у реальних базах даних, де використовують зовнішні ключі для пов'язання таблиць.

Entity Framework використовує запити LINQ для отримання даних з бази даних. Це дозволяє не тільки отримувати рядки даних, але й отримувати пов'язані об'єкти, які належать до різних асоціативних зв'язків.

Також важливою концепцією в Entity Framework є Entity Data Model. Ця модель відповідає класам сутностей в базі даних. Entity Data Model містить три рівні: концептуальний, сховище та зіставлення. Кожен з цих рівнів об'єднує класи сутностей з таблицями бази даних.

Додаток використовує класи сутностей для опису даних на концептуальному рівні. Рівень сховища визначає таблиці, стовпці і відносини між ними в базі даних. Рівень зіставлення встановлює відповідність між властивостями класу суті і стовпцями таблиці.

Це дозволяє взаємодіяти з таблицями бази даних через класи, визначені у додатку.

Entity Framework надає три можливості взаємодії з базою даних:

1. Database First - генерація набору класів, які відображають модель конкретної бази даних.

2. Model First - спочатку розробляється модель бази даних, а потім Entity Framework створює реальну базу даних на сервері.

3. Code First - розробник створює класи моделі даних, які будуть зберігатися в базі даних, а потім Entity Framework генерує базу даних та її таблиці за цією моделлю.

Entity Framework має контекст, який управляє взаємодією між класами та базою даних. Контекст не є специфічним для Code First, це загальна особливість Framework.

Code First додає конструктор моделі, який перевіряє класи, що керуються контекстом, і використовує конвенції або правила для визначення, як ці класи та їх взаємозв'язки описують модель, і як вона повинна відображатися у базі даних.

Code First також надає можливість використовувати модель для створення бази даних за необхідності.

Розділ 3. РЕАЛІЗАЦІЯ ПРОЕКТУ

3.1 Розробка користувачів

Для керування профілем користувача, в контролері “UserController”, було створено наступні методи:

OnBoarding() - метод, який використовуватиметься при першому вході користувача і повідомлятиме користувачів, з спеціальністю "Human Resources", про нового користувача в системі. Для цього, викликається метод InformHRsAboutNewUser() сервісу “UserService”, в якому, з бази даних отримуються id та пошта користувачів, які мали спеціальність "Human Resources". Після цього, за допомогою розроблених раніше методів, відбувається створення сповіщень та надсилання поштових повідомлень.

Update() – метод, який приймає розроблену раніше модель, для оновлення данихпевного користувача, звичайним користувачем, “UpdateUsersInformationByUserIncomeModel”. Після перевірки моделі, викликається метод UpdateUsersInformationByUser(), в якому з бази отримується потрібний користувач, оновлюються відповідні дані з моделі та фіксуються зміни.

Update() – метод, який приймає розроблену раніше модель для оновлення данихпевного користувача адміністратором “UpdateUsersInformationByAdminIncomeModel”. Після перевірки моделі викликається метод UpdateUsersInformationByAdmin(), в якому з бази отримується потрібний користувач, і перевіряється, чи запит на оновлення даних не надаіслав користувач, дані якого необхідно оновити. Якщо дані оновлює інший користувач, проводимо перевірку на права доступу до кожної категорії даних. При наявності клейми "Partial profile management (contact information)", відбувається оновлення скайпу, номера телефону, офісу та робочих годин користувача. Далі, такими ж блоками умови, оновлюються інші дані користувача. Якщо, дані користувача оновлюються тим самим користувачем,

оновлюються лише дозволені при проектуванні дані.

GetUser() – метод, в якому отримується Id користувача, і повертається модель “UserViewModel”, з основними даними користувача. Для цього викликається метод GetUserById(), в якому з бази отримується користувач, проводиться перевірка його наявність в системі та заповнення даних моделі. Додатково створено метод GetAllSkillsByUserId(), в якому отримуються навички користувача, і GetUserSpecialityName(), для отримання спеціальності.

GetList() – метод, в якому приходить вхідна модель “GetPaginatedListUserIncomeModel”, для отримання всіх користувачів, з відповідними параметрами та використовується пагінація. Після перевірки вхідної моделі, викликається метод GetListByFilter(), сервісу “UserService”. Спочатку, в методі створено змінну “itemsToSkipCount”, в яку занесено кількість моделей даних, які потрібно пропустити. Далі, створено колекцію, типізовану командами, та розміщено блок умови, в якому, якщо в вхідній моделі є Id проекту, в колекцію добавляється команда проекту. З отриманих команд, проводиться вибірка користувачів, які ще не видалені, та за допомогою методу Select(), отримуємо лише колекцію Id користувачів. Далі, проводиться вибірка всіх користувачів з бази даних, окрім адміністратора, і з них, за допомогою методу Where(), проводиться фільтрація по кожному з властивостей моделі. Для цього, в умові робиться перевірка по властивості, лише в випадку, якщо властивість моделі не null. Після вибірки проводиться сортування користувачів по імені, пропускається потрібна кількість, та формується для кожного, вихідна модель, додаючсь дані, необхідні для пагінації.

GetUserStatus() – метод, для отримання статусу користувача, який приймає id користувача. В методі за допомогою блоків умов, перевіряється тип дня, і в залежності від цього, додається відповідний статус. Якщо ж це звичайний робочий день, за допомогою методу IsUserAvailable(), перевіряється чи користувач в момент запиту знаходиться на роботі. Для цього, необхідно отримати теперішню годину та хвилину за допомогою структури “DateTime”, та прирівняти до вказаного користувачем часу.

GetUserProjects() – метод для отримання всіх проектів користувача, в якому, за допомогою методу GetProjectsByUserId(), отримуються всі проекти даного користувача. Вибірка проводиться через команду, адже зв'язок починається з моделі “TeamUser”. Після отримання проектів, для кожного формується модель “ProjectViewModel”, відсортована по статусу проекту та його назві.

GetAllSpecialities() – метод, для отримання занесених в базу спеціальностей користувачів.

GetAllUsersSkills() – метод, для отримання занесених в базу навичок користувачів.

DisableUser() – метод, для видалення користувача, в якому, з бази отримується потрібний користувач та для властивості “IsDisabled”, вказується значення true, а даті видалення, вказується теперішній час. Далі, відбувається видалення користувача з усіх команд, та вказується всім прикріпленим до нього квикам підтведження відпусток, статус “Approved”, після чого, робиться те саме для квитків підтвердження понаднормових годин. Також, потрібно видалити всі відпустки та квитки підтвердження відпусток, які до них прикріплені, даного користувача.

Всі вказані вище дії, виконуються за допомогою вибірки даних з необхіднох таблиці та перебір їх за допомогою методу foreach(), після чого фіксуються зміни.

AllBirthdays() – метод, для отримання всіх дат, днів народжень користувачів, разом з фото та іменем користувача.

GetYearStatistics() – метод, для отримання кількості нових та видалених користувачів за рік. В ньому проводиться звичайна вибірка з бази даних, з вказаним параметром та функції Count().

GetHrStatisticExelReport() – метод, для отримання статистики по користувачам за певний період, в форматі excel, захищений атрибутом ClaimsAuthorization(), який забороняє доступ, якщо в заголовку запиту немає прав “HolidaysManagement”. В ньому викликається метод GetUserListForStatistic(), для формування моделі, яку буде занесено в excel

документ. В ньому отримуються користувачі, в яких перший робочий день, входить в вибірку та проходить сортування по ньому, для кожного формується модель, яка містить дату та ім'я користувача. Так само проводиться вибірка для користувачів, в яких дата видалення входить в вибірку. Після формування моделі, вона передається в метод GetHrStatisticExelReport(), сервісу "UserExcelExport". В методі створюється документ з робичим листом, під назвою "UserStatistic", після чого, викликається метод AddHeaderToTable(), в якому добавляється відповідний заголовок та його стилізація. Для добавлення самих користувачів, створено метод AddUsersToTable(), який буде спільним для видалних та доданих користувачів. Він приймає колекцію користувачів та координати початкової клітинки, починає перебір коритувачів та занесення в таблицю. Принцип роботи з excel документом описаний в розділі про логування часу (рис. 3.1).

Name	Added User		Name	Removed User
Андрей Щадило	12.05.2020			
Oleksiy	07.05.2020			
Olena Smol	30.04.2020			
Ihor Bordun	30.04.2020			
One Love	28.04.2020			
Olena Max	27.04.2020			
Olena Developer	24.04.2020			
mark PM	24.04.2020			
BIBIbi PM	24.04.2020			
Ihor Chipak	24.04.2020			
Tatiana Husnay	23.04.2020			
Maxim Chief	23.04.2020			
Andryi Matchak	23.04.2020			
Микола Куціль	23.04.2020			
Марк Білик	22.04.2020			
Lorem Ipsum	22.04.2020			
Mike Shevchenko	14.04.2020			
Olena PM	14.04.2020			
Богдан Івахів	13.04.2020			
Olena HR	13.04.2020			
Тарас PM	13.04.2020			
Taras Pylypchuk	13.04.2020			
Тарас Пилипчук	13.04.2020			
Наталя HR	13.04.2020			
Ihor Bordun	12.04.2020			
M Bi	10.04.2020			
Marian - Makar Bilyk	10.04.2020			
Bohdan Ivakhiv	10.04.2020			
Andryi Matchak	10.04.2020			
Olena AM	10.04.2020			

Рисунок 3.1 – HR Excel таблиця по користувачах

3.2 Авторизація та автентифікація

Для забезпечення авторизації та аутентифікації користувачів у проекті використовуються можливості платформи Identity, розроблених допоміжних Owin Middleware та створених методів у контролері "AccountController".

Один з Middleware - клас "ApplicationUserManager", що базується на базовому класі платформи Identity. Клас відповідає за регулювання додаткових даних про користувача.

Ще один клас, необхідний для аутентифікації - "GoogleAuthenticationConfig". У конфігурації GoogleOAuth2, що знаходиться у класі, спочатку отримують "ClientId" та "ClientSecret". Потім за допомогою простору імен "Microsoft.Owin.Security.Google" створюється екземпляр класу "GoogleOAuth2AuthenticationProvider", який відповідає за проведення аутентифікації через Google. У провайдері вказується URL для редіректу. При успішній авторизації додаються email, ім’я користувача, token та token оновлення.

Для автентифікації користувачів використовуються класи "OAuthAuthorizationServerOptions" та "OAuthBearerAuthenticationOptions". Вони містять параметри OAuth, які перебувають у класі Startup і є екземпляр класу "OAuthAuthorizationServerOptions". У цих параметрах визначається час життя токена в 40 хвилин, кінцева точка авторизації, шлях для отримання нового доступу до токену, провайдер Google і клас "ApplicationRefreshTokenProvider", який оброблятиме оновлений токен.

Клас "ApplicationRefreshTokenProvider" успадковує функціонал від системного класу "AuthenticationTokenProvider", щоб забезпечити зміну логіки виконання його методів.

Метод "CreateAsync()" використовує "AuthRepository", щоб отримати вхідний контекст Google OAuth та створити новий токен оновлення, який хешується з використанням методу "ReceiveAsync()". Цей токен в подальшому буде використаний для оновлення існуючого токена доступу. Оскільки токен

оновлення не може жити безкінечно, йому також задано час життя в 24 години, щоб забезпечити безпеку.

Вся функціональність, яка була перерахована вище, є middleware. Ці middleware починають свою роботу з моменту запуску програми та взаємодіють між собою з використанням стандартних схем. Це було досягнуто за допомогою використання системних класів авторизації та аутентифікації як бази для створення власних класів.

В контролері "AccountController" було створено два методи для забезпечення потрібного рівня автентифікації:

Перший з них - метод "GetExternalLogins()", що створений з метою вибору провайдера автентифікації. У цьому методі доступні провайдери отримуються та додаються дані про них, а потім відповідь надсилається.

Другий метод - "GetExternalLogin()", призначений для авторизації користувача. Він перенаправляє користувача до зазначеного провайдера автентифікації та, у разі успішного входу, додає користувача до бази даних як нового, якщо це перший його вхід, або додає права на доступ до існуючого облікового запису. У методі спочатку відбувається перевірка на помилки під час входу. Якщо виявлені помилки, за допомогою методу "ChallengeResult(provider, this)" відбувається редірект на потрібного провайдера. Якщо користувач не автентифікований, то відбувається редірект на провайдера автентифікації. Для отримання даних провайдера було створено метод "GetFromIdentity()" у сервісі "ExternalLoginDataService". Метод приймає клас "ClaimsIdentity".

Після отримання необхідних даних з клеймів формується модель "ExternalLoginDataViewModel". Якщо в методі немає даних від провайдера, тоді користувач буде перенаправлений на сторінку введення своїх правильних даних. З конфігурації отримуються дозволені домени для входу. Якщо вони присутні у конфігурації, перевіряється, чи співпадає email користувача з дозволеними доменами. Якщо не співпадає, то користувач буде перенаправлений для введення правильних даних. Після цього за допомогою "UserManager" отримується користувач з системи. Якщо користувач не є null, тоді він уже зареєстрований.

Клейми користувача будуть очищені, а його дані будуть оновлені в базі даних. Якщо користувач не зареєстрований, тоді всі "Claim" з Owin Middleware будуть отримані для даного користувача. З них отримуються базові дані, що надаються провайдером аутентифікації, та заносяться в модель "ApplicationUser". Користувач буде доданий до бази даних за допомогою методу CreateAsync() з класу "UserManager". Якщо користувач був успішно створений, тоді для нього буде додана роль "User" та додаткові дані.

На завершення, за допомогою методів сервісу "CalendarService", буде перевірено наявність календаря та необхідності створення нового календаря для користувача. Після цього вхід користувача буде завершений.

Для отримання даних про користувача створено метод GetUserInfo(). В цьому методі проводиться отримання основних даних про користувача та формування моделі "UserInfoViewModel". Для перерахування нарахованих днів відпусток використовується метод SetUserVacation(), який був описаний раніше.

3.3. Звіти

Для управління часовими журналами потрібен окремий контролер. Саме тому було створено контролер "TimeLogController", який містить метод "Get" для формування звіту про користувача. В цьому методі задано шлях, в заголовку якого містяться такі параметри:

- userId: унікальний ідентифікатор користувача, за яким формується звіт;
- startDate: дата початку періоду, за який треба згенерувати звіт;
- endDate: кінцева дата, за яким треба згенерувати звіт.

В процесі прийому даних від користувачів, контролер використовує перевірку стану моделі запиту на наявність помилок за допомогою властивості "ModelState.IsValid" у класі "ApiController". Якщо виявлено помилки, то створюється помилка "ArgumentException", яка інформує про помилки під час обробки запиту на контролері. Далі за допомогою платформи Identity контролер знаходить Id користувача, який надіслав запит. Оскільки дата надходить у

вигляді рядка, спочатку необхідно перевірити, чи відповідає вона коректному формату дати, а потім перетворити її в тип "DateTime", щоб з ним працювати. Для цього використовується метод TryParse() зі структури "DateTime", який намагається перетворити рядкове значення в тип "DateTime". Залежно від результату спроби, метод поверне певне буліанівське значення.

Перед тим, як приступити до розробки методу та створення репорту, необхідно визначитися з моделлю даних, яку буде повертати програма. Успішність нашого проекту дуже залежить від якості цієї моделі. Якщо модель структурована погано, то алгоритми, які забезпечують швидкість створення репорту, не зможуть працювати ефективно.

Оскільки дата, на яку створюється звіт, повинна бути відображена по місяцях, то на верхньому рівні моделі має бути розміщений місяць, за який створюється репорт. Також, потрібно включити ім'я та ID користувача, для якого створюється звіт. Адже один користувач може брати участь в декількох проектах одночасно. Модель має містити масив звітів по кожному проекту з підсумками.

Крім того, необхідно розробити модель для відображення вихідних днів, днів відпустки та свят. Ці дні не пов'язані з конкретним проектом, тому необхідно створити окрему модель для їх відображення.

Модель звіту про проект міститиме назву та Id проекту, а також поля для підбиття підсумків за окремими таймлогами та загальний підсумок щодо проекту. Крім того, модель проекту міститиме моделі днів, які стануть основою для подальшого відображення та створення нових таймлогів.

Модель дня, у свою чергу, міститиме день у форматі "DateTime", моделі таймлогів, підсумки за день та буліанівське значення, що вказує, чи є день активним. Ці дані необхідні для правильної роботи звіту, а моделі таймлогів задані окремими типами, що дозволяє зменшити обсяг пам'яті, що виділяється під них.

Підсумки з усіх проектів будуть містити загальний результат за всі дні та окремо за кожен день. Підсумки за конкретний день будуть містити день у форматі "DateTime", значення, що вказує, чи є це вихідним днем, і підсумки за

день.

Після проектування моделі даних нашого TimelogService, ми потребуємо розробити інтерфейс взаємодії з ним.

Інтерфейс буде містити метод GetTimeLogReportByUser(), який буде мати відповідну структуру.

У реалізації методу необхідно спочатку розділити стартову та кінцеву дату вибірки по місяцях. Для цього ми розробили метод, який приймає стартову і кінцеву дату вибірки і повертає масив стартових дат кожного місяця, у форматі "DateTime". Далі проводиться вибірка таймлогів користувача, які входять в діапазон дат, що був вибраний. Краще зробити вибірку з бази даних за весь період, а не окремо за кожний місяць, оскільки база даних може містити тисячі таймлогів і максимальна ефективність буде досягнута, якщо зробити найменшу можливу кількість запитів на вибірку даних. Після вибору всіх таймлогів, які відповідають діапазону дат, ми можемо працювати з ними в коді і робити операції з розділу по місяцях та вибірки за умовами. Далі ми здійснюємо вибірку даних користувача з бази даних, для якого ми будуємо звіт. Після отримання необхідних даних, які є спільними для всіх проектів користувача, ми можемо перейти до перебору масиву місяців, які ми отримали.

Перед тим, як створювати модель репорту, необхідно отримати правильну кінцеву дату. Це можна зробити за допомогою методу DaysInMonth() структури "DateTime", який поверне кількість днів у місяці. Отримане значення передається в конструктор структури, щоб отримати правильний формат кінцевої дати.

Щоб уникнути дублювання коду і створити можливість розширення методу, необхідно створити додатковий метод, який буде повертати масив репортів проектів на конкретний місяць та підсумок.

У методі спочатку отримують масив всіх проектів, на яких працює користувач. Якщо у користувача немає проектів, метод повертає тип даних null. В іншому випадку створюється модель, яка буде повернута з методу. Потрібно припустити можливість обробки ситуації, коли модель не потрібно створювати.

Кожен раз, коли створюється модель, виділяється пам'ять в кучі, що, в подальшому, може призвести до збору сміття та нераціонального використання пам'яті.

Перед тим, як генерувати репорт за окремий проект, створюється масив, у який буде поміщено спільне значення підсумку по днях для всіх проектів, а також змінна isFirst, яка містить інформацію, чи це перший проект, який був перебраний. Це необхідно для того, щоб можна було заповнити масив з підсумками днів, які є спільними для всіх проектів.

При обробці масиву проектів, спочатку створюється модель репорту для кожного проекту з додаванням назви та ідентифікатора проекту. Далі створюється цикл для перебору днів у місяці. Для кожного дня створюється модель, заповнюється дата та значення "IsWeekend", що залежить від "DayOfWeek".

Умовний блок виконується, якщо "isFirst == true" та повідомляє про додавання дня у модель підсумку по днях. Також, у випадку, якщо день не є активним або час репорту більший за сьогоднішній, ітерація циклу пропускається. Якщо умови виконуються, застосовується метод "GetTimelogsReport()", що добавляє модель таймлогів.

Після проходження всіх днів, додається підсумок та властивість "isFirst" отримує значення false. Цикл повторюється для кожного проекту, після чого додається підсумкова модель та дані повертаються. Модель місяця заповнюється даними, що повторюються через виклик методу. Також, додається репорт з типами днів шляхом перебору всіх днів місяця та додавання відпусток, вихідних та свят до відповідної моделі.

3.4. Excel звіти

У контролері "TimeLogController" був створений метод "GetReportInExcel()", що формує репорт користувача у форматі Excel-файлу. Спочатку модель перевіряється та формат дати перетворюється, після чого

запускається метод "GetUserExcelReport()" у сервісі "UserExcelExporter". Цей метод є перехідною ланкою від формування описаного репорту до методу, що перетворює його в Excel-файл.

У методі змінна "userTimeLogReport" отримує сформований репорт, який передається до методу "GenerateUserExcelTableService.GenerateExcelUserReport()".

Для створення Excel-файлу використовується бібліотека "OfficeOpenXml". У методі створюється внутрішня модель "ExcelPackage". Далі відбувається перебір моделі репорту, починаючи з верхнього рівня місяців, та відображення її на файлі Excel. На кожній ітерації створюється окремий лист за допомогою методу "Add()" та змінна із стартовим рядком. Відображення таблиці розбито на три методи:

- AddHeaderToWorkheet() – відповідає за формуванняверхньої частини таблиці;
- AddTotalToWorkheet() – відповідає за формуванняпідсумків в таблиці;
- AddProjectsToBodyWorkheet() – відповідає за формування основного тіла таблиці;

Також було створено додаткові методи:

- AddVacationsToBodyWorkheet() – відповідає за додаванняв тіло таблиці відпусток;
- AddPlanToWorkheet() - відповідає за додавання робочого плану в таблицю;
- AddStyleToWorkheet() - доповнює стилізацію таблиці.

У методі "AddHeaderToWorkheet()" використовую властивість "Cells[]" для вибору клітинки на робочому листі Excel-документа. Після вибору клітинки я можу додавати текст, значення, формули та стилізацію. Метод "LoadFromText()" використовую для завантаження даних у форматі string, наприклад, для внесення імені користувача. Стилізацію додаю після додавання даних за допомогою властивості "Style". Таким чином, я не повторюю стилізацію декілька разів. Віджети "Border" встановлюють рамку для клітинки, а "Fill" вказує тип та колір

заливки.

Після додавання статичних даних додаю кількість днів у заголовок за допомогою циклу "for()". Після кожної ітерації збільшую значення "column" на одиницю. Додаю підсумок, переходжу на наступний рядок та повертаю його з методу.

Метод "AddTotalToWorkheet()" використовується для формування підсумку по проектах. У методі перебираються підсумки по днях, якщо вони не дорівнюють нулю, і додаються. Після чого додавати основний підсумок.

Метод "AddProjectsToBodyWorkheet()" використовується для формування всіх проектів у репорті. Спочатку змінній "column" присвоюється початкове значення 1. Потім перебираються всі проекти користувача за допомогою "foreach()". Спочатку додається назва проекту, потім перебираються дні, і якщо вони не дорівнюють 0, вносяться години. Для певних днів додається стилізація, для вихідних - сірий колір клітинки.

У методі AddVacationsToBodyWorkheet() відбувається додавання всіх наявних відпусток, вихідних та святкових днів у репорт. Це робиться шляхом перебору масиву з типами днів та перевірки їх за допомогою switch(). Крім того, вноситься необхідна стилізація.

У методі AddPlanToWorkheet() план виконання записується шляхом перебору масиву з днями.

Метод AddStyleToWorkheet() призначений для надання стильового оформлення робочому листу. Використовуючи властивість SelectedRange[] робочого листа, весь репорт вибирається та отримує відповідний стиль шрифту.

Після внесення всіх змін до репорту, він зберігається та перетворюється до типу "MemoryStream". Цей тип повертається з методу.

У контролері створюється відповідь про успішне виконання. Файл має визначений тип та назву. Після цього виконання завершується. (див. рис. 3.2)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1	Andryi Matchak																															
2	Activities	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Total
3	**Projects**													22	6				6						12							46
4	FastFive																								12							12
5	Mark-bug fixer													16	6				6													28
6	Good luck													6																		6
7	**Pre-Sales**																															0
8	test olena magic																															0
9	pre																															0
10	test olena test																															0
11	test																															0
12	**Departments**																															0
13	Account managers																															0
14	test 2 claim																															0
15	test claim																															0
16	test department																															0
17	PM department																															0
18	**Others**																															0
19	Lectures																															0
20	Mentorship																															0
21	Interview (Recruitment)																															0
22	Interview (Feedback)																															0
23	Interview (Sales)																															0
24	Vacation																															8
25	**Total work hours**													22	6				6						12							46
26	Plan	8	8	8	0	0	8	8	8	8	8	0	0	8	8	8	8	8	0	0	8	8	8	0	8	0	0	0	8	8	8	160

Рисунок 3.2 – Excel репорт по користувачу

Метод GetAllUsersExcelReport() було розроблено для створення звіту про всіх користувачів. Його функціональність полягає в генерації звіту, що стосується всіх користувачів, і передачі його в метод GenerateExcelAllUsersReport() (таб. 3.1).

Для початку, звіт переглядається на рівень місяців, створюються робочі листи з назвами місяців та викликаються наступні методи:

AddHeaderToWorkheet () - метод для створення заголовку звіту. У ньому фіксуються заголовки, які будуть на початку та в кінці звіту. Кількість інших активностей визначається та фіксується в змінній "countOthers". Для кожного значення з першої частини заголовку необхідно поєднати клітинку з нижньою, використовуючи властивість "Merge". Після занесення данних, виводиться заголовок та з'єднуються клітинки вбік, відповідно до кількості інших активностей. Далі, за допомогою циклу виводяться додаткові активності внизу таблиці. На завершення створюється друга частина заголовку та з'єднуються колонки. Оскільки з'єднані колонки не можуть бути автоматично визначені, для кожної позиції задається її ширина вручну.

Метод AddAllUsersToBodyWorkheet() дозволяє додати всіх користувачів

до тіла таблиці. Значення задано фіксованою позицією для коментарів та стартовою позицією для користувачів. Всі користувачі перебираються і дані додаються до таблиці з відповідним стилеутворенням. Цикл був створений для перебору підрахованих значень, і залежно від значення до них додається колір.

AddTotalToWorkheet() додає підрахунок для всіх користувачів. У цьому методі для кожного поля встановлюються формули для динамічної зміни значень і додається стилеутворення.

Метод AddFormulas() додає формули для кожного користувача. Для цього перебираються всі користувачі і для кожного з них додаються формули для підрахунку підсумків.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1							Others										
2	№	Name	Project Hours	Project Overtime	Pre-Sales	Departments	Interview (Feedback)	Interview (Recruitment)	Interview (Sales)	Lectures	Mentorship	Vacation	Total	Plan	Total out of Plan (%)	Difference	Comments
3	1	AM Olena	39	0	8	19	1	2	1	0	3	0	73	160	45,6%	-87	гшгншгнш
4	2	Bi M	0	0	0	3	0	0	0	0	0	0	3	160	1,9%	-157	try rublia
5	3	bi mark	21	5	0	0	0	0	0	0	0	0	26	160	16,3%	-134	asdS
6	4	BIbi BIBIbi	0	0	0	0	0	0	0	0	0	0	0	160	0,0%	-160	
7	5	Bordun Ihor	261,5	0	13	8	0	0	0	0	0	8	290,5	160	181,6%	130,5	sdfsd
8	6	Chief Maxim	0	0	0	0	0	0	0	0	0	0	0	160	0,0%	-160	
9	7	Chipak Ihor	0	0	0	0	0	0	0	0	0	0	0	160	0,0%	-160	
10	8	Developer Olena	33	9	0	4	0	0	0	0	0	8	54	160	33,8%	-106	негенгнег
11	9	HR Olena	0	0	0	2	0	0	0	0	0	0	2	160	1,3%	-158	fsdfsef
12	10	HR Наталя	17	0	0	0	0	0	0	0	0	16	33	160	20,6%	-127	
13	11	Husnay Tatiana	0	0	0	0	0	0	0	0	0	0	0	160	0,0%	-160	
14	12	Ipsum Lorem	0	0	0	0	0	0	0	0	0	0	0	160	0,0%	-160	
15	13	Ivakhiv Bohdan	153	5	24	23	3	5	0	1	0	0	214	160	133,8%	54	ппавпавпвап
16	14	Marian - Makar Bilyk	256,5	5	0	68,5	59	50	55	49	67	0	610	160	381,3%	450	SDfSD
17	15	Matchak Andryi	21	0	0	0	0	0	0	0	0	0	21	160	13,1%	-139	
18	16	Matchak Andryi	78	0	0	40	0	0	0	0	0	8	126	160	78,8%	-34	opopioppo
19	17	PM Olena	6	3	0	0	0	0	0	0	0	0	9	160	5,6%	-151	zsdgzdfdfvdfv
20	18	PM Тарас	28,5	11	0	8	0	0	0	3	0	16	66,5	160	41,6%	-93,5	
21	19	Portal InVerita	0	0	0	0	0	0	0	0	0	0	0	160	0,0%	-160	ghhfghfghgfh
22	20	Pylypchuk Taras	67	2	0	0	0	0	0	0	0	8	77	160	48,1%	-83	некнекнкен
23	21	Shevchenko Mike	23	0	0	0	0	0	0	0	0	0	23	160	14,4%	-137	
24	22	Білик Марк	69,5	0	0	6	0	0	0	0	0	0	75,5	160	47,2%	-84,5	uyytutyuu
25	23	Івахів Богдан	16	0	0	0	0	0	0	0	0	0	16	160	10,0%	-144	
26	24	Куціль Микола	0	0	0	0	0	0	0	0	0	0	0	160	0,0%	-160	
27	25	Пилипчук Тарас	24	0	0	52	15	0	8	7	0	0	106	160	66,3%	-54	
28			**1114**	**40**	**45**	**233,5**	**78**	**57**	**64**	**60**	**70**	**64**	**1826**	**4000**	**45,6%**	**-2174,5**	

Таблиця 3.1 – Загальний Excel звіт

3.5. Логування часу

Для логування робочого годинника в контролері "TimeLogController" було створено два методи.

Метод "Create()" приймає модель даних "TimeLogIncomeModel" та перевіряє її на наявність помилок. Потім він визначає ID користувача, який надіслав запит. Після цього за допомогою методу CheckIfUserHaveAccess() перевіряється можливість логування часу. Цей метод знаходиться у сервісі

"TimeLogService". Спочатку в методі "CheckIfUserHaveAccess()" за допомогою методу "CheckIfTimeLogPeriodClosed()" перевіряється, чи можливе логування часу у вибраний день. Потім ми отримуємо із бази даних проект, для якого логується час. Усі можливі помилки генеруються за допомогою умовних блоків. Якщо жодний із блоків умов не спрацював, виконання методу завершується, і передбачається, що логування часу дозволено.

У процесі виконання викликається метод логування часу Create() із сервісу TimeLogService. У цьому методі перебирається масив таймлогів і для кожного з них за допомогою методу CheckIfHoursValid() перевіряється валідність годинника. Потім таймлоги логуються у базі даних за допомогою методу CreateTimelogEntities(). У цьому методі створюється масив, у якому зберігатимуться створені таймлоги.

Спочатку перевіряється, чи потрібно створювати звичайний годинник. Якщо це необхідно, то створюється модель "TimeLog" і додається до створеного масиву. Після цього в блоці умови також необхідно перевірити необхідність створення понаднормового годинника і при необхідності створити його. Якщо в моделі немає звичайного годинника, але є понаднормовий, то в першу чергу перевіряється, чи створено звичайний годинник. Якщо так, то понаднормовий годинник заноситься в лог, в іншому випадку генерується помилка.

При створенні понаднормового годинника визначається адміністратор, відповідальний за підтвердження годинника, за допомогою методу GetAdminFromProject(). Якщо він є, то статус квитка надається "Pending", інакше - "Approved". В очікуванні годинника також виконується перевірка та створення.

Потім всі таймлоги з масиву додаються до бази даних за допомогою методу AddRange() і фіксуються зміни. Після цього з методу повертаються 3 таймлоги. Якщо таймлог було створено, повернеться сам об'єкт, інакше повернеться null.

Після створення годин роботи, облік відпусток користувача оновлюється автоматично. Якщо було створено понаднормові години, які потребують підтвердження адміністратора, тоді система створює сповіщення для адміністраторів та відправляє поштові повідомлення відповідним користувачам.

Далі, система створює модель відповіді, включаючи Id таймлогів та

повідомлення для відповідальних осіб. Інформація з цієї моделі передається через хаб, після чого відбувається відправлення відповіді.

Метод Update() обробляє запит з моделі даних "UpdateTimeLogIncomeModel". Спочатку, система проводить перевірку на дозвіл оновлення таймлогу, так само, як при його створенні. Потім, відбувається перевірка, чи є тайлоги для оновлення. Якщо немає, система генерує помилку з відповідним текстом. Далі, важливим кроком є оновлення таймлогу. Після цього, система надсилає сповіщення, оновлює дні відпусток користувача, як і при його створенні, після чого відправляє відповідь.

3.6. Відпустки та свята

“ VacationController”, було створено методи Create(), Update(), Delete(), UpdateStatus(), GetAllUserVacation() і GetUserVacationsForAdmin().

В методі Create() спочатку отримують вхідну модель створення відпустки. Потім проводиться перевірка запиту на наявність помилок та отримання Id користувача за допомогою платформи "Identity". Далі, дані передаються в метод CreateVacation() сервісу "VacationService".

Перед початком обробки запиту у методі CreateVacation() включені блоки перевірок вхідних даних на відповідність запланованим вимогам. Якщо відпустка оплачувана, система перевіряє, щоб не було перевищено максимальний ліміт днів після її створення.

Якщо відпустка включає додаткові дні, система виконує ті ж самі перевірки з врахуванням додаткових днів користувача. В разі, якщо відпустка не оплачується, додаткових перевірок не потрібно. Якщо ж потрібні перевірки не задовольняються, використовується статичний клас "ObjectExistenceChecker" для генерації помилок.

Якщо всі вхідні дані відповідають вимогам, система створює нову модель відпустки і додає її до бази даних. Після успішного створення відпустки визначається список відповідальних людей, які дозволять її користувачеві. Для цього, створено метод GetAdminUsersOnActiveProjects(). Він визначає

адміністраторів, учасників проекту користувача. Після визначення відповідальних користувачів, система видаляє повторюючіся Id і повертає результат з методу.

Далі, надсилається повідомлення на пошту за допомогою сервісу "EmailService", і створюються сповіщення з використанням сервісів "NotificationService" і "UserNotificationService". Після повернення даних з методу CreateVacation(), використовується "TicketNotificationsHub", щоб повідомити адміністраторів про квиток підтвердження відпустки

В методі Update() ми оновлюємо створену відпустку замість створення нової. Ми виконуємо ті самі перевірки та дії, що і при створенні.

В методі Delete() ми видаляємо раніше створену відпустку та інформуємо відповідальних людей про це, скориставшись нашими сервісами.

Метод UpdateStatus() відповідає за підтвердження адміністраторами відпустки. Метод працює на основі раніше розробленого рішення, змінюючи статус моделі "TicketAdminStatus" та, при необхідності, статус моделі "Ticket". У методі контролера ми перевіряємо модель та викликаємо метод UpdateVacationStatusByAdmin (), передавши вхідні дані. В цьому методі ми отримуємо користувача та відпустку з бази даних, створюємо модель "UpdateVacationTicketStatusModel" та передаємо її в сервіс "TicketService" для зміни статусу квитка. Потім ми перераховуємо доступні дні відпустки користувача, використовуючи метод SetUserVacation (). Цей метод працює за алгоритмом, розробленим раніше, де спочатку визначається кількість нарахованих днів за допомогою методу CalculateUserVacationCount (), потім, в залежності від умов, розділяється на теперішній та минулий роки за допомогою методу UpdateUserVacations(). Після оновлення днів відпусток, ми надсилаємо повідомлення про підтвердження відпустки.

Для отримання всіх відпусток, що знаходяться у користувачів, використовується метод GetAllUserVacation(), який представляє кожну відпустку у вигляді моделі "VacationViewModel".

Щоб отримати всі підтверджені заявки на відпустку користувача,

використовується метод GetUserVacationsForAdmin(). Оскільки основою є заявки, а не відпустки, необхідно спочатку отримати з бази даних усі заявки користувачів, пов'язані з цим користувачем. Потім проходить по отриманому масиву заявок у циклі foreach(), для кожної формується модель "VacationModelForAdmin". Сформований масив моделей відправляється назад.

Для свят був створений окремий контролер "HolidayController", в якому є методи для створення, оновлення, видалення та отримання всіх вихідних.

Метод GetList() перевіряє запит на наявність помилок і якщо помилок немає, повертає модель всіх вихідних, відсортованих за датою створення.

Метод Create() створює нове свято у базі даних, після чого надсилає повідомлення про створення електронною поштою та створює нове повідомлення у базі даних для всіх незаблокованих користувачів. Доступ до цього методу надається лише користувачам, які мають права управління святами. Модель, яку приймає даний метод, містить назву свята, його початкову та кінцеву дати. Назва свята обмежена 200 символами

Для оновлення свят в базі даних та повідомлення всім користувачам про оновлення, використовується метод Update(). Цей метод доступний тільки користувачам з Holidays Management Claim. Модель, що приймається в методі, містить ті ж самі дані, що й при створенні нового свята, додавши лише Id цього свята.

Також, метод Delete() видаляє свято з бази даних та повідомляє всі незаблокованих користувачів через нове сповіщення та пошту. Щоб виділити правило групі користувачів, методи можуть бути викликані тільки тим, у кого є Holidays Management Claim. В параметрах методу передається "holidayId".

3.7. Сповіщення

Для створення сповіщень в сервісі "NotificationService" було створено метод CreateNotificationForUsers(). Цей метод приймає масив користувачів, для яких необхідно створити сповіщення, текст та тип сповіщення. Для більш зручного

вибору типу сповіщення, використовується "Enum". У методі створюється сповіщення, для заголовка якого використовується метод GetTitleFromType(), що визначається по типу сповіщення. Після успішного створення сповіщення, його необхідно прикріпити до конкретних користувачів. Для цього в сервісі "UserNotificationService" створено метод CreateList(). У цьому методі для кожного користувача створюється "UserNotification" з вказівкою Id попередньо створеного сповіщення.

Для того, щоб наші користувачі могли зручно отримувати сповіщення, ми створюємо в контролері метод GetList() з використанням пагінації. Це дозволяє віддаляти дані частинами. Перш ніж отримати список сповіщень, ми отримуємо номер поточної сторінки та кількість елементів на одну сторінку. Потім передаємо ці дані в метод GetList() сервісу "UserNotificationService", де отримуємо список всіх сповіщень, що були створені для даного користувача, відсортованих по даті створення. Після чого ми відсікаємо всі елементи до потрібного і створюємо вихідну модель даних для кожного сповіщення, яке потім відправляється в відповідь.

Також в контролері ми добавляємо два методи для видалення сповіщень. Перший метод приймає Id сповіщення та видаляє лише одне сповіщення. Другий метод приймає масив Id та видаляє декілька сповіщень одночасно. Це дозволяє нашим користувачам проводити чистку свого списку сповіщень зручним для них способом.

Для взаємодії з поштою використовується клас SmtpClient, який дозволяє надсилати листи та контролювати процес.

Конфігурація сервісу зберігається в Web.config, де вказуються host, підтримка SSL сертифікату та порт.

Для ініціалізації SmtpClient з конфігурацією був розроблений метод InitializeSmtpClient().

Відправку листів керує метод SendEmailToGroup(). Він отримує заголовок та тіло повідомлення, які додаються до класу MailMessage, що є основою повідомлення разом з іншими параметрами.

Форматування шаблону повідомлення здійснюється за допомогою методів EmailTemplateService та HTML шаблонів. Було створено наступні методи для виконання цих завдань:

- GetHolidayTemplate() –метод формування повідомлення, які стосуються свят;
- GetProjectTemplate() – метод формування повідомлення, які стосуються проектів;

GetTimelogTemplate() – метод формування повідомлення, які стосуються таймлогів;

3.8. Google Calendar

При використанні Google Calendar, BackEnd нашої програми стає посередником для доступу до Google Calendar Api. Для доступу до Google Calendar Api, використовується Google.Apis.Calendar.v3, яку надає Google.

Для ініціалізації данного сервісу, створено окремий метод Init(). Спочатку потрібно дадати в клас GoogleCredential, данні нашого сервісного акаунту, та ключ нашої програми. Також добавляємо “Scope” на відкриття на зчитування, які формують наші права при доступі до Google Calendar Api.

Метод GetList(), отримує Id календаря та дати вибірки, перевіряє запит на помилки, і якщо їх не виявлено, ініціалізує Calendar сервіс та відправляє запит на отримання всіх подій, які створені на календар з данним Id. Після чого методом Execute(), відбирає тільки ті події які входять в вибірку.

Метод Create(), отримує модель данних для створення, перевіряє запит на помилки, і якщо їх не виявлено, ініціалізує Calendar сервіс. Проходить перетворення вхідної моделі в модель Event, яка надана бібліотекою Google.Apis.Calendar.v3.Data. Після чого методом Insert(), відправляє запит на створення.

Метод Update(), отримує модель данних для оновлення, перевіряє запит на помилки і якщо їх не виявлено, ініціалізує Calendar сервіс. Проходить

перетворення вхідної моделі в модель Event, яка надана бібліотекою Google.Apis.Calendar.v3.Data. Після чого методом Patch(), відправляє запит на оновлення.

Метод Delete(), отримує Id календаря та Id події, перевіряє запит на помилки і якщо їх не виявлено, ініціалізує Calendar сервіс та відправляє запит на видалення події з данним Id, які створені на календар з данним Id.

Також, сервіс “CalendarService”, містить метод для перевірки наявності календаря, та метод його створення.

Метод CheckIfCalendarExists(), приймає параметрами email користувача, ініціалізує сервіс. Після цього виконується запит на отримання календарів користувача. Якщо відповіді немає, це означає що в користувача відсутній календар, і викликається метод Create().

Метод Create() приймає параметрами email користувача, ініціалізує сервіс. Далі створюється модель календаря з, namespace “Google.Apis.Calendar.v3.Data”, вноситься часова зона, загальні дані та тип. Після цього надсилається запит на створення календаря та добавляється отримане Id календаря до відповідного поля в моделі користувача.

3.9. Активності

Для керування проектами, в контролері “ProjectController”, Було створено наступні методи:

- GetNamesList() – метод, для отримання назв існуючих проектів;
- GetCountriesList() – метод, для отримання назв країн;
- GetIndustriesList() – метод, для отримання назв існуючих індустрій;
- GetList() – метод, для отримання всіх проектів в моделі “GetProjectViewModel”;
- Get() – метод, для отримання одного проекту в моделі “ProjectViewModel”;

- Create() – метод, для створення нового проекту;
- Update() – метод, для оновлення існуючого проекту;
- Delete () – метод, для видалення існуючого проекту.

У кожному методі, викликається відповідний метод сервісу “ProjectService”.

Для методу GetNamesList(), викликається метод GetProjectsNamesList(), в якому з бази даних вибираються всі проекти, посортовані по імені та статусі. Для того, щоб зробити правильне сортування, використано метод Concat(), який дозволяє об’єднати запити.

Для методу GetCountriesList(), викликається метод GetCountriesList(), в якому отримуютється, створений раніше, масив країн.

Для методу GetIndustriesList(), викликається метод GetIndustriesList(), в якому отримуютється, створений раніше, масив індустрій.

Для методу GetListi(), викликається метод GetListByFilter(), в якому використовується пагінація через вхідну модель. Спочатку отримуємо масив проектів з бази даних, відсортовані по даті створення та типу. Після чого, робиться вибірка з масиву, для кожної властивості вхідної моделі, задається можливість бути null, якщо ж вона є, то фільтрується по ній. Далі пропускається необхідна кількість елементів, та для тих, що потрібно повернути, відбувається створення моделі “ProjectViewModel”, та задаються додаткові параметри про них, за допомогою методу колекції Count().

Для методу Get(), викликається метод GetNotDeletedProject(), в якому за допомогою методу GetProjectById(), отримується проект, а за допомогою GetProjectViewModelByProject(), формується модель “ProjectViewModel”. При формуванні моделі використовуються методи GetUserName(), GetAllTechnologyByProjectId(), GetProjectMembersInfo().

Метод GetUserName(), приймає параметром Id користувача і отримує з бази даних його назву.

Метод GetAllTechnologyByProjectId(), приймає параметром Id проекту і отримує з бази даних, назви технологій які використовуються.

Метод GetProjectMembersInfo(), приймає параметром Id проекту і якщо на проекті є користувачі, за допомогою методу GetUserIdListFromTeamByRole(), вибираю Id всіх звичайних користувачів типу “Member”. Після чого формує потрібну модель даних “ProjectUserModel”.

Для методу Create(), викликається метод CreateProject(), який приймає параметром вхідну модель “CreateProjectIncomeModel” та Id користувача. В методі, в базі даних створюється проект, та фіксуються зміни. Після успішного створення самого проекту, відбувається створення його команди, за допомогою методу CreateTeamForProject(). В методі, за допомогою методу CreateTeamEntity(), сервісу “TeamService”, відбувається створення команди, до якої додаються відповідні користувачі, з їх ролями, за допомогою методу AddUserInProjectTeam(), сервісу “TeamUserService”. Після створення проекту та команди, команда повідомляється про їх залучення до проекту, за допомогою сповіщень та надсилання пошти.

Для методу Update (), викликається метод UpdateProject(), який приймає параметром вхідну модель “UpdateProjectIncomeModel” та Id користувача. В методі, з бази даних отримуємо проект і перевіряємо, чи він уже створений. Далі, за допомогою методу CheckUserRightsForModifyingProject(), перевіряю права на оновлення проекту. Якщо це суперадмін, або користувач з встановленою на стадії планування роллю на проекті то дозволяю оновлення.

Після цього, проводимо перевірку на існування команди даного проекту, і якщо вона не існує в базі, генерується помилка, за допомогою класу “ObjectExistenceChecker”. Далі, відбувається оновлення даних проекту з вхідної моделі, та фіксація змін. Найважчим моментом, було реалізувати оновлення команди. Для цього, було створено метод UpdateUsersOnProjects(), в якому з бази даних обираються всі користувачі на даному проекті. Ці користувачі розділяються на звичайну команду та масив AM-ів, PM-ів. За оновлення звичайної моделі, відповідає метод UpdateUsersInTeam(). В ньому отримуються id користувачів, які внесені в базу даних, та Id користувачів команди, з моделі для оновлення. За допомогою методу колекції Except(), знаходиться різність

вибірок, тобто користувачів, яких необхідно видалити. Далі, починається перебір цих користувачів, і якщо вони не були занесені в базу даних, як вже видалені, встановлюються даті видалення та даті останньої модифікації, теперішній час. За оновлення одного користувача відповідає метод AddUserToTeamUsers(), який приймає в параметрах Id користувача, якого потрібно оновити або створити, його роль в команді та список користувачів. В методі, є блоки умови і в першому, повертається “Guid.Empty”, в другому, якщо користувач уже був в команді, встановлюється нова роль, дату модифікації, та занесення полю “DeleteAt” значення null. В іншому випадку, створюється новий учасник команди за допомогою методу CreateTeamUser(), додається в базу та фіксуються зміни. В кожному з блоків повертаються Id користувача, якого оновлювали. Отже, метод AddUserToTeamUsers(), я використовую для керівника команди та звичайних учасників, заносячи Id які повертаються в попередньо створений масив “newUsers”. Також оновлюються AM та PM, і так само добавляються в відповідний масив. Після оновлення команди, визначаються користувачі, які були видалені, додані, та ті, що залишились без змін. Для кожного з цих масивів користувачів, створено окремий блок умови, в якому вибирається відповідний текст з класів “NotificationTextResources”, “EmailNotificationResources” та надсилаються сповіщення, почтові повідомлення.

Для методу Delete(), викликається метод DeleteProject(), в якому отримується проект та перевіряється його наявність базі. Після цього, в методі GetAllUsersEmailsByProject(), отримується Email всіх користувачів на проекті, та за допомогою методу GetAllUsersOnProject(), самих користувачів. Ні сам проект, ні команда, фактично не видаляються. Для них встановлюється дата видалення теперішнім часом, а користувачів повідомляється через сповіщення та поштове повідомлення.

3.10. Додатковий функціонал. Міграції та Sead метод

При роботі з EF через підхід Code first, який було обрав, ми можемо

оновлювати нашу базу данних за допомогою міграцій. Після команди в консолі менеджера пакетів “update-database”, яка оновлює структуру таблиць відповідно до моделей, запускається метод Sead(), який проводить певні зміни з данними в таблицях. Тому для того, щоб при першому створенні бази данних, були автоматично занесені такі данні як Claims, Roles, Skills, Specialties та створення акаунту суперадміна, створюємо методи які відповідатимуть за їх занесення.

В методі AddOrUpdateClaims(), доступний список Claims, які повинні бути автоматично занесены в базу. Спочатку в методі витягуються всі Claims з бази данних, і перевіряє чи співпадають вони зі списком. Ті данні, яких немає в базі, заносяться в неї зі списку.

В методі CreateRoles(), добавляються ролі User, Admin, SuperAdmin.

В методі CreateSuperAdmin(), добавляється акаунт суперадміна, і після чого до цього користувача приєднюється роль SuperAdmin, та добавляються всі доступні Claims.

В методі SetSpecialtiesAnsSkills(), добавляються зі списку спеціальності та навички, які не занесені до бази данних.

В методі SetSystemSetting(), добавляються початкові системні настройки.

Фонові задачі

Для реалізації фонових задач на платформі .NET, було обрано Quartz.NET. Quartz.NET, представляє відкритий фреймворк для виконання дій зарозкладом, в середовищі ASP.NET. Для початку було додано Nuget-пакет, який називається по імені фреймворка. Треба визначити тригер, який буде керувати виконанням роботи, запускати її та конфігурувати. Клас роботи, повинен реалізувати інтерфейс IJob, який визначає метод Execute (), власне виконує деяку роботу. Потім створюємо саму роботу, яка буде виконуватися у вигляді об'єкта IJobDetail.

Кілька слів з приводу можливих налаштувань відправки. Замість хвилинного інтервалу виконання ми можемо поставити ще ряд інших:

- WithInterval (milliseconds): інтервал виконання в мілісекундах;

- WithIntervalInSeconds (seconds): інтервал в секундах;
- WithIntervalInHours (hours): інтервал в годинах;
- WithRepeatCount (number): визначає кількість повторів.

Момент запуску, який задається в тригері викликом StartNow (), такожмає різні варіанти:

- StartNow (): запуск відразу ж після початку виконання;
- StartAt (): визначає час, коли тригер починає запускати роботу;
- EndAt (): визначає час, коли тригер перестає запускати роботу.

І щоб робота почала виконуватися за розкладом зі стартом додатки,змінив клас Global.asax.cs.

Cron-Expressions - це рядки, які фактично складаються із семи підвиразів, що описують окремі деталі розкладу. Ці суб-вирази відокремлені пробілом та являють собою:

- секунди;
- хвилини;
- години;
- день місяця;
- місяць;
- день тижня;
- рік (необов’язкове поле).

За допомогю описаної вище інстукції, я створив в класі “Scheduler” наступні “Cron” операції:

MoveUsersNotUsedVacationDaysJob() – операція яка виконується першого вересня кожного року. В ній викликається метод MoveUsersNotUsedVacationDays(), в якому для всіх користувачів онуляються поля, які відповідають за нарахування відпусток, а не використані відпустки переносяться в поле кількості відпусток відпусток за минулий рік.

CloseProjects() – операція яка виконується кожного дня о 8 годині. В ній

викликається метод CloseProjectsEndTime(), в якому з бази даних отримую всі проекти, в яких дата закінчення стоїть теперішньою датою, та кожному виставляю статус Finished.

CreateNotificationToCloseProject() – операція яка виконується кожногодня о 8 годині. В ній викликається метод CreateNotificationToCloseProject(), в якому отримуємо дату, на місяць меншу ніж теперішній час, та вибираються проекти, в яких від цієї дати не логувалися години. Для оптимізації запиту я використовую підзапит до таблиці з таймлогами. Якщо отриманий масив проектів не порожній, починаю їх перебір в методі foreach(), для кожного створюючи сповіщення з просьбою закрити проект.

CloseAllProjects – операція яка виконується кожного дня о 8 годині. В ній викликається метод CloseProjects(), в якому спочатку отримуємо дату, на місяць меншу ніж теперішній час, та вибираються проекти, в яких від цієї дати не логувалися години. Для оптимізації запиту, було використано підзапити до таблиці з таймлогами. Якщо отриманий масив проектів не порожній, відбувається їх перебір в методі foreach(). В ньому, за допомогою блоків умов, перевідяється наявність менеджера та акаунт менеджера, створюючи для них сповіщення та надсилаю повідомлення на пошту про закриття проекту. Для цього використовую розроблені раніше відповідні методи. В кінці кожної ітерації, зазначаю статус проекту Finished, та дату закриття встановлюю як дату в момент виконання.

UserOnboarding() – операція яка виконується кожного дня о 5 годині. В ній викликається метод UserOnBoarding(), в якому з бази даних вибираються користувачі, в яких на дану дату закінчується випробувальний термін. Якщо такі користувачі є, для кожного встановлюю дату закінчення випробувального як null, а буліанівському значенню, яке відповідає за те, чи є користувач на випробувальному встановлюю як false.

UpdateUsersVacation() – операція яка виконується кожного дня о 6 годині.В ній викликається метод SetAllUsersVacation(), в якому для всіх користувачів викликається описаний раніше метод нарахування днів відпусток.

Birthday() – операція яка виконується кожного дня о 10 годині. В ній викликається метод SendHappyBirthday(), в якому з бази даних отримуються всі користувачі, в яких день і місяць дня народження, співпадають з теперішньою датою. Для кожного, використовуючи розроблені раніше методи, створюю сповіщення та надсилаю привітання на пошту.

StartNotifyAboutEndOfTrialPeriod() – операція яка виконується кожного дня о 9 годині. В ній викликається метод NotifyAboutEndOfTrialPeriod(), в якому створюються сповіщення та відбувається надсилання поштових повідомлень користувачам, в яких сьогодні закінчується випробувальний термін.

CreateNotifcationTeammateOnVacation() – операція яка виконується кожного дня о 21 годині. В ній викликається метод NotifyAboutTeammateOnVacation(), в якому з бази отримуються користувачі, та перевіряю, чи є в них погоджена відпустка, в якої стартова дата є завтрашнім числом. Якщо є, отримую Id користувачів, які знаходяться з даним в одній команді на проекті, і кожному надсилаю сповіщення з потрібним текстом.

Розділ 4. ТЕСТУВАННЯ

4.1. Вхід та реєстрація

Розроблена веб-застосунок - це вибрана методика роботи. Для його розробки був використаний програмний комплекс, що базується на веб-технологіях. Тому він без проблем працює в актуальних браузерах і відповідає всім веб-стандартам.

Інсталяція та системні вимоги

Щодо інсталяції та системних вимог - на щастя, він не потребує встановлення на користувацький пристрій. Але все життєво важливий є веб-браузер, що підтримує актуальні веб-стандарти. Також необхідно мати стабільний доступ до інтернету, зі швидкісним діапазоном не менше 50 мбіт за секунду. Для того, щоб мати можливість проводити тести та опитування в рамках програми, клієнтський додаток мусить мати відповідний дозвіл.

Інструкція з використання програмного продукту

Коли користувач заходить на клієнтський додаток, щоб працювати в системі, йому необхідно авторизуватися. Щоб зробити це, він мусить успішно пройти процедуру авторизації в системі. Рисунок 4.1 демонструє вигляд сторінки авторизації користувача.

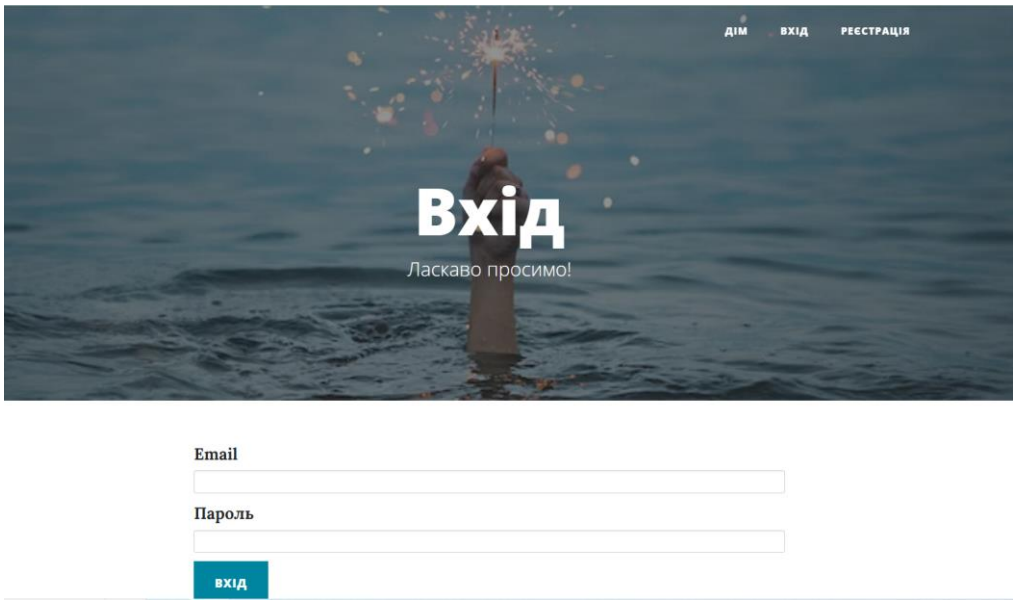

Рисунок 4.1 — Форма авторизації

У випадку якщо користувач ще ніколи не користувався системою або хоче створити новий профіль, то йому необхідно зареєструватися в системі. На рисунку 4.2 зображено сторінку реєстрації.

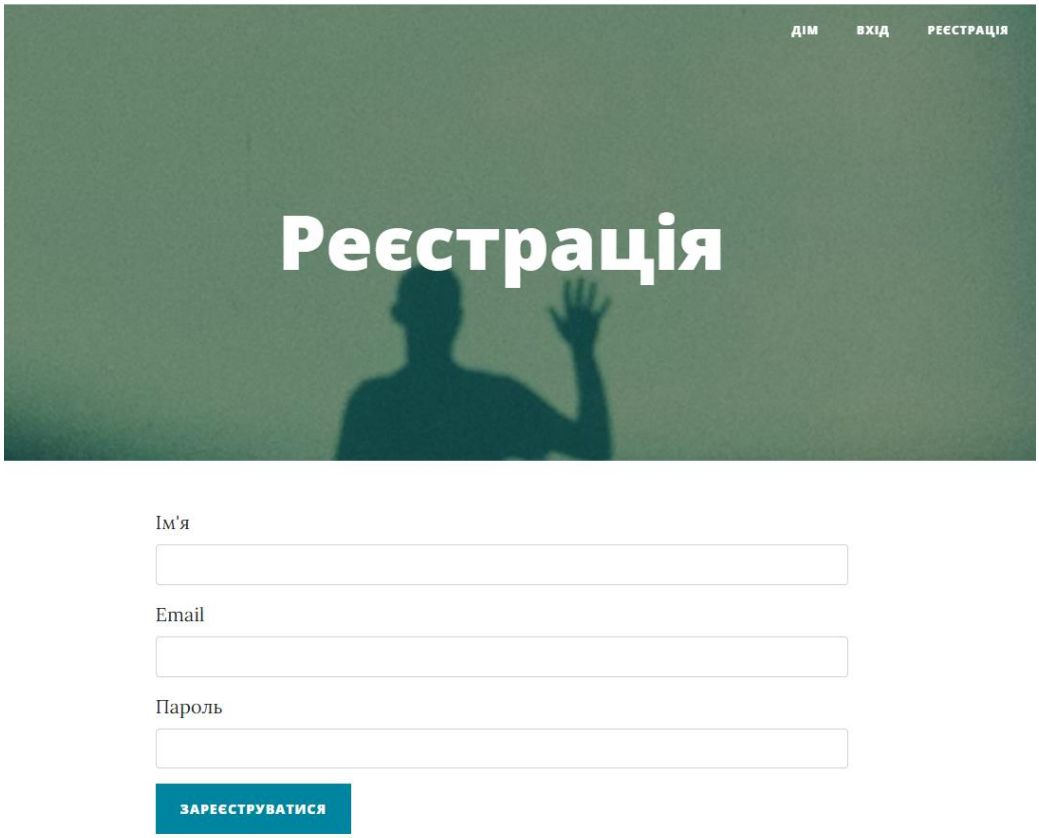

Рисунок 4.2 — Форма реєстрації

Після того, як користувач авторизувався в системі, він отримує доступ до головного меню системи. За допомогою цього меню користувач має доступ до усіх опитувальників системи, або має можливість вийти з свого профілю.

4.2. Список питань та тести

На головній сторінці користувач має доступ до всіх тестів, які були створені адміністратором. На рисунку 4.3 зображено основне меню мого сайту.

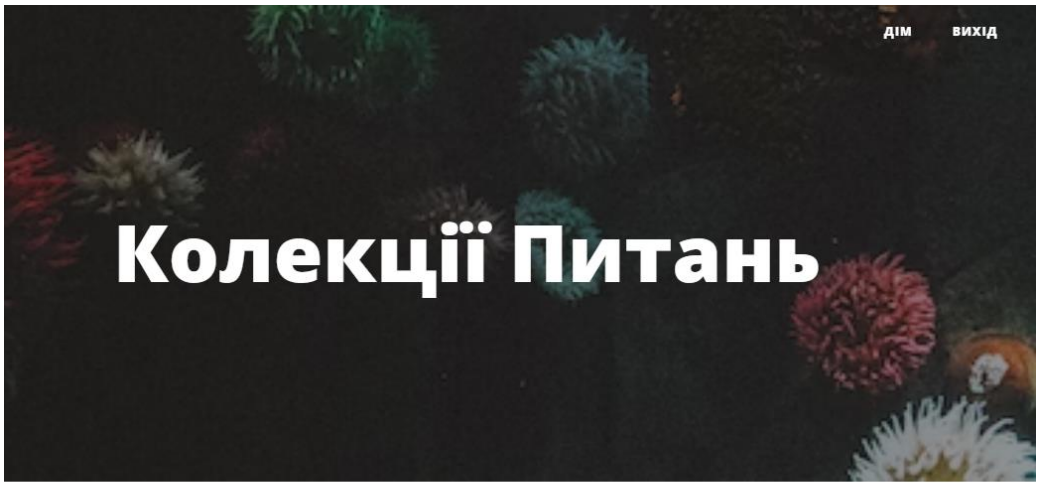

Рисунок 4.3 — Головне меню системи

При переходжені у вибраний тест, нас зустрічає основна інформація про тест та можливість переглянути наші відповіді на цей тест. На рисунку 4.4 зображено головну сторінку тесту.

Рисунок 4.4 — Сторінка тесту

Відповіді даються в відкритій формі. На рисунку 4.5 зображено сторінку з питанням.

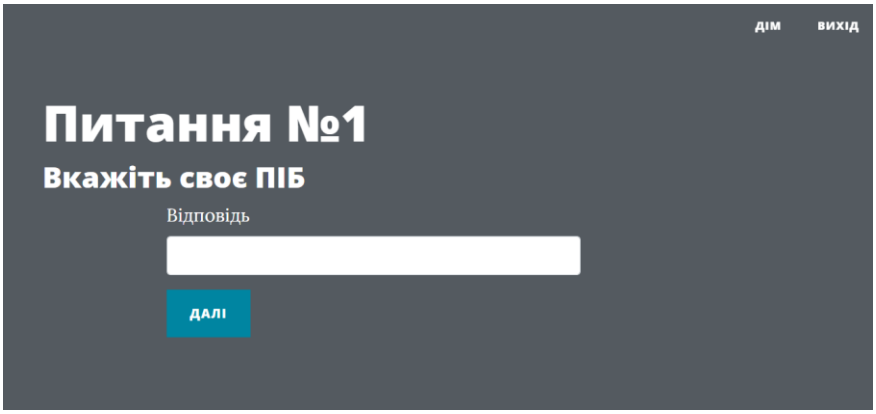

Рисунок 4.5 — Форма відповіді

Після того як відповіли на всі запитання відображається сторінка подяки.

На рисунку 4.6 зображено сторінку подяки яка відображається після проходження кожного тесту.

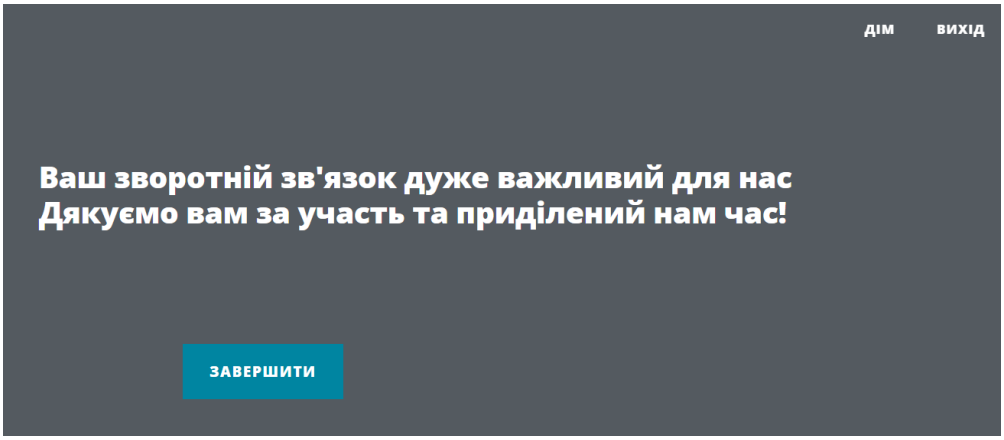

Рисунок 4.6 — Сторінка подяки

Можемо переглянути свої відповіді в любий момент. На рисунку 4.7 зображено таблицю відповідей користувача.

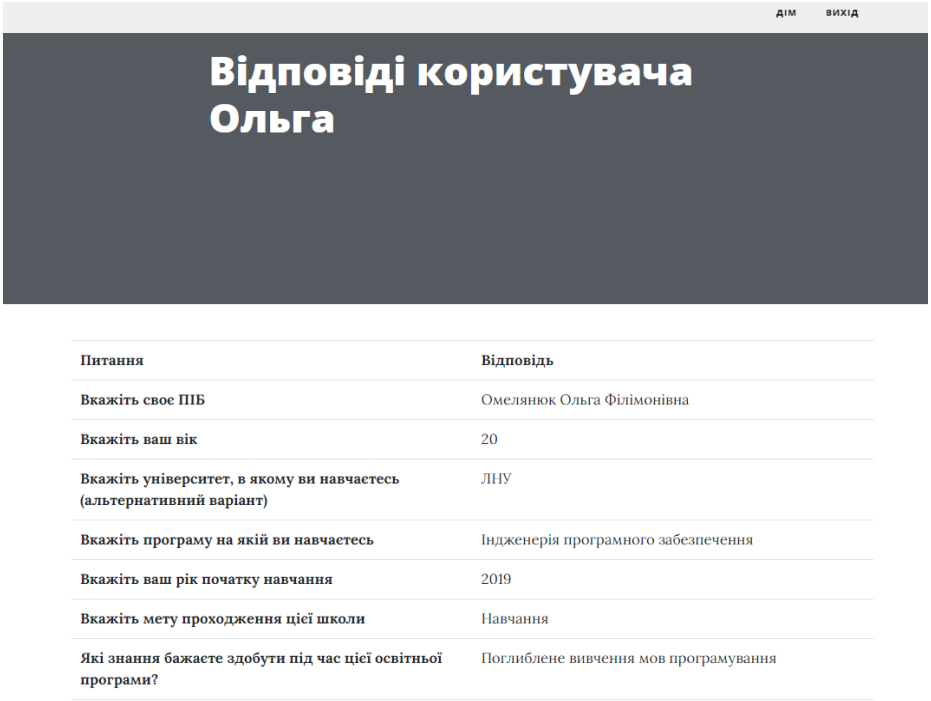

Рисунок 4.7 — Сторінка перегляду відповідей

При вході з даними Email: admin123@gmail.com Пароль:1;

Основне меню має кнопку створення опитувальника. На рисунку 4.8 зображено основну сторінку сайту користувача-адміністратора та основні відмінності від звичайної сторінки.

4.3. Меню адміністратора

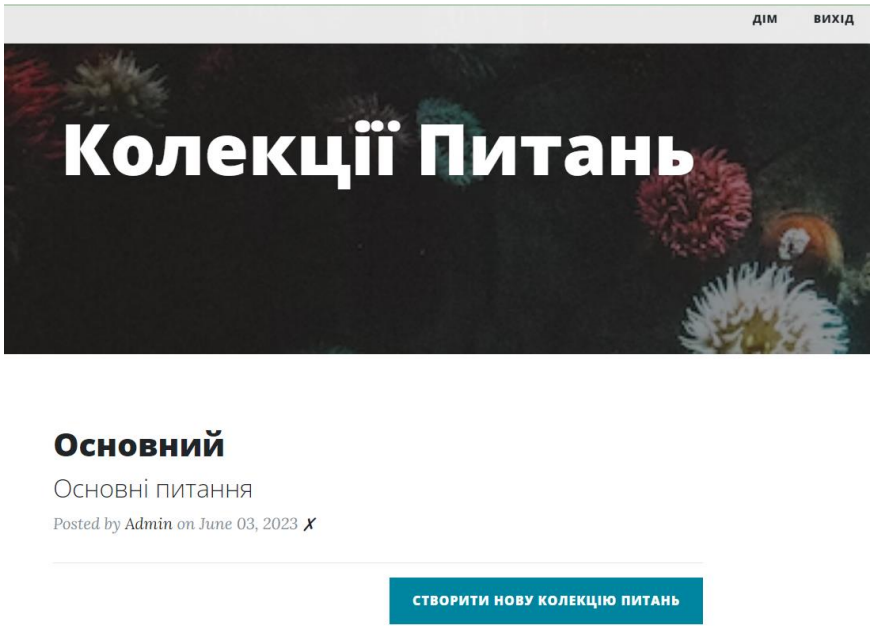

Рисунок 4.8 — Головне меню адміністратора

При створенні тесту адміністратор спочатку вводить основну інформацію, а також функціонально URL зображення яке буде у головному меню тесту.

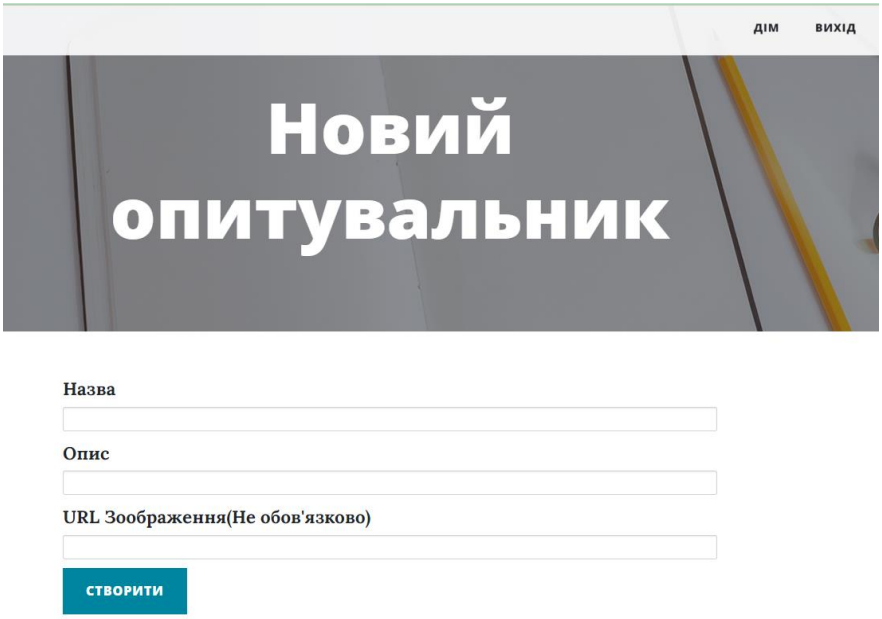

Рисунок 4.9 — Сторінка створення тесту

Після створення адміністратор додає питання до тесту. Після завершення додавання всіх питань натискає кнопку готово, після чого даний тест відображається у головному меню для всіх користувачів. На рисунку 4.10 зображено сторінку та форму за допомогою якої можна додавати питання до новоствореного блоку питань.

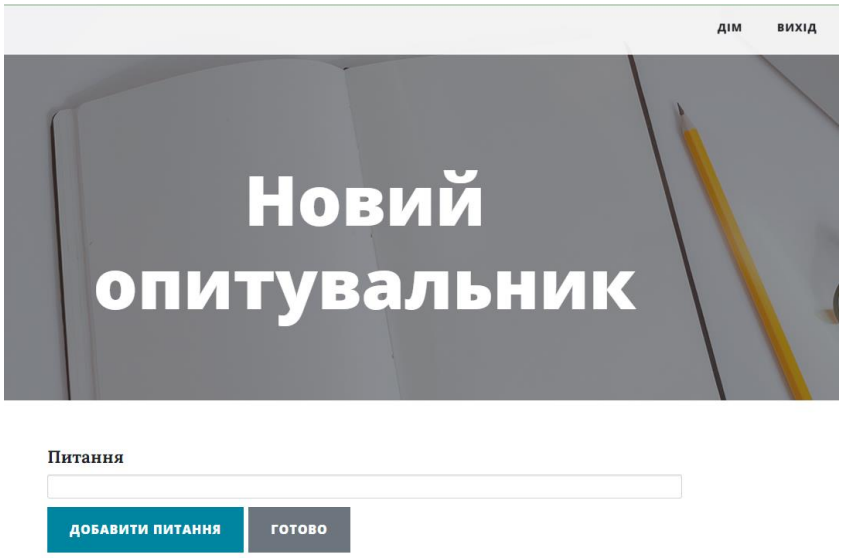

Рисунок 4.10 — Сторінка додавання питань до тесту

Також адміністратор може видалити тест та переглядати відповіді усіх користувачів які пройшли тест використовуючи ту ж кнопку що і користувачі. Але на відміну від версії користувача, адміністратор потрапляє на сторінку з таблицею користувачів, після чого адміністратор має змогу вибрати чиї відповіді оглядати. На рисунку 4.11 зображено таблицю користувачів, які проходили даний тест.

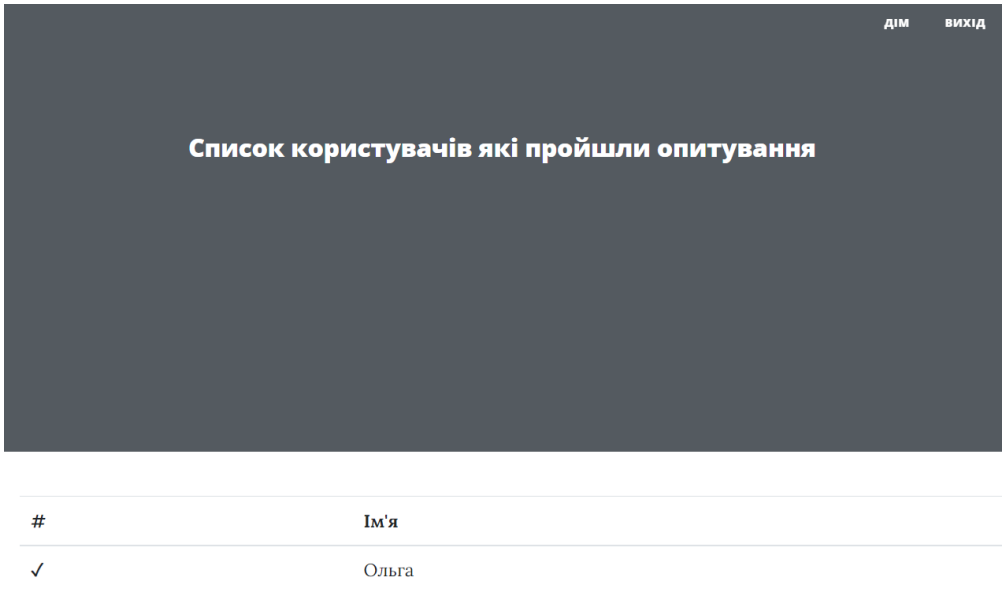

Рисунок 4.11 — Таблиця користувачів які пройшли тест

ВИСНОВКИ

Під час аналізу наявного програмного забезпечення для тестування та опитування в навчальному процесі, були вивчені різні системи, які можуть вирішувати дані задачі.

На основі цього аналізу було створено програмний продукт, який є засобом для автоматизації процесу тестування та опитування, забезпечуючи швидку та ефективну роботу.

Були розглянуті методи та засоби для розробки програмної системи, аргументація вибору веб-технологій та триланкової архітектури. Це дозволяє збільшити гнучкість та зручність системи під час розробки, супроводу та використання.

В результаті тестувань було підтверджено коректність отриманих результатів, що обгрунтовує відповідність системи поставленим вимогам.

Розроблена система призначена для використання викладачами, студентами та адміністраторами. Викладачі можуть створювати тести та опитування, студенти – їх проходити, а адміністратори – змінювати інформацію про користувачів.

Програмне забезпечення може бути встановлено на будь-якій операційній системі з браузером, що підтримує останні веб-стандарти та має постійний доступ до інтернету.

Під час розробки було використано різноманітні технології, що дозволило покращити їх розуміння. Також були створені декілька прототипів програмного забезпечення, спрямованих на вирішення різних аспектів поставленої в дослідженні задачі. Вони послужили основою для розробки фінального продукту - розробленого програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Адам Фримен. ASP.NET Core MVCс примерами на C# дляпрофессионалов. 6-е издание. 2017р.
2. Роберт Мартин. Чистая архитектура. Искусство разработки программногообеспечения. 2018р.
3. Юрген Аппело Agile-менеджмент. Лидерство и управление командами.2018р.
4. M. Kozlenko and A. Bosyi, "Performance of spread spectrum system with noise shift keying using entropy demodulation," 2018 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications andComputer Engineering (TCSET), Slavske, 2018, pp. 330-333, doi: 10.1109/TCSET.2018.8336213.
5. Патерни. URL:https://metanit.com/sharp/patterns/
6. ASP.NET. URL: https://metanit.com/sharp/mvc5/
7. Google OAuth. URL: https://developers.google.com/identity/protocols/oauth2

ДОДАТОК А

Основний код програми

```
from flask import Flask, render_template, redirect,
url_for, flash, abort, session, request
from functools import wraps
from flask_bootstrap import Bootstrap
from flask_ckeditor import CKEditor
from datetime import date
from werkzeug.security import generate_password_hash,
check_password_hash
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy.orm import relationship
from flask_login import UserMixin, login_user,
LoginManager, login_required, current_user, logout_user
from forms import CreatePostForm, NewUserForm, Login,
CreateQuestion, START, GiveAnswer, END
from flask_gravatar import Gravatar

app = Flask(__name__)
app.config['SECRET_KEY'] =
'8BYkEfBA6O6donzWlSihBXox7C0sKR6b'
Bootstrap(app)
login_manager = LoginManager()
login_manager.init_app(app)


@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))


##CONNECT TO DB
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///blog.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)


##CONFIGURE TABLES


class User(UserMixin, db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String, nullable=False)
    email = db.Column(db.String, nullable=False,
unique=True)
    password = db.Column(db.String, nullable=False)
    user = db.relationship('BlogPost', backref='user')

class BlogPost(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    author = db.Column(db.String(250), nullable=False)
    title = db.Column(db.String(250), unique=True,
nullable=False)
    subtitle = db.Column(db.String(250), nullable=False)
    date = db.Column(db.String(250), nullable=False)
    img_url = db.Column(db.String(250), nullable=False)
    user_id = db.Column(db.Integer,
db.ForeignKey('user.id'))

class Questions(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    blog_id = db.Column(db.Integer,
db.ForeignKey('blog_post.id'))
    text = db.Column(db.Text, nullable=False)

class Answers(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String, nullable=False)
    blog_id = db.Column(db.Integer, nullable=False)
    text_questions = db.Column(db.Text, nullable=False)
    text_answer = db.Column(db.Text, nullable=False)

with app.app_context():
    db.create_all()

def admin_only(function):
    @wraps(function)
    def wrapper(*args, **kwargs):
        if not '_user_id' in session.keys():
            flash('Page locked, login as admin.')
            return redirect(url_for('login'))
        if int(session['_user_id']) == 1:
            return function(*args, **kwargs)
        else:
            abort(403)
    return wrapper
```

```
@app.route('/')
def get_all_posts():
    posts = BlogPost.query.all()
    return render_template("index.html", all_posts=posts,
logged=current_user.is_authenticated)

@app.route('/check/<int:post_id>')
@admin_only
def check_table(post_id):
    questions = Answers.query.filter_by(blog_id=post_id)
    names = set()
    for question in questions:
        names.add(question.name)
    return render_template('check_answ.html',
logged=current_user.is_authenticated, names=names,
post_id=post_id, users=User())

@app.route('/check/<int:post_id>/<int:user_id>')
def check(post_id, user_id):
    user = User.query.filter_by(id=user_id).first()
    answer_user = Answers.query.filter_by(name=user.name)
    answer_post = []
    for x in answer_user:
        if x.blog_id == post_id:
            answer_post.append(x)
    print(answer_post)
    return render_template('check.html',
logged=current_user.is_authenticated, user=user,
answer=answer_post)

@app.route('/register', methods=['GET', 'POST'])
def register():
    form = NewUserForm()
    if form.validate_on_submit():
        email = form.email.data
        if User.query.filter_by(email=email).first():
            flash('You are already registered with this
email. Come in.')
            return redirect(url_for('login'))
        new_user = User(
            name=form.name.data,
            email=email,

password=generate_password_hash(form.password.data,
salt_length=8))
```

```
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('login'))
    return render_template("register.html", form=form,
logged=current_user.is_authenticated)


@app.route('/login', methods=['GET', 'POST'])
def login():
    form = Login()
    user =
User.query.filter_by(email=form.email.data).first()
    if form.is_submitted():
        if user:
            if check_password_hash(user.password,
form.password.data):
                login_user(user)
                return redirect(url_for('get_all_posts'))
            else:
                flash('Не вірний email або пароль')
                return redirect(url_for('login'))
        else:
            flash('Користувач не знайдений')
            return redirect(url_for('login'))
    return render_template("login.html", form=form,
logged=current_user.is_authenticated)


@app.route('/logout')
def logout():
    logout_user()
    return redirect(url_for('login'))


@app.route("/post/<int:post_id>", methods=['GET',
'POST'])
def show_post(post_id):
    if not current_user.is_authenticated:
        flash('Спочатку увійди')
        return redirect(url_for('login'))
    length_questions =
len(Questions.query.filter_by(blog_id=post_id).all())
    form = START()
    if form.validate_on_submit():
        return redirect(url_for('give_answers',
```

```
post_id=post_id, question_id=0,
logged=current_user.is_authenticated))
    requested_post = BlogPost.query.get(post_id)
    return render_template("post.html",
post=requested_post, form=form, len=length_questions,
l=current_user, logged=current_user.is_authenticated)

@app.route("/post/<int:post_id>/<int:question_id>",
methods=['GET', 'POST'])
def give_answers(post_id, question_id):
    questions =
Questions.query.filter_by(blog_id=post_id).all()
    if question_id+1 > len(questions):
        return redirect(url_for('end'))
    question = questions[question_id]
    form = GiveAnswer()
    if form.validate_on_submit():
        print(post_id)
        new_answer = Answers(name=current_user.name,

text_questions=question.text,
                             blog_id=post_id,
                             text_answer=form.answer.data
                             )
        db.session.add(new_answer)
        db.session.commit()
        return redirect(url_for('give_answers',
post_id=post_id, question_id=question_id+1))
    print(question)
    return render_template('questions.html', form=form,
logged=current_user.is_authenticated, question=question,
                           N=question_id+1)


@app.route('/post/end', methods=['GET', 'POST'])
def end():
    form = END()
    if form.validate_on_submit():
        return redirect(url_for('get_all_posts'))
    return render_template('End.html', form=form,
logged=current_user.is_authenticated)


@app.route("/new-post", methods=['GET', 'POST'])
@admin_only
```

```
def add_new_post():
    form = CreatePostForm()
    if form.validate_on_submit():
        new_post = BlogPost(
            title=form.title.data,
            subtitle=form.subtitle.data,
            img_url=form.img_url.data,
            author=current_user.name,
            date=date.today().strftime("%B %d, %Y"),
            user_id=current_user.id
        )
        db.session.add(new_post)
        db.session.commit()
        return redirect(url_for("add_new_question"))
    return render_template("make-post.html", form=form,
logged=current_user.is_authenticated)

@app.route("/new-question", methods=['GET', 'POST'])
@admin_only
def add_new_question():
    form = CreateQuestion()
    if 'chancel' in request.form:
        form.question.validators = []
        return redirect(url_for('get_all_posts',
logged=current_user.is_authenticated))
    if form.validate_on_submit():
            last_blog_post =
BlogPost.query.order_by(BlogPost.id.desc()).first()
            new_question = Questions(
                text=form.question.data,
                blog_id=last_blog_post.id
            )
            db.session.add(new_question)
            db.session.commit()
            return redirect(url_for('add_new_question',
form=form, logged=current_user.is_authenticated))
    return render_template("create_question.html",
form=form, logged=current_user.is_authenticated)



@app.route("/delete/<int:post_id>")
@admin_only
def delete_post(post_id):
    post_to_delete = BlogPost.query.get(post_id)
```

```
    question =
Questions.query.filter_by(blog_id=post_id).all()
    for q in question:
        db.session.delete(q)
    db.session.delete(post_to_delete)
    db.session.commit()
    return redirect(url_for('get_all_posts'))


if __name__ == "__main__":
    app.run(debug=True)
```

HTML-код головної стрінки

```
{% include "header.html" %}

  <!-- Page Header -->
  <header class="masthead" style="height: 480px; background-image:
url('https://images.unsplash.com/photo-1470092306007-
055b6797ca72?ixlib=rb-1.2.1&auto=format&fit=crop&w=668&q=80')">
    <div class="overlay"></div>
    <div class="container">
      <div class="row">
        <div class="col-lg-8 col-md-10 mx-auto">
          <div class="site-heading">
            <h1>Колекції Питань</h1>
            <span class="subheading"></span>
          </div>
        </div>
      </div>
    </div>
  </header>

  <!-- Main Content -->
  <div class="container">
    <div class="row">
      <div class="col-lg-8 col-md-10 mx-auto">
        {% for post in all_posts %}
        <div class="post-preview">
          <a href="{{ url_for('show_post', post_id=post.id) }}">
            <h2 class="post-title">
              {{post.title}}
            </h2>
            <h3 class="post-subtitle">
              {{post.subtitle}}
            </h3>
```

```
          </a>
          <p class="post-meta">Posted by
            <a href="#">{{post.author}}</a>
            on {{post.date}}
            {% if current_user.id == 1 %}
              <a href="{{url_for('delete_post', post_id=post.id) }}">✘</a>
            {% endif %}
          </p>
        </div>
        <hr>
        {% endfor %}

        <!-- New Post -->
        {% if current_user.id == 1 %}
          <div class="clearfix">
            <a class="btn btn-primary float-right"
href="{{url_for('add_new_post')}}">Створити нову колекцію питань</a>
          </div>
        {% endif %}
      </div>
    </div>
  </div>
  <hr>

{% include "footer.html" %}
```

Html-код header

```
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-
to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>Мій Опитувальник</title>

  <!-- Bootstrap core CSS -->
  <link href="{{ url_for('static',
filename='vendor/bootstrap/css/bootstrap.min.css') }}" rel="stylesheet">

  <!-- Custom fonts for this template -->
```

```
  <link href="{{ url_for('static', filename='vendor/fontawesome-
free/css/all.min.css')}}" rel="stylesheet" type="text/css">
  <link
href='https://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic
' rel='stylesheet' type='text/css'>
  <link
href='https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,
600italic,700italic,800italic,400,300,600,700,800' rel='stylesheet'
type='text/css'>

  <!-- Custom styles for this template -->
  <link href="{{ url_for('static', filename='css/clean-blog.min.css')}}"
rel="stylesheet">

</head>

<body>

  <!-- Navigation -->
  <nav class="navbar navbar-expand-lg navbar-light fixed-top" id="mainNav">
    <div class="container">
      <a class="navbar-brand" href="{{url_for('get_all_posts')}}">
      </a>
      <button class="navbar-toggler navbar-toggler-right" type="button" data-
toggle="collapse" data-target="#navbarResponsive" aria-
controls="navbarResponsive" aria-expanded="false" aria-label="Toggle
navigation">
        Menu
        <i class="fas fa-bars"></i>
      </button>
      <div class="collapse navbar-collapse" id="navbarResponsive">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item">
            <a class="nav-link" href="{{ url_for('get_all_posts') }}">Дім</a>
          </li>
          {% if not logged %}
          <li class="nav-item">
            <a class="nav-link" href="{{ url_for('login') }}">Вхід</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{{ url_for('register') }}">Реєстрація</a>
          </li>
          {% endif %}
          {% if logged %}
```

```
          <li class="nav-item">
            <a class="nav-link" href="{{ url_for('logout') }}">Вихід</a>
          </li>
          {% endif %}
        </ul>
      </div>
    </div>
  </nav>
```

Html-код footer

```
  <!-- Footer -->
  <footer>
    <div class="container">
      <div class="row">
        <div class="col-lg-8 col-md-10 mx-auto">
          <ul class="list-inline text-center">
            <li class="list-inline-item">
              <a href="#">
                <span class="fa-stack fa-lg">
                  <i class="fas fa-circle fa-stack-2x"></i>
                  <i class="fab fa-twitter fa-stack-1x fa-inverse"></i>
                </span>
              </a>
            </li>
            <li class="list-inline-item">
              <a href="#">
                <span class="fa-stack fa-lg">
                  <i class="fas fa-circle fa-stack-2x"></i>
                  <i class="fab fa-facebook-f fa-stack-1x fa-inverse"></i>
                </span>
              </a>
            </li>
            <li class="list-inline-item">
              <a href="#">
                <span class="fa-stack fa-lg">
                  <i class="fas fa-circle fa-stack-2x"></i>
                  <i class="fab fa-github fa-stack-1x fa-inverse"></i>
                </span>
              </a>
            </li>
          </ul>
          <p class="copyright text-muted">Copyright &copy; Your Website
2023</p>
        </div>
```

```
      </div>
    </div>
  </footer>

  <!-- Bootstrap core JavaScript -->
  <script src="{{ url_for('static',
filename='vendor/jquery/jquery.min.js')}}"></script>
  <script src="{{ url_for('static',
filename='vendor/bootstrap/js/bootstrap.bundle.min.js')}}"></script>

  <!-- Custom scripts for this template -->
  <script src="{{ url_for('static', filename='js/clean-blog.min.js')}}"></script>

</body>

</html>
```