Форма № Н-9.02

Вінницький технічний фаховий коледж
(повне найменування вищого навчального закладу)

Комп'ютерне відділення
(повне найменування інституту, назва факультету (відділення))

Інформатики та інформаційних комп'ютерних технологій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломного проекту
молодшого спеціаліста
(освітньо-кваліфікаційний рівень)

на тему *Менеджер обліку працівників*

Виконав: студент 4 курсу, групи 4ОК2
напряму підготовки 12 Інформаційні технології, спеціальності 123 комп'ютерна інженерія
Костюк Р.С.
(прізвище та ініціали)

Керівник *Стець А.М.*
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)

Вінниця 2022

Анотація

В процесі роботи над завданням дипломного проекту реалізовано веб-додаток «Менеджер обліку працівників», який планується використовувати в якості навчального матеріалу факультативних курсів з веб-програмування. Додаток спроектовано з використанням сучасних технологій та унікальних рішень розробки програмного забезпечення. У якості основи структури проекту використовувався архітектурний патерн REST, який забезпечує розділення логіки, візуальної частини та джерела даних на окремі реалізації.

Збереження інформації виконується за допомогою системи керування базами даних SQL Server Express, яка забезпечує з’єднання та взаємодію з самою базою даних.

Також проведено техніко-економічне обгрунтування доцільності розробки та проведені відповідні економічні розрахунки. У розділі безпеки життєдіяльності розглянуто організаційні вимоги щодо роботи за комп’ютером.

Annotation

In the process of working on the task of the diploma project, the web application "Workers Accounting Manager" was implemented, which is planned to be used as an educational material of optional courses in web programming. The application is designed using modern technologies and unique software development solutions. The architectural pattern REST was used as the basis of the project structure, which provides the division of logic, visual part and data source into separate implementations.

Information is stored using the SQL Server Expess database management system, which connects and interacts with the database itself.

Feasibility study of expediency of development is also carried out and the corresponding economic calculations are carried out. The section on life safety discusses the organizational requirements for working with a computer.

Зміст

Вступ

Вивчення клієнт-серверних технологій відіграє важливу роль у процесі формування інженера для сфери інформаційних технологій. Знання того, як створити правильне API, виконувати запити до сервера, обробляти дані та відсилати результати назад до клієнта є ключовими знаннями для створення сучасних веб-додатків.

В проекті «Менеджер обліку працівників» для серверної частини використовуються фреймфорк ASP.NET Core і система управлання базами даних під назвою SQL Server Express. Особливістю SQL Server Express є те, що вона використовує додаток LocalDB, який дає можливість розробнику працювати з базою даних, не переміщаючи її з сервера на ПК і навпаки. При з'єднанні необхідна інфраструктура SQL Server створюється та запускається автоматично, що дозволяє застосунку використовувати базу даних без виконання складного налаштування. Інструменти розробника дозволяють використовувати ядро СУБД SQL Server для створення та перевірки коду Transact-SQL без зобов'язань з управління повноцінним екземпляром SQL Server.

Під час створення додатку база даних була приєднана та використовується як файл mdf для можливості її використання на робочому ПК.

Для реалізації взаємодії проекту з базою даних використовується пряме підключення бази даних до проекту. Воно створює програмний зв’язок між проектом і файлом mdf. Це дає можливість додавати, змінювати та видаляти записи таблиць бази даних, що потребує використання SQL-запитів для роботи з таблицями.

Такий підхід зменшує накладні витрати, час створення і реалізацію програми.

1 Техніко-економічне обґрунтування дипломного проекту

1.1 Економічна доцільність розробки

З метою ефективної організації навчального процесу з вивчення сучасних засобів та технологій створення веб-застосунків в рамках відповідного факультативного курсу була поставлена задача розробки досить простого проекту, послідовність розробки та реалізації якого відповідала б орієнтовному тематичному плану курсу. Базовими для курсу є знання з основ програмування мовою C# та мовою взаємодії з базами даних SQL. При цьому тематика предметної області повинна бути актуальною і зрозумілою здобувачам освіти старшого шкільного віку.

Типовою предметною областю з вивчення основ веб-програмування є реалізація додатку веб-комерції. Проведений мною аналіз можливих варіантів програмних продуктів, які можна створити (простий інформаційний сайт, музичний сервіс, інтернет магазин тощо) та перегляд програмних проектів прикладів для слухачів курсів показали, що досить цікавим і універсальним з точки зору подальшого розширення функціоналу і застосування нових технологій є реалізація менеджеру обліку працівників. Ця область є досить актуальною і зрозумілою для людей, що знайомі з елементарними поняттями економіки та бізнесу, вони розуміють її призначення і можливі функції, які їм цікаво реалізовувати самостійно.

В базовому варіанті створення такого проекту потребує приблизно 20 годин. Це дає можливість покласти його в основу пятидесятигодиного курсу, враховуючи, що цей курс буде забезпеченим наявністю базових знань та детальними інструкціями, які знаходитимуться в мережі у вільному доступі. Тому основну роботу з реалізації модулів і збирання проекту слухачі можуть виконувати самостійно в дистанційному режимі, при потребі звертаючись до викладача.

Аудиторні години при цьому виділяються для аналізу альтернативних засобів розробки та проектних рішень, виконання спеціальних завдань з отримання базових навичок, обговорення можливих шляхів вдосконалення проекту тощо.

Створена мною програма являє собою веб-застосунок з такими категоріями як «Працівники», «Посади», «Оператори», «Панель приладів», «Рахунки». В проекті було реалізовано засоби однофакторної авторизації. Записи даних акаунтів користувачів зберігаються в окремій таблиці бази даних. Також реалізована обробка винятків, яка є критичним компонентом кожного програмного додатка.

Завдяки використанню фреймворка ASP.NET проект було реалізовано як симуляцію single-page application, також відомий як веб-застосунок чи веб-сайт, що вміщується на одній сторінці з метою забезпечення користувачу можливості використання досвіду користування настільною програмою.

В моєму застосунку весь необхідний код сайту – HTML та CSS – завантажується разом зі сторінкою, або динамічно довантажується за потребою, зазвичай у відповідь на дії користувача. Сторінка веде діалог за допомогою подій у відповідь на дії користувача. Сторінка не оновлюється і не перенаправляє користувача до іншої сторінки у процесі роботи з нею.

1.2 Обгрунтування та вибір аналогу

За аналог було взято сервіс "Monday" [1]. Цей продукт є схожим за функціоналом на той, що розробляється, і реалізований іншими мовами програмування, такими як JavaScript, NodeJS.

monday.com – сервіс з управління проектами, який допомагає організаціям керувати задачами, проектами та працювати над ними командами. Станом на 2020 рік компанія обслуговувала понад 100 000 організацій.

Monday надає користувачам можливість об’єднуватись в групи та працювати над спільним проектом, відслідковуючи прогрес створення, виконання задач. Також він може інтегруватись з сервісами-нагадуваннями, щоб користувач ніколи не пропускав головних подій. Система надає функціонал для реєстрації на сайті (рисунок 1) та інтерфейс для відслідковування прогресу (рисунок 2).

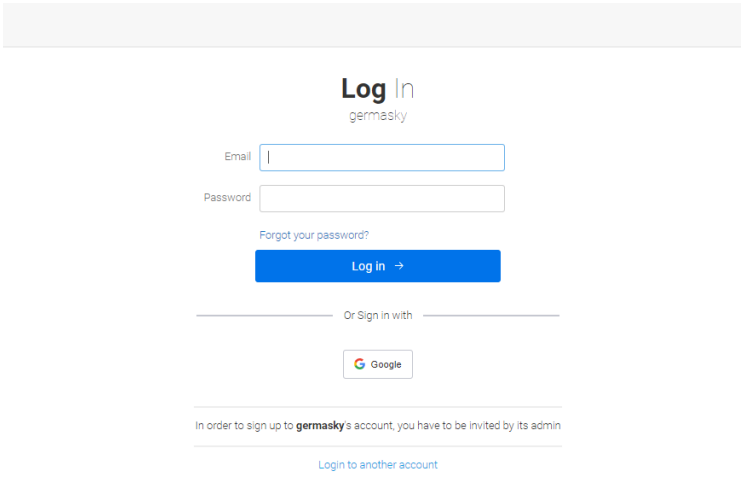

Рисунок 1 – Функціонал для реєстрації на сайті «monday.com»

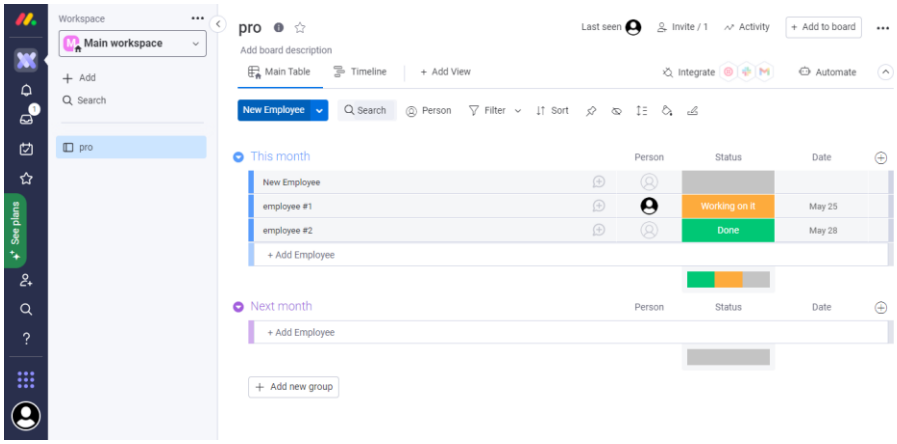

Рисунок 2 – Головне вікно «Monday»

Результати порівняння системи, що розроблена, та її аналогу продемонстровано в таблиці 1.1.

Таблиця 1.1 – Порівняльні характеристики нового продукту і аналогів

Особливість ПЗ	ПЗ, що розроблено	Monday
Зручність роботи	так	так
Дизайн інтерфейсу	так	так
Мова інтерфейсу	англійська	англійська
Можливість інтеграції з іншим ПЗ	ні	так
Можливість викостання в односторінковому режимі	так	так
Можливість створення звітності	так	ні
Необмежене використання	так	ні
Гнучкість системи	так	ні

1.3 Прогнозування попиту на розробку

Даний програмний продукт розроблено для подальшого створення на його базі навчального курсу. Переваги використання даної програми як наскрізного проекту або проекту-задачі мають велику ціну під процесу навчання або підготовки до перевірки знань слухачів курсів. Вивчення процесу створення подібних систем є цікавим та надає знання та вміння щодо реалізації тих чи інших рішень при створенні подібних програмних продуктів.

1.4 Загальний висновок про необхідність розробки

Виконаний техніко-економічний аналіз доцільності розробки, дозволяє зробити висновок про навчальну-методичну корисність (підвищення якості вивчення програмування в навчальних закладах) і локальну (можливість продажу, допомога компаніям в реалізації системи керування складу працівників) економічну ефективність розробки.

Також, провівши аналіз ринку, зроблено висновок, що в доступі знаходиться мала кількість подібного програмного продукту, який би задовільнив користувача за доступною ціною.

1.5 Постановка задач дослідження

Метою дипломного проекту було реалізація програмного продукту, що забезпечує можливість подальшого створення навчального курсу на його базі.

За результатами аналізу даних було сформульовано наступні вимоги до системи:

- реєстрація користувачів з метою отримання прав доступу;
- облік користувачів;
- безпека користувачів в системі;
- сповіщення про завершені події;
- зручне відображення інформації з таблиць БД;
- можливість реєстрації та запрошення нових користувачів;
- можливість додавання, редагування, видалення записів про посади та працівників.

Вирішення поставлених задач вимагало проведення комплексного дослідження усіх можливих рішень, розробки і реалізації оптимального архітектурного рішення.

2 Теоретична частина

2.1 Архітектура системи .NET Framework

.Net Framework – це платформа для розробки програмного забезпечення, розроблена Microsoft для створення та запуску програм Windows. Платформа .Net складається з інструментів розробника, мов програмування та бібліотек для створення настільних і веб-додатків. Він також використовується для створення веб-сайтів, веб-сервісів та ігор.

Фреймворк Microsoft .Net можна використовувати як для створення програм на основі форм, так і для веб-додатків. Веб-сервіси також можна розробляти за допомогою .Net Framework.

Фреймворк також підтримує різні мови програмування, такі як Visual Basic і C# [2]. Таким чином, розробники можуть вибрати і вибрати мову для розробки необхідної програми.

На рисунку 3 показано, як технології розподілені всередині різних реалізацій .NET.

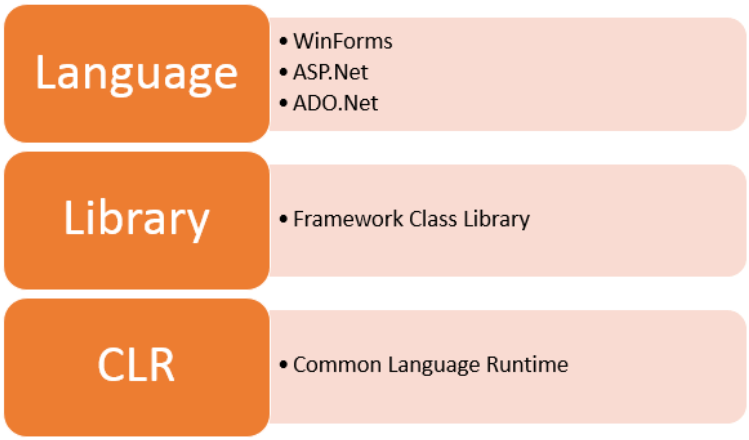

Рисунок 3 – Архітектура системи

2.1.1 .NET Framework і .NET 6

Варто відзначити, що .NET тривалий час розвивався переважно як платформа для Windows під назвою .NET Framework. У 2019 році вийшла остання версія цієї платформи - .NET Framework 4.8. Вона більше не розвивається. [3]

З 2014 року Microsoft став розвивати альтернативну платформу - .NET Core, яка вже призначалася для різних платформ і повинна була увібрати в себе всі можливості застарілого .NET Framework і додати нову функціональність. Потім Microsoft послідовно випустив ряд версій цієї платформи: .NET Core 1, .NET Core 2, .NET Core 3, .NET 5. І поточною версією є платформа .NET 6. Тому слід розрізняти .NET Framework, який призначений переважно для Windows, і кроссплатформенний .NET 6.

Тому для майбутніх проектів слід використовувати актуальні версії платформи .NET

2.2 Платформа ASP.NET core

ASP.NET – це веб-фреймворк з відкритим вихідним кодом для створення веб-програм на платформі .NET (dotNET). Він створений Microsoft, а версія 1.0 була випущена в 2002 році, щоб дозволити розробникам створювати динамічні веб-програми, служби та сайти. Фреймворк створений для роботи зі стандартним протоколом HTTP, який є стандартним протоколом, який використовується в усіх веб-додатках.

ASP.NET є наступником технології ASP (Active Server Pages) і є значним оновленням з точки зору гнучкості та потужності. Це розширення платформи .NET з додатковими інструментами та бібліотеками спеціально для створення в Інтернеті, включаючи веб-програми та веб-сайти.

Остання версія ASP.NET – це кросплатформна версія під назвою ASP.NET Core, яка була випущена в 2016 році. ASP.NET все ще підтримується

та оновлюється, але Microsoft зосереджується на розробці нової кросплатформної версії.

ASP.NET Core характеризується розширюваністю [4]. Фреймворк побудований із набору щодо незалежних компонентів. І розробник має можливість використати вбудовану реалізацію цих компонентів, або розширити їх за допомогою механізму спадкування, або створити і застосовувати свої компоненти зі своїм функціоналом.

ASP.NET Core включає в себе фреймворк MVC, який об'єднує функціональність MVC, Web API і Web Forms у одній програмній моделі ASP.NET Core MVC з доданням ряду додаткових функцій(рисунок 4).

Також було спрощено управління залежностями і конфігурацію проекту.

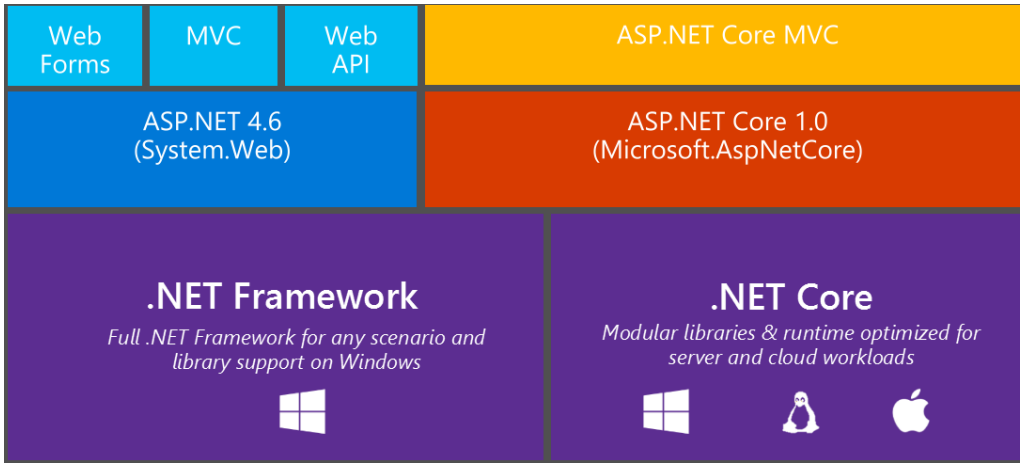

Рисунок 4 – Структура ASP.Net Core

Платформа має такі переваги[1]:

- можливість розробки та запуску в ОС Windows, macOS та Linux;
- відкритий вихідний код та орієнтація на спільноту;
- інтеграція сучасних клієнтських платформ та робочих процесів розробки;

- підтримка розміщення служб віддаленого виклику процедур (RPC) за допомогою gRPC;
- хмарна система конфігурації з урахуванням середовища;
- спрощений високопродуктивний модульний конвеєр HTTP-запитів;
- наступні можливості розміщення:
  - служби IIS;
  - HTTP.sys.
- управління паралельними версіями;
- інструментарій, що спрощує процес сучасної веб-розробки.

2.2.1 ASP.NET MVC

Платформа ASP.NET MVC є фреймворком для створення сайтів і веб-додатків за допомогою реалізації патерну MVC.

Концепція патерну (шаблону) MVC (model - view - controller) передбачає поділ програми на три компоненти:

- Контролер (controller) представляє клас, що забезпечує зв'язок між користувачем та системою, представленням та сховищем даних. Він отримує дані, що вводяться користувачем, і обробляє їх. І в залежності від результатів обробки надсилає користувачеві певний висновок, наприклад, у вигляді представлення.
- Подання (view) – це власне візуальна частина або інтерфейс програми користувача. Як правило, html-сторінка, яку користувач бачить зайшовши на сайт.
- Модель (model) є класом, що описує логіку використовуваних даних. У цій схемі модель є незалежним компонентом – будь-які зміни контролера або представлення не торкаються моделі. Контролер і

представлення є відносно незалежними компонентами, і їх можна змінювати незалежно один від одного.

Загальну схему взаємодії цих компонентів представлено на рисунку 5:

Рисунок 5 – Схема взаємодії компонентів шаблону MVC

У цій схемі модель є незалежним компонентом – будь-які зміни контролера або представлення не торкаються моделі. Контролер і представлення є відносно незалежними компонентами, і їх можна змінювати незалежно один від одного. [5]

Завдяки цьому реалізується концепція розподілу відповідальності, у зв'язку з чим легше побудувати роботу над окремими компонентами. Крім того, внаслідок цього додаток має кращу тестованість. І якщо нам, припустимо, важлива візуальна частина чи фронтенд, ми можемо тестувати представлення незалежно від контролера. Або ми можемо зосередитися на бекенді та тестувати контролер.

Конкретні реалізації та визначення даного патерну можуть відрізнятися, але через свою гнучкість і простоту він став дуже популярним останнім часом, особливо у сфері веб-розробки.

Свою реалізацію патерну представляє платформа ASP.NET MVC. 2013 ознаменувався виходом нової версії ASP.NET MVC - MVC 5, а також релізом Visual Studio 2013, яка надає інструментарій для роботи з MVC5.

2.3 Tag-helper (помічники тегів)

Помічники тегів дозволяють коду на стороні сервера брати участь у створенні та відтворенні елементів HTML. Наприклад, вбудований ImageTagHelper може додати номер версії до імені зображення [6]. Щоразу, коли зображення змінюється, сервер створює нову унікальну версію для зображення, тому клієнти гарантовано отримають поточне зображення (замість застарілого кешованого зображення). Існує багато вбудованих помічників тегів для звичайних завдань, таких як створення форм, посилань, завантаження ресурсів тощо, і ще більше доступно в загальнодоступних сховищах GitHub і як пакети NuGet. Помічники тегів створені на C#, і вони націлені на елементи HTML на основі імені елемента, імені атрибута або батьківського тегу. Наприклад, вбудований LabelTagHelper може орієнтуватися на елемент HTML <label>, коли застосовуються атрибути LabelTagHelper. Якщо ви знайомі з допоміжниками HTML, допомічники тегів зменшують явні переходи між HTML і C# у представленнях Razor. У багатьох випадках помічники HTML надають альтернативний підхід до конкретного допоміжного засобу тегів, але важливо визнати, що допоміжні теги не замінюють допоміжників HTML і для кожного допомічника HTML немає допоміжного засобу. Помічники тегів у порівнянні з помічниками HTML пояснюють відмінності більш детально.

Помічники тегів не підтримуються компонентами Razor.

Спосіб зробити роботу програміста більш продуктивною і здатним створювати більш надійний, надійний і підтримуваний код, використовуючи інформацію, доступну лише на сервері.

Наприклад, раніше проблема щодо оновлення зображень полягала в тому, щоб змінювати назву зображення, коли ви змінюєте зображення. Зображення слід кешувати з міркувань продуктивності, і якщо ви не зміните назву зображення, ви ризикуєте отримати застарілу копію. Історично, після редагування зображення, ім’я потрібно було змінити, а кожне посилання на зображення у веб-програмі потрібно було оновити. Це не тільки дуже трудомістко, але й схильне до помилок (ви можете пропустити посилання, випадково ввести неправильний рядок тощо). Вбудований ImageTagHelper може зробити це за вас автоматично. ImageTagHelper може додати номер версії до імені зображення, тому щоразу, коли зображення змінюється, сервер автоматично створює нову унікальну версію для зображення. Клієнти гарантовано отримають поточне зображення. Ця надійність і економія робочої сили надаються практично безкоштовно за допомогою ImageTagHelper.

Більшість вбудованих помічників тегів націлені на стандартні елементи HTML і надають атрибути на стороні сервера для цього елемента. Наприклад, елемент <input>, який використовується в багатьох представленнях у папці Views/Account, містить атрибут asp-for. Цей атрибут витягує ім’я вказаної властивості моделі у відтворений HTML.

2.4 Bootstrap

Bootstrap – це бібліотека HTML, CSS та JS, яка зосереджена на спрощенні розробки інформаційних веб-сторінок (на відміну від веб-програм). Основна мета його додавання до веб-проекту – застосувати вибір кольору, розміру, шрифту та макета Bootstrap до цього проекту. Таким чином, основним фактором є те, чи знайдуть відповідальні розробники ці варіанти до смаку. Після додавання до проекту Bootstrap надає базові визначення стилів для всіх елементів HTML. Результатом є уніфікований вигляд тексту, таблиць і елементів форм у всіх веб-браузерах. Крім того, розробники можуть

скористатися перевагами класів CSS, визначених у Bootstrap, щоб додатково налаштувати зовнішній вигляд свого вмісту. Наприклад, у Bootstrap передбачені таблиці світлих і темних кольорів, заголовки сторінок, більш помітні лапки та текст із виділенням.

Bootstrap також постачається з кількома компонентами JavaScript у вигляді плагінів jQuery. Вони забезпечують додаткові елементи інтерфейсу користувача, такі як діалогові вікна, підказки та каруселі. Кожен компонент Bootstrap складається з HTML-структури, декларацій CSS і в деяких випадках супровідного коду JavaScript. Вони також розширюють функціональні можливості деяких існуючих елементів інтерфейсу, включаючи, наприклад, функцію автозаповнення для полів введення.

2.4.1 Bootstrap в ASP.NET MVC

Bootstrap в ASP.NET являється css-фреймворком для створення адаптивних веб-додатків. Нині дедалі зростає кількість користувачів мобільного інтернету, все більше людей для веб-серфінгу використовують смартфони, планшети. Це призвело до збільшення кількості сайтів з адаптивним дизайном і відповідно до збільшення популярності засобів для розробки подібних сайтів, у тому числі фреймворку Bootstrap.

Проекти ASP.NET MVC 5 вже містять всі необхідні файли Bootstrap (рисунок 6).

Функціональність Bootstrap міститься у файлі стилів bootstrap.css та скрипті bootstrap.js. Підключення ж відповідних бібліотек відбувається через майстер-сторінку.

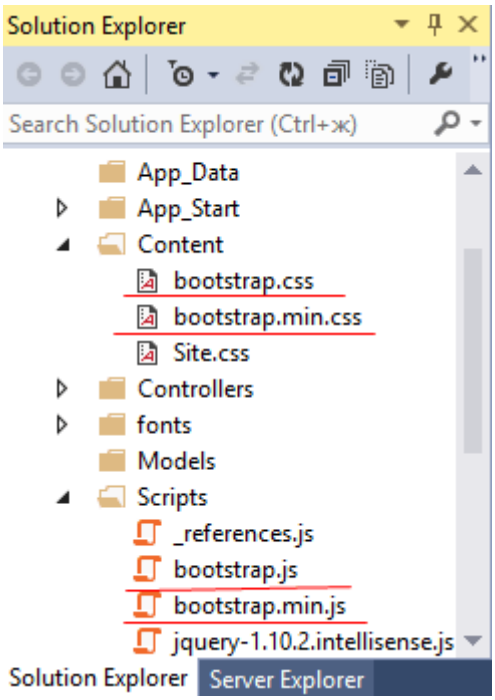

Рисунок 6 – Файли бібліотеки Bootstrap

Також на майстер-сторінці ми можемо побачити використання таких класів CSS, як navbar, navbar-inverse, navbar-fixed-top і т.д. - все це класи bootstrap [7], що дозволяють адаптувати інтерфейс сторінки до різних пристроїв. Зовнішній вигляд сторінки представлений на рисунку 7.

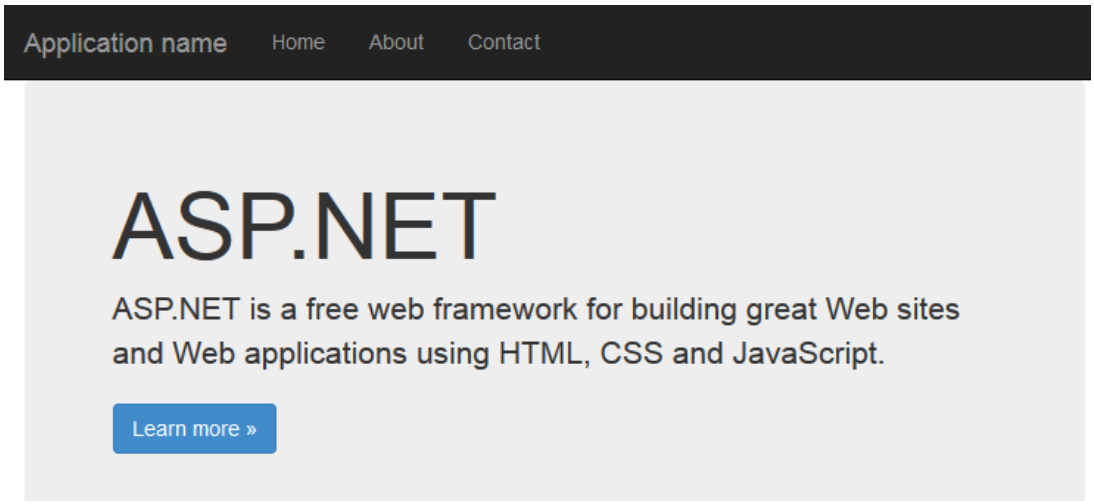

Рисунок 7 – Вигляд сторінки при використанні бібліотек Bootstrap

2.5 База даних SQL

SQL (Structured Query Language) – це мова домену, яка використовується в програмуванні та призначена для керування даними, що зберігаються в системі керування реляційною базою даних (RDBMS), або для обробки потоку в системі керування реляційними потоками даних (RDSMS). Це особливо корисно для роботи зі структурованими даними, тобто даними, що включають відношення між сутностями та змінними. SQL пропонує дві основні переваги перед старими API читання-запису, такими як ISAM або VSAM. По-перше, він представив концепцію доступу до багатьох записів за допомогою однієї команди. По-друге, це усуває необхідність вказувати, як досягти запису, напр. з індексом або без. Єдине рішення для створення власного веб-інтерфейсу і веб-API.

Razor Pages спрощує написання коду для сценаріїв сторінок та підвищує його ефективність.

Blazor дозволяє використовувати у браузері мову C# разом із JavaScript. спільне використання серверної та клієнтської логік додатків, написаних за допомогою .NET.

SQL, що спочатку базувався на реляційній алгебрі та кортежному реляційному обчисленні, складається з багатьох типів операторів, які неофіційно можуть бути класифіковані як підмови, зазвичай: мова запитів даних (DQL), мова визначення даних (DDL), мова керування даними (DCL), і мова маніпулювання даними (DML). Сфера застосування SQL включає запити даних, маніпулювання даними (вставка, оновлення та видалення), визначення даних (створення та зміна схеми) та контроль доступу до даних. Хоча SQL по суті є декларативною мовою (4GL), він також включає процедурні елементи.

SQL був однією з перших комерційних мов, яка використовувала реляційну модель Едгара Ф. Кодда. Модель була описана в його впливовій статті 1970 року «Реляційна модель даних для великих спільних банків даних».

Незважаючи на те, що вона не повністю дотримується реляційної моделі, описаної Коддом, вона стала найпоширенішою мовою баз даних.

SQL став стандартом Американського національного інституту стандартів (ANSI) у 1986 році та Міжнародної організації зі стандартизації (ISO) у 1987 році. Відтоді стандарт було переглянуто, щоб включити більший набір функцій. Незважаючи на існування стандартів, більшість коду SQL вимагає принаймні деяких змін перед перенесенням в різні системи баз даних.

Для забезпечення додатку інформацією та реалізувати можливість подальшого її збереження, для проекту була створена база даних у середовищі SQL Server Express.

2.5.1 SQL Server Express LocalDB

На кінцевому етапі розробки проекту база даних повинна бути переміщена у віртуальний диск в мережі для можливості використання її на інших пристроях, під'єднаних до інтернету. Для цього проводимо від'єднання БД від серверу та змінюємо правила безпеки для отриманого файлу з метою надання можливості редагування іншим користувачам.

Під час створення програмного рішення файл бази даних безпосередньо знаходиться на робочому ПК. Для цього була створена окрема папка WorkersDB, у директорії рішення, для зберігання файлу mdf та ldf для збереження лог-інформації. Оскільки у кінці додаток буде завантажений на хост, який використовує всі файли побудови проекту, тому базу даних потрібно додатково зберігати в різних копіях задля унеможливлення небажаної зміни або видалення.

LocalDB в Microsoft SQL Server Express - це компонент SQL Server Express, орієнтований на розробників [8]. Він доступний у SQL Server Express з додатковими службами.

При установці LocalDB копіюється мінімальний набір файлів, необхідних для запуску компонента Компонент SQL Server Database Engine.

Після встановлення LocalDB можна встановити з'єднання за допомогою спеціального рядка підключення. При з'єднанні необхідна інфраструктура SQL Server створюється та запускається автоматично, що дозволяє застосунку використовувати базу даних без виконання складного налаштування. Кошти розробника дозволяють використовувати ядро СУБД SQL Server для створення та перевірки коду Transact-SQL без зобов'язань з управління повноцінним екземпляром SQL Server.

2.6 Архітектурне рішення REST

2.6.1 REpresentational State Transfer

REST, або REpresentational State Transfer, – це архітектурний стиль для забезпечення стандартів між комп'ютерними системами в Інтернеті, що полегшує взаємодію систем між собою. REST-сумісні системи, які часто називають системами RESTful, характеризуються тим, що вони не мають стану та відокремлюють проблеми клієнта і сервера.

На рисунку показано схему роботи REST системи (рисунок 8).

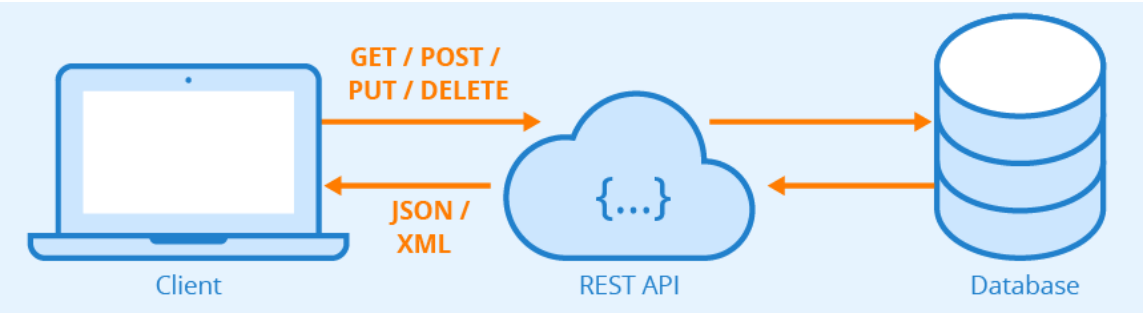

Рисунок 8 – Схема роботи REST системи

2.6.2 Розділення клієнта і сервера

В архітектурному стилі REST реалізацію клієнта і реалізацію сервера можна виконувати незалежно один від одного, не знаючи один про одного. Це означає, що код на стороні клієнта можна змінити в будь-який час, не впливаючи на роботу сервера, а код на стороні сервера можна змінити, не впливаючи на роботу клієнта.

Поки кожна сторона знає, який формат повідомлень надсилати іншій, їх можна зберігати модульними та окремими. Відокремлюючи проблеми інтерфейсу користувача від проблем зберігання даних, ми покращуємо гнучкість інтерфейсу на різних платформах і покращуємо масштабованість, спрощуючи компоненти сервера. Крім того, поділ дозволяє кожному компоненту розвиватися незалежно.

Використовуючи інтерфейс REST, різні клієнти звертаються до тих самих кінцевих точок REST, виконують однакові дії та отримують однакові відповіді.

Як видно, архітектура REST дуже проста в плані використання. По виду запиту відразу можна визначити, що він робить, не розбираючись в форматах (на відміну від SOAP, XML-RPC). Дані передаються без застосування додаткових шарів, тому REST вважається менш ресурсоємним, оскільки не треба питати запит щоб зрозуміти що він повинен зробити і не треба переводити дані з одного формату в інший.

2.6.3 Практичне застосування

Найголовніше перевага сервісів у тому, що з ними працювати може будь-яка система, будь то сайт, flash, програма та ін. Оскільки методи парсингу XML і виконання запитів HTTP присутні майже скрізь.

Архітектура REST дозволяє серйозно спростити це завдання. Звичайно в дійсності, того що описано мало, адже не можна будь-кому давати можливість змінювати інформацію, тобто необхідна ще авторизація та

автентифікація. Але це досить просто дозволяється за допомогою різного типу сесій або HTTP Authentication.

2.7 Microsoft Azure

Microsoft Azure, раніше відома як Windows Azure, є загальнодоступною платформою хмарних обчислень Microsoft. Він надає цілий ряд хмарних послуг, включаючи обчислення, аналітику, зберігання та мережу [9]. Користувачі можуть вибирати з цих служб для розробки та масштабування нових програм або запуску існуючих програм у загальнодоступній хмарі.

Платформа Azure має на меті допомогти підприємствам справлятися з проблемами та досягати своїх організаційних цілей.

Azure пропонує 4 різні форми хмарних обчислень:

- Інфраструктура як послуга (IaaS);
- Платформа як послуга (PaaS);
- Програмне забезпечення як послуга (SaaS);
- Безсерверна реалізація.

2.7.1 Схема користування Microsoft Azure

Корпорація Майкрософт стягує плату за Azure на основі оплати по мірі використання, тобто передплатники щомісяця отримують рахунок, який стягує з них лише ті ресурси, які вони використали [10].

Після підписки на Azure клієнти отримують доступ до всіх служб, включених на портал Azure. Передплатники можуть використовувати ці служби для створення хмарних ресурсів, таких як віртуальні машини (VM) і бази даних.

2.8 Висновки до розділу

У даному підрозділі проаналізовано теоретичні відомості з питань, які розглядаються з метою виконання проектної частини дипломної роботи.

Описано проектування багатошарової архітектури системи та подано обґрунтування даного вибору.

Описано засоби мови програмування, вивчення яких передбачається навчальною програмою дисципліни «Програмування», системи управління базами даних, вивчення яких передбачається навчальною програмою дисципліни «Організація баз даних», та архітектурні рішення, вивчення яких передбачається навчальною програмою дисципліни «Розробка та тестування програмного забезпечення» розглянуто технології для співставлення об’єктів бази даних та об’єктів мови програмування.

3 Проектна частина

Як вже зазначалось вище, завданням даної роботи є реалізація програми обліку працівників. Система складається з незалежних один від одного модулів та розміщена на локальному диску.

3.1 Аналіз предметної області

На основі аналізу вимог до системи було сформовано перелік сутностей для реалізації системи (таблиця 3.1).

Таблиця 3.1 – Перелік сутностей та їх атрибутів

Сутність	Назва класу	Атрибути
Посада	Seat	Назва, Опис.
Працівник	Worker	Ім’я, Посада, Ціна, Дата народження.
Оператор	Operator	Ім’я, Електронна пошта, Пароль, Номер телефону.
Рахунок	Bill	Дата, Оператор, Сума.

3.2 Реалізація бази даних для сутностей

Для реалізації додатку було створено рішення в середовищі Visual Studio з назвою OnlineWorkersManager та база даних з назвою WorkersAspDb, що містить таблицю для сутностей(рисунок 9).

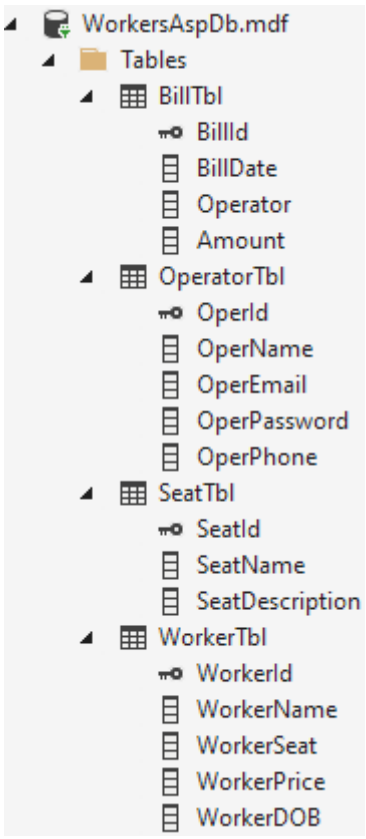

Рисунок 9 – Структура бази даних WorkersAspDb

BillTbl являється таблицею для рахунків (лістинг 3.1). Дана таблиця має зв'язок з таблицею Operator а точніше з рядком OperId. Поле Amount є розрахунковим, і вноситься програмним кодом, а не користувачем.

Лістинг 3.1 –Таблиця BillTbl

```
CREATE TABLE [dbo].[BillTbl] (
[BillId]   INT  IDENTITY (1, 1) NOT NULL,
[BillDate] DATE NOT NULL,
[Operator] INT  NOT NULL,
[Amount]   INT  NOT NULL,
PRIMARY KEY CLUSTERED ([BillId] ASC),
CONSTRAINT [FK2] FOREIGN KEY ([Operator]) REFERENCES
[dbo].[OperatorTbl] ([OperId])
);
```

OperatorTbl є таблицею для збереження даних для входу користувачів, які не мають адміністраторських переваг (лістинг 3.2).

Лістинг 3.2 –Таблиця OperatorsTbl

```
CREATE TABLE [dbo].[OperatorTbl] (
[OperId]       INT          IDENTITY (1000, 1) NOT NULL,
[OperName]     VARCHAR (50) NOT NULL,
[OperEmail]    VARCHAR (50) NOT NULL,
[OperPassword] VARCHAR (50) NOT NULL,
[OperPhone]    VARCHAR (50) NOT NULL,
PRIMARY KEY CLUSTERED ([OperId] ASC)
);
```

SeatTbl дана таблиця призначена для збереження даних про посади (лістинг 3.3). Поле SeatDescription є необов’язковим для заповненням.

Лістинг 3.3 –Таблиця SeatTbl

```
CREATE TABLE [dbo].[SeatTbl] (
[SeatId] INT                    IDENTITY (100, 1) NOT NULL,
[SeatName] VARCHAR (50)         NOT NULL,
[SeatDescription] VARCHAR (150) NULL,
PRIMARY KEY CLUSTERED ([SeatId] ASC)
```

WorkerTbl є елементом бази даних, що має призначення в збереженні даних про працівників (лістинг 3.4). Поле WorkerSeat є пов’язане з таблицею SeatTbl та зобов’язує користувача при додаванні працівника обирати тільки ті посади, які були додані в таблицю SeatTbl.

Лістинг 3.4 – Таблиця WorkerTbl

```
CREATE TABLE [dbo].[WorkerTbl] (
[WorkerId]    INT          IDENTITY (100, 1) NOT NULL,
[WorkerName]  VARCHAR (50) NOT NULL,
[WorkerSeat]  INT          NOT NULL,
[WorkerPrice] INT          NOT NULL,
```

Продовження лістингу 3.4

```
[WorkerDOB] DATE NOT NULL,
PRIMARY KEY CLUSTERED ([WorkerId] ASC),
CONSTRAINT [FK5] FOREIGN KEY ([WorkerSeat]) REFERENCES
[dbo].[SeatTbl] ([SeatId])
);
```

3.3 Реалізація рівня доступу до даних

На даному рівні було реалізовано спосіб зберігання та доступу до даних, шляхом реалізації двох інтерфейсів та навігаційних стрічок.

3.3.1 Реалізація підключення до бази даних

Підключення до бази даних здійснюється через клас Functions.cs (лістинг 3.5) та містить в собі методи getData, що відповідає за отримання даних від бази даних, і setData, що має на свої меті змінити дані, які вказав користувач. Під час розробки проекту, з метою зменшення витрат та збільшення ефективності роботи, використовулась локальна база даних.

Кожен постачальник даних платформи .NET Framework має об'єкт Connection, що успадковує з DbConnection, а також із властивості ConnectionString, що залежить від постачальника. Конкретний синтаксис рядка підключення для кожного постачальника міститься в його властивості ConnectionString. В даному випадку рядок підключення знаходиться в змінні ConnString.

Лістинг 3.5 – Клас Functions.cs

```
public class Functions{
     private SqlConnection Con;
     private SqlCommand cmd;
     private DataTable dt;
     private SqlDataAdapter sda;
     private string ConnString;
```

Продовження лістингу 3.5

```
            public Functions(){
                ConnString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Roman
Kostiuk\Documents\WorkersAspDb.mdf;Integrated
Security=True;Connect
Timeout=15;Encrypt=False;TrustServerCertificate=False;";
                Con = new SqlConnection(ConnString);
                cmd = new SqlCommand();
                cmd.Connection = Con;
            }
            public DataTable getData(string Query)
            {
                dt = new DataTable();
                sda = new SqlDataAdapter(Query, ConnString);
                sda.Fill(dt);
                return dt;
            }
            public int SetData(string Query)
            {
                int Cnt = 0;
                if(Con.State == ConnectionState.Closed)
                {
                    Con.Open();
                }
                cmd.CommandText = Query;
                Cnt = cmd.ExecuteNonQuery();
                Con.Close();
                return Cnt;
            }
        }
```

3.3.2 Реалізація інтерфейсу Admin та Operator

Даний інтерфейс дає доступ до сторінки Dashboard.aspx, Operators.aspx, Seats.aspx, Workers.aspx. Також забезпечує наявність навігаційної стрічки адміністратора (рисунок 10), що має посилання до всіх наявних сторінок, та кнопку виходу, що переносить на сторінку входу (додаток А).

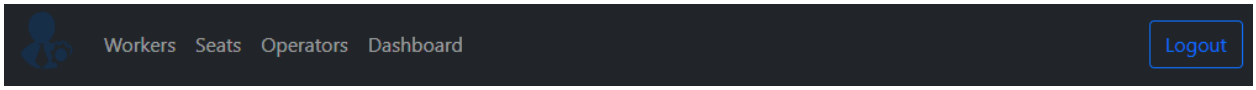

Рисунок 10 – Вигляд навігаційної стрічки адміністратора

Дана навігаційна стрічка знаходиться на всіх сторінках до яких має доступ адміністратор. Це зумовлено використанням додаткових функцій тегів (лістинг 3.6).

Лістинг 3.6 – Використання тегу asp

```
	<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
	</asp:Content>
	<asp:Content ID="Content2" ContentPlaceHolderID="Mybody" runat="server">
```

Цей тег дозволяє використовувати різні елементи, класи а також цілі конструкції як частину користувацького інтерфейсу для інших сторінок. При цьому не потрібно кожен раз переписувати програмний код та виконувати налаштування відступів. Достатньо просто присвоїти йому значення властивості ID та ContentPlaceHolderID та використати в бажаній конструкції.

Навігаційна стрічка для користувача типу Operator не має можливостей переходу між сторінками (рисунок 11).

Рисунок 11 – Вигляд навігаційної стрічки оператора

3.4 Реалізація авторизації користувачів

Для реалізації авторизації користувачів було створено власну систему авторизації на основі однофакторної системи (додаток Б).

Ключовими полями є електронна пошта та пароль. Вся функціональність зі зміною записів виконується за допомогою SQL запитів та представлення на прикладі функції додавання запису (лістинг 3.7).

Лістинг 3.7 – Функція додавання запису оператору

```
      string SName = SNameTb.Value;
          string SEmail = SEmailTb.Value;
          string Password = OperPassTb.Value;
          string Phone = PhoneTb.Value;
          string    Query    =    "insert    into    OperatorTbl
values('{0}','{1}','{2}','{3}')";
          Query = string.Format(Query, SName, SEmail, Password,
Phone);
          Con.SetData(Query);
          ShowOperators();
          ErrMsg.InnerText = "Operator Added!";
```

Також було реалізовані функції видалення (лістинг 3.8) та редагування записів операторів (лістинг 3.9). Дані функції використовують SQL-запити для зміни записів таблиць бази даних. Щоб змінити записи інших таблиць потрібно просто змінити назву потрібної таблиці та імена рядків.

Лістинг 3.8 – Функція видалення запису оператора

```
      string SName = SNameTb.Value;
      string Query = "delete from  OperatorTbl where OperId =
{0}";
      Query =
string.Format(Query,OperGV.SelectedRow.Cells[1].Text);
      Con.GetData(Query);
      Con.SetData(Query);
      ShowOperators();
      ShowOperator();
      ErrMsg.InnerText = "Operator Deleted!";
```

Лістинг 3.9 – Функція видалення запису оператора

```
        string SName = SNameTb.Value;
        string SEmail = SEmailTb.Value;
        string Password = OperPassTb.Value;
        string Phone = PhoneTb.Value;
        string Query = "update OperatorTbl set OperName =
'{0}',OperEmail='{1}',OperPassword='{2}',OperPhone = '{3}' where
OperId = {4}";
        Query = string.Format(Query, SName, SEmail, Password,
Phone, OperGV.SelectedRow.Cells[1].Text);
        Con.SetData(Query);
        ShowOperators();
        ErrMsg.InnerText = "Operator Updated!"
```

В цих функціях використовується метод ShowOperators(), задача якого показувати список операторів з актуальними даними запису, неоновлюючи сторінку. Сам вигляд сторінки Operators.aspx представлений на рисунку 12.

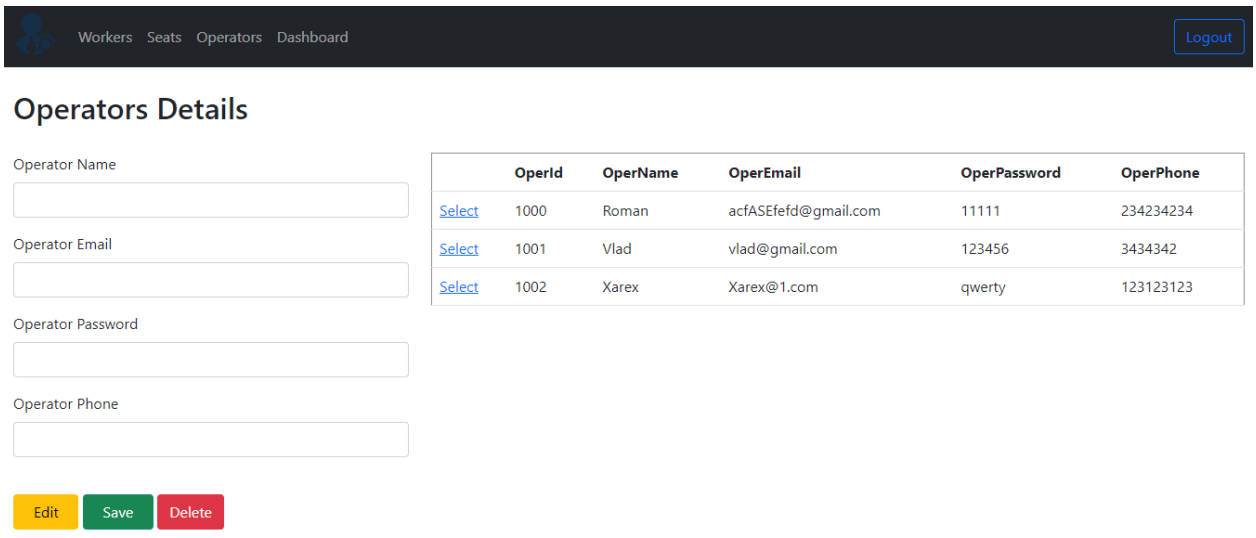

Рисунок 12 – Вигляд сторінки Operators.aspx

Також було налаштовано поле вводу Operator Email. Щоб уникнути введення користувачами інкоректних даних, які не є електроною адресою, було присвоєно типу поля властивість “email” (лістинг 3.10). У випадку

неправильного введення даних з'являється повідомлення-підсказка про необхідність правильного формату ведення (рисунок 13).

Лістинг 3.10 – HTML код поля Operator Email

```
      <div class="mb-3">
      <label for="SEmailTb" class="form-label">Operator
Email</label>
      <input type="email" class="form-control" id="SEmailTb"
runat="server">
```

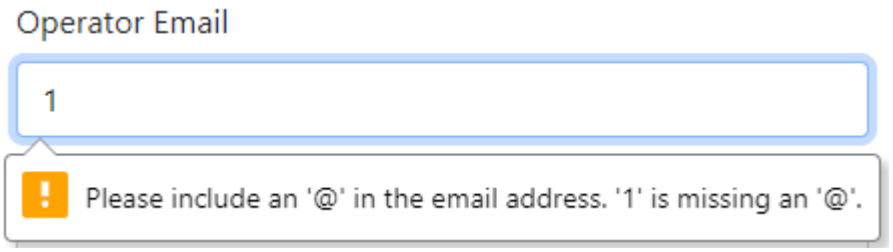

Рисунок 13 – Повідомлення підсказка поля Operator Email

3.5 Створення функцій додавання посад та працівників

Сторінки «Seats.aspx» та «Workers.aspx» дають можливість додавати, видаляти та редагувати записи для адміністраторів. Для операторів дані сторінки недоступні. Основним елементом цих двох сторінок є GridView, який виводить саму інформацію о записах, та дає змогу їх вибирати (лістинг 3.11).

Лістинг 3.11 – Елемент GridView в сторінці Seat.aspx

```
<asp:GridView runat="server"
class="table table-hover" ID="OperGV"
AutoGenerateSelectButton="True"
OnSelectedIndexChanged="OperGV_SelectedIndexChanged">
</asp:GridView>
```

Заради збільшення ефективності роботи користувача та зменшення часу заповнення записів працівників, вибір посади для працівників здійснюється через метод GetSeats() (лістинг 3.12).

Лістинг 3.12 – Метод GetSeats()

```
      private void GetSeats(){
      string Query = "Select * from SeatTbl";
      SeatCb.DataTextField =
Con.getData(Query).Columns["SeatName"].ToString();
      SeatCb.DataValueField =
Con.getData(Query).Columns["SeatId"].ToString();
      SeatCb.DataSource = Con.getData(Query);
      SeatCb.DataBind();}
```

Даний метод бере назви про існуючі посади з таблиці SeatTbl та виводить їх в випадаючий список (рисунок 14), тим самим розробнику не потрібно робити обов'язкову перевірку даних поля на правильність.

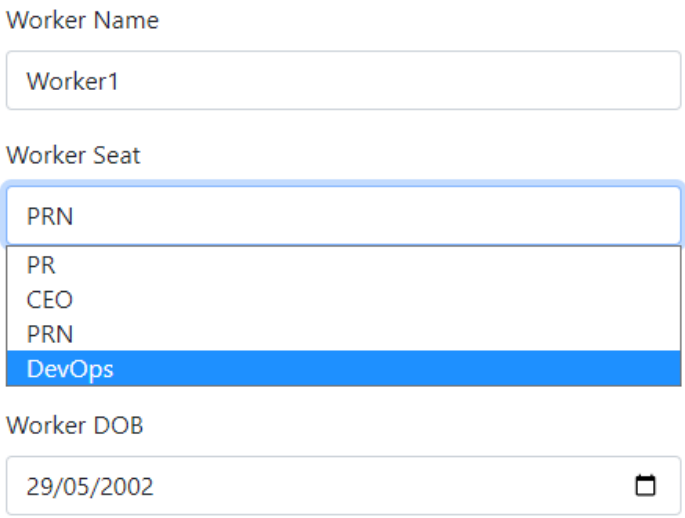

Рисунок 14 – Випадаючий список з назвами посад

3.6 Реалізація сторінки виплат

Сторінка Billing є єдиною сторінкою яка є доступною для оператора (додаток В). Нею мають право користуватись тільки користувачі, що роблять виплати по заробітнім платам працівникам вигляд якої проілюстрований на рисунку 15.

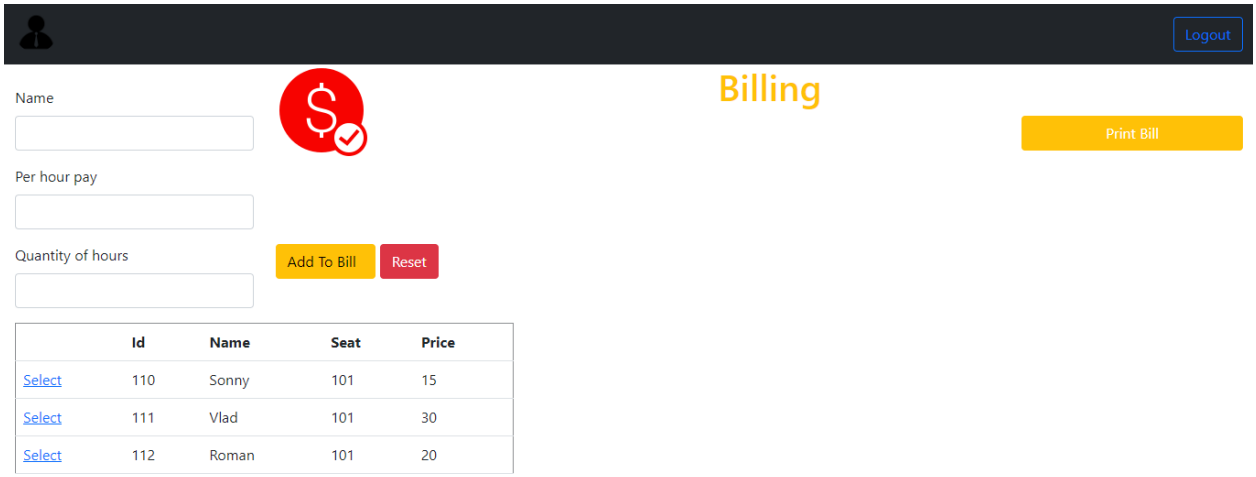

Рисунок 15 – Вигляд сторінки Billing.aspx

Щоб здійснити виплату необхідно спочатку вибрати потрібного працівника. Для полегшення навігації працівників оператором, було додано таблицю WorkerGv (лістинг 3.13) для виведення записів з елементу бази даних WorkerTbl.

Лістинг 3.13 – Таблиця WorkerGv

```
	protected void WorkerGV_SelectedIndexChanged(object sender,
EventArgs e){
	WorkerNameTb.Value = WorkerGV.SelectedRow.Cells[2].Text;
	WorkerPriceTb.Value = WorkerGV.SelectedRow.Cells[4].Text;
	}
```

Далі після вибору працівника потрібно ввести число пропрацьованих ним годин. Після натискання на кнопку Add To Bill (лістинг 3.14), прорахує його зарплатню, яка обчислюється за формулою (3.1):

$$З = Т_г \cdot t \qquad (3.1)$$

де Тг – число робочих годин;

t – число годин роботи оператора.

Лістинг 3.14 – Функція кнопки Add To Bill

```
      int total = Convert.ToInt32(WorkerQtyTb.Value) *
Convert.ToInt32(WorkerPriceTb.Value);
      DataTable dt = (DataTable)ViewState["Bill"];
      dt.Rows.Add(BillGV.Rows.Count + 1,
      WorkerNameTb.Value.Trim(),
      WorkerPriceTb.Value.Trim(),
      WorkerQtyTb.Value.Trim(),
      total
      );
      ViewState["Bill"] = dt;
      this.BindGrid();
      GrdTotal = 0;
      for (int i = 0; i <= BillGV.Rows.Count - 1; i++)
      {
      GrdTotal = GrdTotal +
Int32.Parse(BillGV.Rows[i].Cells[4].Text.ToString());
      }
      Amount = GrdTotal;
      GrdTotTb.InnerHtml = "$" + GrdTotal;
      WorkerNameTb.Value = string.Empty;
      WorkerPriceTb.Value = string.Empty;
```

Після обрахування, програма внесе працівника на виплату в додаткову таблицю BillGrid (лістинг 3.15). Потрібно розуміти, що для здійснення виплат можна вибирати та обраховувати відразу декількох працівників з функцією сумуванням сум виплат, як це показано на рисунку 16.

Лістинг 3.15 – Додаткова таблиця BillGrid

```
protected void BindGrid()
{
BillGV.DataSource = (DataTable)ViewState["Bill"];
BillGV.DataBind();
}
```

Рисунок 16 – Вигляд додаткової таблиці BillGrid

Також оператору надана можливість роздрукувати звіт за допомогою принтера, або зберегти записи з додаткової таблиці BillGrid (рисунок 17).

Сам формат друку в обох варіантах незмінний, шрифт розмір та координати тексту залежать безпосередньо від формату таблиці BillGrid. Цю функцію реалізовано, щоб дати можливим оператору швидко дістати звіт по зарплатні та роздрукувати її.

Після цього в таблицю BillTbl вносяться дані про дату виплати, оператора, здійснив операцію та загальну суму.

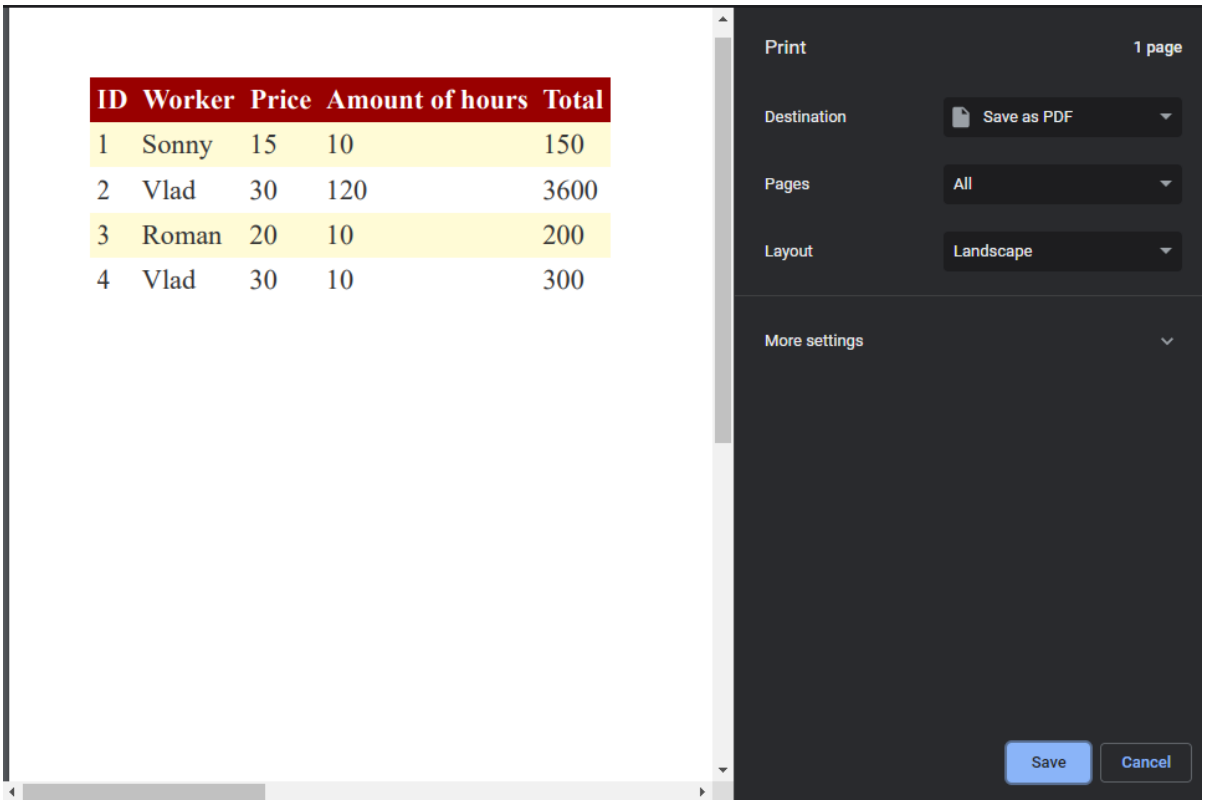

Рисунок 17 – Вигляд діалогового вікна PrintPanel()

3.7 Створення сторінки для відображення статистики

Сторінка Dashboard, є єдиною сторінкою, в якій не потрібно вводити дані (додаток Г). Вона представляє інформацію для адміністраторів про кількість записів в таблицях (рисунок 18) та обрахунку суми здійснених виплат (лістинг 3.16).

Рисунок 18 – Статистика про кількість записів в сутностях

Лістинг 3.16 – Розрахунок суми витрат на виплати

```
      private void SumSell()
      {
      string Query = "Select Sum(Amount) from BillTbl";
      FinanceTb.InnerText ="$  " +
Con.getData(Query).Rows[0][0].ToString();
      FinanceTb.DataBind();
      }
```

3.8 Висновки до розділу

У цьому розділі описано практичну частини дипломної роботи: розмежування компонентів системи на різні рівні, проектування кожного рівня, розробку серверної частини, розробку клієнтської частини. Описані усі використані технології та приведено приклад їх використання в ході розробки. Показано графічні приклади інтерфейсу клієнтської частини.

4 Економічна частина

4.1 Розрахунок витрат на розробку інтернет-застосунка менеджера працівників

Інтернет-застосунок або вебзастосунок – це прикладне програмне забезпечення, яке працює у веб-браузері, на відміну від програм, які працюють локально й безпосередньо в операційній системі (ОС) пристрою. Веб-програми постачаються у всесвітній мережі користувачам із активним мережевим підключенням.

Вебзастосунки створюються для таких ж цілей що і звичайні застосунки, наприклад: редагування тексту, створення мелодії, розваги, та інші. Але через те що самий код знаходиться в мережі, кожен користувач з інтернет-з’єднанням може користуватися даним застосунком з будь якого пристрою що може під’єднатися до сторінки програми. Це є головною перевагою даного типу застосунків.

Деякі компанії створюють веб-аналоги звичайних досить популярних застосунків, які мають менший функціонал. Це дозволяє задіяти менше ресурсів ЕОМ при використанні веб програм.

При розробці нового програмного забезпечення потрібно зробити розрахунок економічного ефекту від реалізації програмного продукту. Розробка нового продукту повинна бути вигідною як і для розробника, так і споживача. При купівлі проекту, клієнт сподівається зменшити витрати часу, ресурсів і т.д. Отже, необхідно оцінити економічний ефект для всіх сторін. Для цього в економічній частині потрібно розрахувати:

- скласти кошторис витрат на розробку програмного продукту;
- розрахувати експлуатаційні витрати, пов’язані з використанням нового програмного продукту;
- розрахувати обсяг роботи, який може бути виконаний з застосуванням нового програмного продукту.

Кошторис витрат на проведення розробки програмного продукту включає розрахунок таких основних статей витрат:

1. Основна заробітна плата розробників, яка розраховується за формулою (4.1):

$$З_0 = \frac{М}{Т_р} t \text{ [грн.]}, \quad (4.1)$$

де М – місячний посадовий оклад конкретного розробника, грн.,

Тр – число робочих днів в місяці (22),

t – число днів роботи розробника.

Розрахунки зведено до таблиці 4.1.

Таблиця 4.1 – Зведений розрахунок заробітної плати

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	35000	1250	22	35000
Інженер-програміст	23000	821,4	22	23000
Всього				58000

2. Додаткова заробітна плата розробників.

Розраховується як 10%-15% від основної заробітної плати розробників.

$$ЗП_{додаткова} = (35000 + 23000) * 0,1 = 5800 \text{ грн.}$$

3. Нарахування на соціальні потреби.

Нарахування на соціальні потреби, які складають 22 % беруться від суми основної та додаткової заробітної плати:

$$ЄСВ = 0{,}22 * (58000 + 5800) = 14036 \text{ грн.}$$

Тобто, нарахування на соціальні потреби складають 14036 грн.

4. Амортизація персонального комп'ютера.

Розмір амортизаційних відрахувань при умові рівномірного перенесення зносу основних фондів на вартість виконаних робіт розраховується за формулою (4.2):

$$A = \frac{В_{поч} - В_{лікв}}{Т_{к.в} * 12} * t \text{ [грн]} \qquad (4.2)$$

де $В_{поч}$ – балансова вартість обладнання, грн.,

$В_{лікв}$ – ліквідаційна вартість обладнання грн.,

$Т_{кв}$– термін корисного використання обладнання, цілі місяці.

t – кількість місяців експлуатації обладнання.

Результати розрахунків зведено до таблиці 4.2.

Таблиця 4.2 – Розрахунок амортизаційних відрахувань

Найменування обладнання	Балансова вартість, грн.	Ліквідаційна вартість, грн	Термін використання обладнання, міс	Величина амортизаційних відрахувань, грн.
Комп'ютер	30500	18000	24	520,8
Периферійне обладнання	24000	15000	24	375
Всього				895,8

5. Витрати на матеріали, що були використані на розробку програмного продукту.

Витрати на матеріали, що були використані на розробку програмного продукту – канцелярське приладдя.

Проведені розрахунки зведено до таблиці 4.3.

Таблиця 4.3 – Розрахунок вартості матеріалів

Найменування матеріалу, марка, тип, сорт	Ціна, грн	Витрачено	Вартість витрачених Матеріалів, грн
Папір А4	225	1	225
Тонер чорний	106	1	106
Ручка кулькова чорна	10	3	30
Маркер текстовий	23	1	23
Всього			384

6. Витрати на силову електроенергію, розраховуються за формулою (4.3):

$$В_с = В * П * Ф \text{ [грн.]} \quad (4.3)$$

де, В – вартість 1 кВт-години електроенергії, В = 5,12 грн./кВт,

П – установлена потужність комп'ютера та інших пристроїв, П=0,4кВт,

Ф – фактична кількість годин роботи комп'ютера, Ф=496 год.

$$В_с = 5,12 \cdot 0,4 \cdot 496 = 1015,8 \text{ [грн.]}$$

7. Інші витрати – приймаємо як 80-150% від основної заробітної плати розробників.

$$В_{інші} = 58000 \cdot 1{,}5 = 87000 \text{ [грн.]}$$

8. Сума всіх попередніх витрат дає загальні витрати на розробку програмного продукту.

$В_{сум}$ = 58000 + 14036 + 895,8 + 384 + 1015,8 + 87000 = 161331,6 [грн.]

4.2 Розрахунок експлуатаційних витрат у споживача, пов'язані з використанням нового програмного продукту

1. Розрахунки заробітної плати. Заробітна плата обслуговуючого персоналу $З_{обс}$, розрахуємо за формулою (4.4)

$$З_{обс} = 12 * М * \beta \text{ [грн.]} \qquad (4.4)$$

де 12 – число місяців;

М – місячний посадовий оклад конкретного інженерно-технічного працівника – 23000 грн.

β – доля часу, який витрачає працівник на виконання робіт з застосуванням програми – 0,5 (50%).

$$З_{обс.} = 12 \cdot 23000 \cdot 0{,}5 = 138000 \text{ [грн/рік]}$$

2. Додаткова заробітна плата обслуговуючого персоналу. Розраховується як 10-12% від основної заробітної плати обслуговуючого персоналу.

$$З_д = 138000 \cdot 0,12 = 16560 \text{ [грн./рік]}$$

3. Нарахування на заробітну плату. Нарахування на заробітну плату $Н_{зп}$ в 2022 році складають 22% від суми основної та додаткової заробітної плати розробника, тобто від $З_о+З_д$.

$$Н_{зп} = (138000 + 16560) \cdot 0,22 = 34003,2 \text{ [грн.]}$$

4. Витрати на електроенергію. При живленні із електромережі витрати на електроенергію розраховуються за формулою (4.5):

$$В_в = В * П * Ф * К_n * \beta \quad (4.5)$$

де В – вартість 1 кВт-години електроенергії– 5,12 грн./кВт.;

П – потужність комп'ютера разом з іншими приладами – 0,5 кВт;

Ф – фактична кількість годин роботи комп'ютера разом з принтером та іншими приладами за рік – 3168 годин;

$К_п$ – коефіцієнт використання потужності. $К_п$ = 0,9;

β – доля часу, який витрачає працівник на виконання конкретних робіт з застосуванням даної програми в загальному часі своєї роботи – 0,5.

$$В_с = 5,12 \cdot 0,5 \cdot 3168 \cdot 0,9 \cdot 0,5 = 3649,536 \text{ [грн./рік]}$$

5. Амортизаційні відрахування. Амортизаційні відрахування розраховуємо за формулою (4.6):

$$В_в = \frac{В_{поч} - В_{лікв}}{Т_{к.в}} [грн.] \quad (4.6)$$

де $В_{поч}$– балансова вартість персонального комп'ютера, грн

$В_{лікв}$– ліквідаційна вартість,грн

$Т_{кв}$ – термін корисного використання, цілі місяці

$$В_в = \frac{30500 - 10700}{24} = 825 [грн.]$$

6. Витрати на ремонт комп'ютерної техніки.

Витрати на ремонт комп'ютерної техніки можна розрахувати за формулою (4.7):

$$Р = [(0,04 / 0,1) * Ц + Зр] * β \ , \quad (4.7)$$

де, Ц – балансова вартість апаратури, грн.;

$З_р$ – заробітна плата окремо найнятих робітників, що зайняті проведенням ремонтних робіт, грн.;

β – доля часу, яку витрачає працівник на виконання конкретних робіт з застосуванням даного програмного продукту в загальному часі своєї роботи.

4-10% - від вартості обладнання складають матеріальні витрати на ремонт комп`ютерної техніки.

$$Р = [(0,04 \div 0,1) \cdot 30500 + 15000] \cdot 0,5 = 13600 [грн./рік]$$

7. Інші витрати. Інші витрати приймаємо, як 6-10% від загальної суми попередніх витрат.

$В_{інші}$ = (161331,6 + 16560 + 34003,2 + 3649,536 + 825 + 13600) · 0,6 = 137981,6 [грн.]

8. Експлуатаційні витрати при використанні програмного продукту. Сума витрат за всіма попередніми статтями і дає величину експлуатаційних витрат при використанні програмного продукту.

$E_в$ = 161331,6 + 16560 + 34003,2 + 3649,536 + 825 + 13600 + 137981,6 = 367950,936 [грн.]

4.3 Розрахунок обсягу робіт при використанні програмного продукту

$$Q_1 = \left(\frac{F * 60 * \beta}{t_1}\right) (\text{умов. од.}) \quad (4.8)$$

$$Q_2 = \left(\frac{F * 60 * \beta}{t_2}\right) (\text{умов. од.}) \quad (4.9)$$

де Q_1 – обсяг робіт при застосуванні старого програмного продукту, умовних одиниць, штук, тощо;

Q_2 – обсяг робіт при застосуванні нового програмного продукту, умовних одиниць, штук, тощо;

F – ефективний фонд часу роботи за рік;

β – доля часу, яку витрачає працівник на виконання конкретних робіт з застосуванням даного програмного продукту, в загальному часі своєї роботи;

t_i – час виконання конкретної функції, хвилин.

Початкові дані:

- ефективний фонд часу роботи за рік F = 3168 год.;
- доля часу, яку витрачає працівник на виконання конкретних робіт з застосуванням даного програмного продукту, в загальному часі своєї роботи β = 0,5;

- час виконання конкретної функції старого програмного продукту $t_1 = 10$ хв.;
- час виконання конкретної функції нового програмного продукту $t_2 = 6$ хв.

$$Q_1 = \frac{3168 \cdot 60 \cdot 0.5}{10} = 9504 \text{ [умов. од.]}$$

$$Q_1 = \frac{3168 \cdot 60 \cdot 0.5}{6} = 15640 \text{ [умов. од.]}$$

Отже, видно, що застосування нового програмного продукту підвищує продуктивність при виконанні певної функції в 15640/9540 = 1,6 рази.

4.4 Висновки

В даній економічній частині дипломного проекту я розрахував та проаналізував всі можливі витрати для створення даного вебзастосунку та отримав наступні результати:

- витрати на заробітну плату становлять 58000 грн.
- витрати на розробку програмного продукту становлять: 241768,504 грн.
- експлуатаційні витрати при використанні програмного продукту склали: 91544 грн.
- застосування нового програмного продукту підвищує продуктивність в 1.6 рази.

Щоб краще порівняти економічну частину проекту, був проведений пошук в мережі на програмні продукти, що мають в собі аналоги основних функцій даного проекту. В мережі не було знайдено подібних продуктів від

прибуткових комерційних компаній. Інші компанії, які мали більш менш схожі проекти, закривали економічну інформацію про доцільність створення ПЗ від звичайних користувачів. Тому проаналізувати економічну доцільність створення даного рішення за допомогою порівняння різних програмних проектів неможливо.

В ході економічних розрахунків ефективності розробки програмного продукту було встановлено економічну доцільність створення даного програмного продукту.

5 Охорона праці

Охорона праці – це система правових, соціально-економічних, організаційно-технічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці.

5.1 Аналіз умов праці

Для розробки програмного продукту використовувалося житлова кімната, в якому розміщено одне робоче місце. Проаналізую умови праці приміщення, де я працював над програмою: це житлова кімната, яка знаходиться на другому поверсі багатоповерхового будинку. Характеристики приміщення приведені в таблиці 5.1.

Таблиця 5.1 – Характеристики приміщення

Назва приміщення	Висота	Довжина	Ширина	Місце розташування	Орієнтація вікон
Кімната	2,5м	4м	3м	м. Козятин	На північ

У приміщенні розташовано 1 комп'ютер, 1 письмовий стіл, 1 шафа, 1 комод.

Приміщення має природнє та штучне загальне освітлення. Регулювання природним освітленням відбувається за допомогою жалюзів.

Для вентиляції приміщення використовується природна та штучна вентиляція. Для регулювання природного вентилювання, використовується вікно та шахта вентиляції. Штучна вентиляція виконується за допомогою вентилятора та кондиціонера, який знаходиться в сусідній кімнаті.

Підтримка мікроклімату відбувається протягом року. Взимку температура підтримується системою опалювання будинку. В теплі періоду року для контролю мікро клімату використовується кондиціонер та система вентиляції.

Це приміщення має допустимі умови праці.

5.1.1 Шкідливі та небезпечні фактори

У даному приміщені можна виділити такі небезпечні фактори:

- небезпеку ураженням електричним струмом та виникнення пожежонебезпечної ситуації через коротке замикання або необережних дій працівника з електроприладами.
- підвищена запиленість і загазованість повітря робочої зони через постійну роботу електричних компонентів ЕОМ;
- постійна дія шкідливого випромінюваннння ЕОМ. Таке як електромагнітне випромінювання від системного блоку, що призводить до пригнічення нервової системи та порушенням розумової працездатності;
- підвищений рівень шумів на робочому місці. Під час роботи ПК утворюються шуми від роботи системного блоку та принтера. Довготривалість даного явища знижує ефективність роботи працівника;
- постійна робоча поза що супроводжується виконанням маленьких однакових рухів, які призводять до дискомфортів спини, рук, плеч, шиї.
- відсутність чи недостача природнього світла, підвищена яскравість світла, недостатня освітленість робочої зони.

Так як вікна орієнтовані на північ, то впродовж дня природного освітлення вистачає.

5.2 Організаційно-технічні заходи

Загальна площа приміщення становить 12 м2, висота – 2,5 м, приміщення має одне вікно. У приміщенні знаходиться лише одне робоче місце, тому для працюючого в приміщенні припадає: 12 : 1= 12 (м2/ос.) робочої площі. Відповідно до норм та вимогам на одного працівника, який використовує ЕОМ має припадати не менше 6 кв. м. (у нашому випадку 12), а об'єм повітря в приміщенні має бути не менше 15 куб. м ((12 ·2,5)/2=15м3). Висота приміщення – не менше 2,5 м. Отже, нормативу забезпечення працюючих об'ємом повітря та площею в офісі дотримано. Спроектовано одне вікно розмірами (довжина 1,8 м., висота 1,4 м.).

У приміщенні розташовано: 1 комп'ютер, 1 письмовий стіл, 1 крісло офісне, одна шафа для зберігання документів та комод для розміщення навчальних матеріалів.

У кімнаті поєднане природне та штучне освітлення, як загальне, так і місцеве – настільна лампа, яка вмикається при недостатньому природному освітленні з вікна. На вікнах є штори і жалюзі, які можна повністю закрити, щоб виключити відблиски на екрані монітора.

Робочий стіл повинен бути сконструйований так, щоб забезпечити оптимальне розташування обладнання, що використовується на робочій поверхні, з урахуванням кількості, розмірів, елементів дизайну та характеру виконуваної роботи. Гнучкий, рухомий робочий простір є перевагою. Розмір столу визначається всіма компонентами, необхідними для роботи місця, і він повинен забезпечувати безперешкодне переміщення пристрою. Стільниця повинна бути матовою і теплоізоляційною, з низьким відбиттям.

Робочий стілець забезпечує збереження робочого положення в положенні сидячи, і чим довше це положення протягом робочого дня, тим суворішими повинні бути вимоги до створення зручних і правильних робочих сидінь.

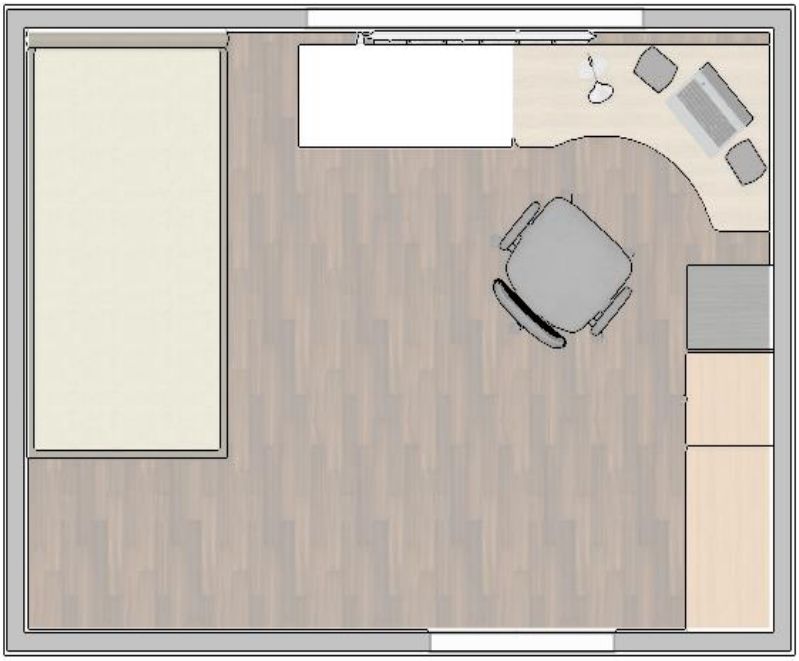

Рисунок 5.1 – План приміщення

5.3 Санітарно-гігієнічні заходи

За гігієнічними нормами це приміщення віднесено до другого класу. Другий клас містить приміщення з прийнятними умовами роботи, але де використовуються спеціальні заходи (опалення, штучне освітлення, ергономічний робочий простір тощо) для забезпечення ідеальних умов роботи для програмного продукту.

В якості санітарно-гігієнічних умов праці наведено підтримання чистоти, вологість 40-60 відсотків, температура в приміщенні 18-24 °С. У кабінетах необхідна припливно-витяжна вентиляція, що в даному випадку здійснюється за рахунок квартирок, дверей, кондиціонера та вентиляційної шахти.

Природне і штучне освітлення в приміщеннях регламентується нормами в залежності від характеру зорової роботи, виду освітлення, фону, контрасту об’єкту.

Дане приміщення відноситься до 2 групи зорової роботи. Робота, що виконується відноситься до зорової роботи високого класу точності.

Для природного освітлення нормованими параметрами є коефіцієнт природної освітленості КПО(ен%), який не повинен бути меншим за 1,5%.

Освітленість при штучному освітленні рівна 300 лк, при загальному освітленні і 450 лк при комбінованому, що відповідає встановленим стандартам. Необхідні заходи щодо дотримання освітленості в нормі передбачають регулярне очищення вікон і світильників від забруднень, своєчасну заміну ламп, що перегоріли.

Шум створюють системний блок, а точніше блок живлення в системному блоці - менше 40 дБА (1 м. від поверхні), джерело безперебійного живлення - менше 40 дБА, динаміки - менше 45 дБА. Відповідно до норми для приміщень даного типу допустимий рівень звукового тиску складає 65 ДБА і реальний рівень шуму не перевищує допустимі значення.

Таблиця 5.2 – Карта умов праці

Фактор	Показник	Значення	
		Нормовані	Дійсні
1	2	3	4
Небезпека	Ступінь небезпеки ураження електричним струмом	Без підвищеної небезпеки	Без підвищеної небезпеки
	Клас пожежонебезпечних приміщень (ПУЕ)	П-IIа,В	П-IIа,В
Мікроклімат	Категорія мікроклімату виробничого приміщення	оптимальні	допустимі
	Клас чистоти приміщення	1	1
	Тип вентиляції	Комбінована	Природня
	Швидкість руху повітря	<0.1м/с	0.05м/с
	За характером навколишнього середовища	Оптимальний	Допустимий

Продовження таблиці 5.2

1	2	3	4
Освітлення	Зорова робота	Високо напружена	Високо напружена
	Характеристика зорової роботи	Середня точність	Середня точність
	Розмір об'єкту розрізнення	0,5 – 1,0 мм	0,8 мм
	Розряд зорової роботи	III	III
	Норм. КПО бокове природне	1,5-3,5%	3%
	Норм. КПО суміщене	–	–
	Освітленість, лк	300лк	300лк
Шуми та вібрації	Категорія вібрації, категорія шуму	Широкосмужна	Широкосмужна
	Тип приміщення	Кімната	Кімната
	Допустимий рівень звуку, тиску	65 дБА	50 дБА
Хімічні речовини	Хімічні речовини	–	–
Статична електрика	ГДР електростатичного поля	До 20В/м	0,5В/м

5.4 Заходи по забезпеченню техніки безпеки

У кімнаті є один комп'ютер та додаткові пристрої. Ці гаджети мають напругу живлення 220 В. Від струмопровідних елементів екрановані всі електричні елементи в приміщенні. Це виключає ризик ураження електричним струмом для персоналу в приміщенні.

Забороняється:

- торкатися проводів живлення та заземлення, а також з'єднувальних кабелів;

- порушувати порядок включення та вимкнення апаратних блоків;
- забороняється класти сторонні предмети на обладнання;
- забороняється працювати за комп'ютером у мокрому одязі та мокрими руками;
- палити в приміщенні, де розташовані комп'ютери.

Перш ніж працювати за комп'ютером, необхідно отримати дозвіл на роботу від уповноваженої особи.

Робота на комп'ютері вимагає:

- дотримання інструкцій з експлуатації обладнання;
- працювати на клавіатурі чистими сухими руками, не натискаючи клавіш без потреби чи навмання;
- відповідне виконання завдання з програмним забезпеченням.

У цій кімнаті є багато електричних елементів, які можуть стати причиною пожежі, якщо вони вийдуть з ладу. Крім того, завжди існує ризик короткого замикання. Щоб усунути ці небезпеки, використовуються запобіжники. Усі пристрої також проходять регулярні технічні перевірки та експлуатуються відповідно до чинного законодавства.

До організаційно-технічних заходів щодо організації електронебезпеки належить правильне використання електроприладів, а також надання часу всім електроприладам на відпочинок.

Не використовуйте пристрої ручної роботи для підключення ПК до джерела живлення. Рекомендується використовувати заземлені розетки. Корпус комп'ютера та периферійні пристрої не повинні стикатися з водою (або іншими рідинами, які несуть електрику).

5.5 Протипожежні заходи

У цій кімнаті може загорітися папір, дерев’яні меблі, паркет. Основними причинами пожежі можуть бути коротке замикання та перевантаження мережі.

Рекомендується ізолювати проводку та надійно закріпити, щоб не було нагрівання системи ЕОМ. Також необхідно регулярно проводити зовнішній огляд пристроїв на наявність механічної несправності, що може призвести до ураження електричним струмом.

Якщо потрібно, наберіть 101, щоб зв’язатися з пожежною службою.

5.6 Розробка заходів захисту від негативної дії ультрафіолетового випромінювання на робочих місцях

Проблема ультрафіолетового випромінювання як виробничого та екологічного чинника обумовлена широким використанням джерел постачання в народному господарстві, збільшенням рівнів сонячного випромінювання у зв'язку зі зменшенням озонового шару, зростанням кількості захворювань, зокрема злоякісних і доброякісних пухлин шкіри, та інших порушень стану здоров'я, що викликаються ультрафіолетової радіацією.

При тривалій відсутності УФВ в організмі розвивається «світлове голодування». Тому воно необхідно для нормальної життєдіяльності людини. Однак, при тривалому впливі великих доз УФВ можуть наступити серйозні поразки очей і шкіри. Зокрема, це може призвести до розвитку раку шкіри, кератитів (запалень рогівки) і помутніння кришталика очей (фотокератиту, який характеризується прихованим періодом від 0,5 до 24 годин). Для профілактики несприятливих наслідків, викликаних дефіцитом УФВ,

використовують сонячне випромінювання, влаштовуючи солярії, інсоляцію приміщень, а також застосовуючи штучні джерела УФВ.

На промислових підприємствах джерелами ультрафіолетових випромінювань є дуга електрозварювання, ртутно-кварцові лампи, лазери, інші прилади та установки. Формування й вплив на працюючих оптичного випромінювання в ультрафіолетової області відбувається при електрогазозварювальних процесах, на роботах з плазменними технологіями (різка металу, напилювання, наплавлення металу), при використанні різних світильників та випромінювачів з кварцовими, ртутними, галогенними лампами ультрафіолетове сушіння, установки для знезараження повітря, поверхонь та води, різні медичні та інші випромінювачі (перукарське устаткування, манікюрні лампи, солярії та інші).

Захист від УФВ досягається:

- екрануванням робочих місць;
- засобами індивідуального захисту;
- спеціальним фарбуванням приміщень і раціональним розташуванням робочих місць.

При визначенні захисної відстані використовують дані безпосередніх вимірів у конкретних умовах. Найбільш раціональним методом захисту є екранування джерел випромінювання різними матеріалами і світлофільтрами. Екрани виконуються у виді щитів, ширм, кабін. Повний захист від УФВ всіх областей забезпечує флінтглас (скло, яке вміщує оксид свинцю).

У якості ЗІЗ використовують спецодяг (куртки, брюки, рукавички, фартухи), із спеціальних тканин, що не пропускають УФВ (льняні, бавовняні, поплін); захисні окуляри та щитки із світлофільтрами. Для захисту рук застосовують мазі із вмістом речовин, що служать світлофільтрами (салол, саліцилово-метиловий ефір).

Стіни і ширми у цехах фарбують у світлі кольори (сірий, жовтий, блакитний), застосовуючи цинкове чи титанове білило для поглинання УФВ.

Висновки

В результаті роботи над завданням дипломного проекту було створено рішення Microsoft Visual Studio 2019 Community з метою реалізації менеджера обліку працівників, яке включає проекти бібліотек класів ASP.NET Core та компоненти Bootstrap.

Програмний код реалізовано мовами C# та HTML.

Розробка базувалась на використанні заначної кількості сучасних технологій та підходів: односторінкового інтерфейсу та багаторівневої архітектури, REST API, WebAPI і т.д.

У якості бази даних використовується SQL Server Express. В процесі виконання було реалізовано введення та виведення, друк даних, підсистема CRUD (Create, Read, Update, Delete), а також можливість роботи із MDF файлами. В проєкті використані SQL-запити для відбору, редагування та забезпечення цілісності даних.

Даний додаток надає можливості для створення навчального курсу на базі проекту «Менеджер обліку працівників».

З метою забезпечення підтримки додатку було реалізовано модульну архітектурну на всіх рівнях. Для контрольованого доступу до бази даних було створено рольову модель користувачів.

Список використаних джерел

1. germasky.monday.com URL: https://germasky.monday.com/ (дата звернення: 25.04.2022).
2. SOFTWARE DEVELOPMENT USING THE .NET FRAMEWORK URL: https://confcontact.com/2013_04_04_dnu/35_Shytik.htm (дата звернення: 27.04.2022).
3. .NET vs. .NET Framework for server apps URL: https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server (дата звернення: 30.04.2022).
4. ASP.NET Core fundamentals overview URL: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/?view=aspnetcore-6.0&tabs=windows (дата звернення: 01.05.2022).
5. Тепляков С. Паттерны проектирования на платформе .NET. СПБ “Питер”, 2018 р., 320 с.
6. Tag Helpers in ASP.NET Core URL: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/intro?view=aspnetcore-6.0 (дата звернення: 10.05.2022).
7. METANIT.COM «Bootstrap в ASP.NET MVC 5», URL: https://metanit.com/sharp/mvc5/14.1.php (дата звернення: 15.05.2022).
8. SQL Server Express LocalDB URL: https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-express-localdb?view=sql-server-ver16 (дата звернення: 20.05.2022).
9. Azure documentation URL: https://docs.microsoft.com/uk-ua/azure/?product=popular (дата звернення: 27.06.2022).
10. Pay as you go with Azure URL: https://azure.microsoft.com/en-us/pricing/purchase-options/pay-as-you-go/ (дата звернення: 29.06.2022).

Додатки

Додаток А

Лістинг А – Код елемента навігаційної стрічки

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="AdminMaster.master.cs"
Inherits="OnlineWorker.View.Admin.AdminMaster" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<link rel="stylesheet"
href="../../Asset/Lib/Bootstrap/css/bootstrap.min.css"/>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
   <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand">
      <div class="logo-image">
        <img src="../../Asset/Images/icon_admin.png"
        style="width: 46px;
        height: 46px;
        border-radius: 50%;
        overflow: hidden;
        margin-top: -6px;">
      </div>
    </a>
    <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse"
id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">

        <li class="nav-item">
          <a class="nav-link" href="Workers.aspx">Workers</a>
        </li>
            <li class="nav-item">
          <a class="nav-link" href="Seats.aspx">Seats</a>
        </li>
            <li class="nav-item">
```

```
          <a class="nav-link"
href="Operators.aspx">Operators</a>
        </li>
            <li class="nav-item">
          <a class="nav-link"
href="Dashboard.aspx">Dashboard</a>
        </li>
      </ul>

      <a href="../Login.aspx" class="btn btn-outline-primary my-
2 my-sm-0">Logout</a>


    </div>
  </div>
</nav>
    <form id="form1" runat="server">
    <div>
        <asp:ContentPlaceHolder ID="Mybody" runat="server">

        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

Додаток Б

Лістинг Б – Код класу Login.aspx.cs

```
    public partial class Login : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Con = new Models.Functions();
        }
        public static string OperName="";
        public static int OperKey;
        //Add this Method
        Models.Functions Con;
        protected void SaveBtn_Click(object sender, EventArgs e)
        {

            if (AdminRadio.Checked)
            {
                if(EmailId.Value == "Admin@gmail.com" &&
UserPasswordTb.Value == "Admin")
                {
                    Response.Redirect("Admin/Dashboard.aspx");
                }else
                {
                    InfoMsg.InnerText = "Invalid Admin!";
                }
            }else
            {
                string Query = "select
OperId,OperName,OperEmail,OperPassword from OperatorTbl where
OperEmail = '{0}' and OperPassword = '{1}'";
                Query = string.Format(Query, EmailId.Value,
UserPasswordTb.Value);
                DataTable dt = Con.getData(Query);
                if(dt.Rows.Count == 0)
                {
                    InfoMsg.InnerText = "Invalid User!";

                }else
                {
                    OperName = EmailId.Value;
                    OperKey =
Convert.ToInt32(dt.Rows[0][0].ToString());
                    Response.Redirect("Operator/Billing.aspx");}
            }
        }
    }
}
```

Додаток В

Лістинг В – Код класу Billing.aspx.cs

```
  public partial class Billing : System.Web.UI.Page
    {
        Models.Functions Con;
        DataTable dt = new DataTable();
        int Operator = Login.OperKey;
        protected void Page_Load(object sender, EventArgs e)
        {
            Con = new Models.Functions();
            ShowWorkers();
            if (!this.IsPostBack)
            {
                DataTable dt = new DataTable();
                dt.Columns.AddRange(new DataColumn[5] {
                    new DataColumn("ID"),
                    new DataColumn("Worker"),
                    new DataColumn("Price"),
                    new DataColumn("Amount of hours"),
                    new DataColumn("Total")
                });
                ViewState["Bill"] = dt;
                this.BindGrid();
            }
        }
        private void InsertBill()
        {
            try
            {
                string Query = "insert into BillTbl
values('{0}','{1}',{2})";
                Query = string.Format(Query,
System.DateTime.Today, Operator, Amount);
                Con.SetData(Query);
                ErrMsg.InnerText = "Bill Inserted!";
            }
            catch (Exception Ex)
            {
                ErrMsg.InnerText = Ex.Message;
            }
        }
        int GrdTotal;
        protected void BindGrid()
        {
            BillGV.DataSource = (DataTable)ViewState["Bill"];
            BillGV.DataBind();
        }
```

```
        public static int Amount;
        //Add this Method
        public override void VerifyRenderingInServerForm(Control
control)
        {
        }
        private void ShowWorkers()
        {
            string Query = "select WorkerId as Id, WorkerName as
Name, WorkerSeat as Seat, WorkerPrice as Price from WorkerTbl";
            WorkerGV.DataSource = Con.getData(Query);
            WorkerGV.DataBind();
        }
        int Key = 0;
        protected void WorkerGV_SelectedIndexChanged(object
sender, EventArgs e)
        {
            WorkerNameTb.Value =
WorkerGV.SelectedRow.Cells[2].Text;
            WorkerPriceTb.Value =
WorkerGV.SelectedRow.Cells[4].Text;
            if (WorkerNameTb.Value == "")
            {
                Key = 0;
            }
            else
            {
                Key =
Convert.ToInt32(WorkerGV.SelectedRow.Cells[1].Text);
            }
        }
        // private void UpdateStock()
        //{
        //    int newQty;
        //    newQty = Convert.ToInt32(WorkerQtyTb.Value);
        //    string Query = "update WorkerTbl set WorkerQty =
'{0}' where WorkerId = {1}";
        //    Query = string.Format(Query,
newQty,WorkerGV.SelectedRow.Cells[1].Text);
        //    Con.SetData(Query);
        //    ShowWorkers();
        //    ErrMsg.InnerText = "Quantity Updated!";
        //}
        protected void AddToBillBtn_Click(object sender,
EventArgs e)
        {
            try
            {
                if (WorkerNameTb.Value == "" ||
WorkerPriceTb.Value == "" || WorkerQtyTb.Value == "")
                {
```

```
                    ErrMsg.InnerText = "Missing Data";
                }
                else
                {
                    int total =
Convert.ToInt32(WorkerQtyTb.Value) *
Convert.ToInt32(WorkerPriceTb.Value);
                    DataTable dt = (DataTable)ViewState["Bill"];
                    dt.Rows.Add(BillGV.Rows.Count + 1,
                        WorkerNameTb.Value.Trim(),
                        WorkerPriceTb.Value.Trim(),
                        WorkerQtyTb.Value.Trim(),
                        total
                        );
                    ViewState["Bill"] = dt;
                    this.BindGrid();
                    GrdTotal = 0;
                    for (int i = 0; i <= BillGV.Rows.Count - 1;
i++)
                    {
                        GrdTotal = GrdTotal +
Int32.Parse(BillGV.Rows[i].Cells[4].Text.ToString());
                    }
                    Amount = GrdTotal;
                    GrdTotTb.InnerHtml = "$" + GrdTotal;
                    WorkerNameTb.Value = string.Empty;
                    WorkerPriceTb.Value = string.Empty;
                }
            }
            catch (Exception Ex)
            {
                ErrMsg.InnerText = Ex.Message;
            }
        }
        protected void PrintBtn_Click(object sender, EventArgs
e)
        {
            InsertBill();
        }
        protected void ResetBtn_Click(object sender, EventArgs
e)
        {
            try
            {

                DataTable dt = (DataTable)ViewState["Bill"];
                dt.Rows.Clear();
                ViewState["Bill"] = dt;
                this.BindGrid();
                ViewState["Bill"] = dt;
                this.BindGrid();
```

```
                GrdTotal = 0;
                for (int i = 0; i <= BillGV.Rows.Count - 1; i++)
                {
                    GrdTotal = GrdTotal +
Int32.Parse(BillGV.Rows[i].Cells[4].Text.ToString());
                }
                Amount = GrdTotal;
                GrdTotTb.InnerHtml = "$" + GrdTotal;
                WorkerNameTb.Value = string.Empty;
                WorkerPriceTb.Value = string.Empty;
            }
            catch (Exception Ex)
            {
                ErrMsg.InnerText = Ex.Message;
            }
            ;
        }
    }
}
```

Додаток Г

Лістинг Г – Реалізація сторінки Dashboard

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace OnlineWorker.View.Admin
{
    public partial class Dashboard : System.Web.UI.Page
    {
        Models.Functions Con;
        protected void Page_Load(object sender, EventArgs e)
        {
            Con = new Models.Functions();
            GetWorkers();
            GetSeats();
            SumSell();
            GetOperators();
            GetOperator();
        }
        private void GetOperator()
        {
            string Query = "Select * from OperatorTbl";
            OperCb.DataTextField =
Con.getData(Query).Columns["OperName"].ToString();
            OperCb.DataValueField =
Con.getData(Query).Columns["OperId"].ToString();
            OperCb.DataSource = Con.getData(Query);
            OperCb.DataBind();
        }
        private void GetWorkers()
        {
            string Query = "Select Count(*) from WorkerTbl";
            WorkerNumTb.InnerText =
Con.getData(Query).Rows[0][0].ToString();
            WorkerNumTb.DataBind();
        }
        private void GetSeats()
        {
            string Query = "Select Count(*) from SeatTbl";
            SeatNumTb.InnerText =
Con.getData(Query).Rows[0][0].ToString();
            SeatNumTb.DataBind();
        }
```

```
        private void GetOperators()
        {
            string Query = "Select Count(*) from OperatorTbl";
            OperNumTb.InnerText =
Con.getData(Query).Rows[0][0].ToString();
            OperNumTb.DataBind();
        }
        private void SumSell()
        {
            string Query = "Select Sum(Amount) from BillTbl";
            FinanceTb.InnerText ="$  " +
Con.getData(Query).Rows[0][0].ToString();
            FinanceTb.DataBind();
        }
        private void SumPayByOperator()
        {
            string Query = "Select Sum(Amount) from BillTbl
where Operator = " + OperCb.SelectedValue.ToString()+" ";
            TotalTb.InnerText = "$  " +
Con.getData(Query).Rows[0][0].ToString();
            TotalTb.DataBind();
            Con.getData(Query);
            GetOperator();
        }
        protected void OperCb_SelectedIndexChanged(object
sender, EventArgs e)
        {
            SumPayByOperator();


        }
    }
}
```