КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

ПРОГРАМНИЙ КОМПЛЕКС ДЛЯ АНАЛІЗУ ЯКОСТІ ПОСЛУГ ІНТЕРНЕТ-ПРОВАЙДЕРІВ ТА ДОСТУПНОСТІ РЕСУРСІВ ЛОКАЛЬНОЇ МЕРЕЖІ

Випускна кваліфікаційна робота

студентки 4 курсу

Спеціальність: 123 «Комп'ютерна інженерія»

Оксани ОМЕЛЬЧУК

Науковий керівник

доцент кафедри комп'ютерної інженерії

Сергій ЗАГОРОДНЮК

Рецензент

канд. фіз.-мат. наук

доцент кафедри геоінформатики

ННІ «Інститут геології» Всеволод ДЕМИДОВ

До захисту допускаю: Завідувач кафедри

Юрій БОЙКО

Ухвалено на засіданні кафедри "____"_____2022 р., протокол № ____

Київ 2022

РЕФЕРАТ

Обсяг роботи 50 сторінок, 24 ілюстрації, 16 джерел посилань.

WINDOWS-СЛУЖБА, ASP.NET CORE, БАЗА ДАНИХ, ВЕБ-САЙТ, SQL-ЗАПИТ, ФАЙЛОВИЙ ПОТІК, INTERNET INFORMATION SERVICE, ПРОТОКОЛ TCP.

Об’єктом роботи є процес моніторингу доступності локальної мережі та зовнішніх Інтернет-ресурсів на основі статистичних даних. Предметом роботи є програмний комплекс для формування статистики доступності зовнішніх Інтернет-ресурсів та локальної мережі.

Метою роботи є створення програмного комплексу для накопичення статистичних даних про доступність локальної мережі та зовнішніх Інтернет-ресурсів та створення веб-сайту для відображення накопиченої інформації.

Методи розроблення: розробка служби на основі надсилання запитів на вузли, створення програми з веб-інтерфейсом на платформі ASP.NET Core. Інструменти розроблення: програмне забезпечення Microsoft Visual Studio з середовищем розробки, яке є інтегрованим, для створення програмного забезпечення та інших програмних застосунків, мова програмування C#.

Результати роботи: виконано загальний огляд переваг та недоліків розробки Windows-служб та проектів ASP.NET Core MVC. Створено програмний комплекс, який складається з служби для моніторингу доступності локальної мережі та Інтернет-ресурсів та веб-сайту для формування та відображення отриманих статистичних даних за певний період у вигляді таблиці результатів. Даний програмний продукт дозволяє перевіряти доступність ресурсів Інтернету та локальної мережі та накопичувати отримані результати.

Розроблений продукт «Програмний комплекс для аналізу якості послуг Інтернет-провайдерів та доступності ресурсів локальної мережі» має широкий спектр галузей використання, тому що може бути застосований як на великих підприємствах так і для особистого користування.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. З моменту виникнення та появи Всесвітньої мережі Інтернет в Україні, загальна кількість Інтернет-провайдерів, організацій, що забезпечують доступ до мережі, збільшується кожного дня. Постачальники послуг Інтернет, на даний час, мають дуже багато тарифів, додаткових опцій та функцій, які змушують користувача вибрати саме цю організацію, проте не всі провайдери доброчесно виконують свої обов’язки перед своїми клієнтами. І через це у користувачів з’являються проблеми з доступом до мережі Інтернет, а це є досить критичним у сьогоденні, тому що багато компаній та організацій працюють в онлайн-режимі і через виникнення збоїв у роботі мережі працівники не можуть виконувати задану для них роботу вчасно. Тож, постає проблема з ключовим параметром для аналізу якості послуг Інтернет-провайдерів – доступністю до ресурсів мережі Інтернет.

Актуальність роботи. Нерідко постає проблема з вибором доброчесного Інтернет-провайдера, що забезпечує стабільний доступ до мережі Інтернет, та перевірки вибраної організації на чесне виконання обов’язків. Отже, доцільно створити службу для моніторингу доступності локальної мережі та Інтернет-ресурсів, а також веб-програма для зручного перегляду отриманих статистичних даних за певний період роботи.

Мета та завдання роботи. Розробити програмний комплекс для формування статистики доступності зовнішніх Інтернет-ресурсів та ресурсів локальної мережі школи І-ІІІ ступенів №286 м. Києва. Для здійснення поставленої мети сформульовані такі завдання:

- Дослідити переваги та недоліки системних служб операційної системи Windows та веб-програми платформи ASP.NET MVC.
- На комп’ютері секретаря вказаного закладу освіти створити і зареєструвати в операційній системі Windows 10 моніторингову службу для

накопичення даних щодо доступності ресурсів локальної мережі та Інтернет-ресурсів.

- Створити веб-програму для підрахунку параметрів доступності накопиченої інформації.
- Опублікувати створену веб-програму на веб-сервері Internet Information Service, що є вбудованим компонентом операційної системи Windows 10.

Об’єкт, засоби та методи розробки. Об’єктом розроблення проекту «Програмний комплекс для аналізу якості послуг Інтернет-провайдерів та доступності ресурсів локальної мережі» є процес моніторингу, накопичення даних для перевірки доступності ресурсів Інтернету та локальної мережі та графічного відображення отриманої інформації.

Перед розробкою програмного комплексу був проведений аналіз програм зі схожим функціоналом, визначено переваги та недоліки даних застосунків, а також був розроблений алгоритм роботи створеної служби та продумано функціонал користувацького інтерфейсу для веб-сайту.

Під час розробки програмного комплексу проведені тестування роботи служби та програми з веб-інтерфейсом з метою покращення функціоналу та вирішення проблем, які виявилися.

Для розробки програмного продукту було використано інтегроване середовище розробки Microsoft Visual Studio, яке стабільно працює та має широкий функціонал для зручної роботи над проектом. Код програми написаний мовою програмування C#, що є досить поширеною у використанні та має багато бібліотек, що значно спрощує розробку програми. Для створення веб-програми використано платформу ASP.NET Core, що має велику кількість переваг у роботі та є досить стабільною для використання.

Можливі сфери використання Даний продукт «Програмний комплекс для формування статистики доступності локальної мережі та зовнішніх Інтернет-

ресурсів» може бути використаний як на підприємстві так і в особистому користуванні, що забезпечує велику сферу використання.

РОЗДІЛ 1 ОСНОВНІ ВІДОМОСТІ

1.1 Загальні відомості про Windows-служби

Служби Windows – це програми, які в основному працюють в фоновому режимі та виконують задані їм завдання. Служби керують великим спектром функцій, включаючи мережі підключення, облікові дані користувача та багато інших функцій. Служби можуть бути системними, мережевими або локальними в залежності від того, якими ресурсами вони керують. Service є невід'ємною складовою операційних систем, оскільки забезпечують стабільну роботу та керують процесами даних систем.

Служби Windows можуть працювати в декількох режимах:

- Заборонений до запуску;
- Ручний режим(служба в даному режимі працює за запитом);
- Автоматичний запуск при завантаженні системи;
- Обов'язкова служба чи драйвер (дана служба автоматично запускається та немає можливості автоматичного вимкнення користувачем).

Для запуску, контролю та перегляду зареєстрованих служб в системі Windows використовується програма Service Control Manage. Дана програма відображає список служб, які зареєстровані в операційній системі, та дозволяє керувати даними службами, змінюючи їхні налаштування та запускати чи зупиняти дані застосунки.

Права користувача.

Служби Windows початково працюють від користувача «LocalSystem». Даний користувач має найбільше прав у системі, загальна кількість яких навіть переважає права надані обліковому запису Administrator. «LocalSystem» є «віртуальним» користувачем тому, якщо програмі потрібно зберігати дані пов'язані з користувачем, то задану службу необхідно запускати від користувача, попередньо вказавши пароль до вказаного облікового запису. Видалити та

зареєструвати службу можна тільки від користувача, який має адміністративні права у системі.

Transmission Control Protocol (TCP) – один з основних протоколів передачі даних Інтернету. Головною функцією даного протоколу є управління передачею даних Інтернету. Даний протокол є поширеним у використанні, тому що характеризується стабільною довгостроковою роботою.

У цьому протоколі можна виділити 4 рівні, а саме: канальний (на даному етапі інформація закодовується та розділяється на пакети для подальшого відправлення), міжмережевий (будується маршрут для передачі сформованих пакетів на вказані IP-адреси з використанням маски), транспортний (відповідає за доставлення пакета даних з проведенням перевірки доставлення, якщо пакет не був отриманий, то надсилається запит на повторну відправку) та прикладний (забезпечує підтримку зв'язку між хостами та здійснює перетворення інформації, яка була отримана). За допомогою цього протоколу можна надсилати запити до певних вузлів, вказавши порт, комбінація IP-адреси та порта називається сокетом.

Переваги протоколу TCP:

- Надійність передачі даних, оскільки отримується підтвердження отримання даних.
- Впорядкованість відісланих даних, оскільки даний протокол нумерує пакети, які надсилає та забезпечує правильну послідовність пакетів.

1.2 Загальні характеристики ASP.NET MVC

ASP.NET MVC – платформа, за допомогою якої можна створювати веб-програми та веб-сайти з використанням шаблона Model View Controller (MVC).

MVC шаблон являє собою взаємодію трьох компонентів, а саме: представлення, контролера та моделі.

Контролер (controller) – це клас, за допомогою якого починає працювати створена програма. Даний клас встановлює взаємодію між такими компонентами, як: модель та представлення. Під час отримання введених користувачем даних, контролер відповідно до внутрішньої логіки проводить звернення до моделі та створює графічне представлення.

Представлення (View) – це користувацький інтерфейс, тобто візуальна частина програми, за допомогою якої кінцевий користувач зможе взаємодіяти з програмою чи сайтом.

Модель (Model) – це група класів, в яких описана логіка використання та взаємодії з даними. У таких класах зазвичай вказуються атрибути даних, для подальшої роботи з отриманою інформацією та її представлення на сторінці розробленого сайту.

ASP.NET Core MVC – це легка для використання платформа, яка має відкритий вихідний код та широкі можливості для тестування, яка розроблена для використання ASP.NET Core.

ASP.NET Core MVC являє собою метод створення динамічних веб-сайтів, який базується на використанні шаблонів, які полегшують процес створення веб-сторінок та спрошують подальше виправлення помилок, якщо вони виникнуть. Для зручності роботи з проектом такого типу створено окремі каталоги для моделі, контролера та представлення з встановленням взаємозв’язку між ними, що досить спрощує створення складних веб-програм. ASP.NET Core MVC дозволяє опрацьовувати вхідні дані, які вводить користувач, передаючи їх у контролер у формі параметрів для методів.

Переваги використання ASP.NET Core:

- Можливість розробки веб-програми для різних операційних систем, зокрема ОС Windows, Linux, macOS.
- Відкритий вихідний код.
- Razor Pages – це новий впроваджений елемент, який оптимізує

програмні сценарії, які створені на основані на веб-сторінках, що значно спрощує створення веб-інтерфейсу на сторінці.

- Обробка значно більшої кількості запитів в порівнянні з іншими платформами.
- Спрощене керування залежностями та конфігурацією проекту.
- Розповсюдження пакетів платформи через NuGet.

1.3 Internet Information Service (IIS)

Internet Information Service (IIS) – це досить гнучкий веб-сервер від корпорації Microsoft, який працює в операційній системі Windows для обслуговування HTML- сторінок чи файлів. Веб-сервер IIS дозволяє отримувати від віддалених комп'ютерів користувачів запити та відправляти відповідні відповіді. Ця базова функціональність дозволяє обмінюватися певною інформацією та доставляти її через глобальні мережі(WAN) чи локальні мережі (LAN). Веб-сервер дозволяє програмі обробляти повідомлення, які отримуються через певні TCP порти (за замовчуванням використовується для трафіку http 80 порт та для https трафіку 443 порт). Найчастіше IIS використовується для розміщення ASP.NET програм та статичних сайтів.

IIS працює через багато стандартних протоколів та мов. HTML використовується для створення користувацьких елементів (текст, кнопки, розміщення зображень). Також IIS використовує такі протоколи для передачі гіпертексту: HTTP, HTTPS. HTTP- це протокол передавання гіпертексту, який є основою всесвітньої павутини і використовується для завантаження веб-сторінок за допомогою гіпертекстових посилань. HTTP- це протокол прикладного рівня, метою роботи якого є передача інформації між мережевими пристроями. Протокол HTTP використовує у своїй роботі технологію «клієнт-сервер», роботу якого можна описати, як відправлення запиту клієнтом, який являється

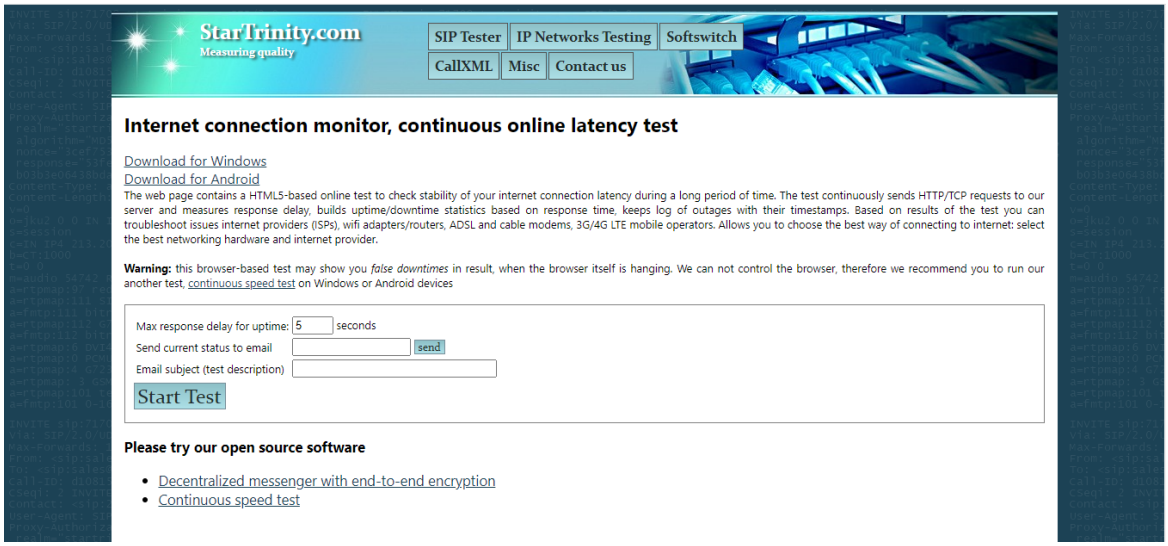
ініціатором з’єднання, отримання сервером запиту та відправлення результату назад клієнту.

HTTPS – протокол передачі гіпертексту, який є безпечним та використовує шифрування за допомогою протоколів SSL та TLS.

РОЗДІЛ 2 АНАЛІЗ ПРОГРАМ ЗІ СХОЖИМ ФУНКЦІОНАЛОМ

2.1 StarTrinity.com

StarTrinity.com- веб-сторінка на якій розміщено онлайн-тест на основі HTML5 для перевірки затримки Інтернет-з'єднання протягом довготривалого періоду часу. Даний тест безперервно відправляє HTTP/ TCP запити на сервер та вимірює затримку в отриманні відповіді, будує статистику безвідмовної роботи часу простою на основі часу відповіді, веде журнал перебоїв з їх часовими мітками. За результатами тесту можна усувати неполадки Інтернет-провайдерів (ISP), адаптерів або маршрутизаторів WI-FI, ADSL кабельних модемів, операторів мобільної мережі 3G / 4G LTE. Дозволяє вибрати кращий спосіб підключення до Інтернету: вибрати краще мережеве обладнання і Інтернет-провайдера.



«Рисунок 1 - Графічне представлення StarTrinity.com»

Переваги даної програми:

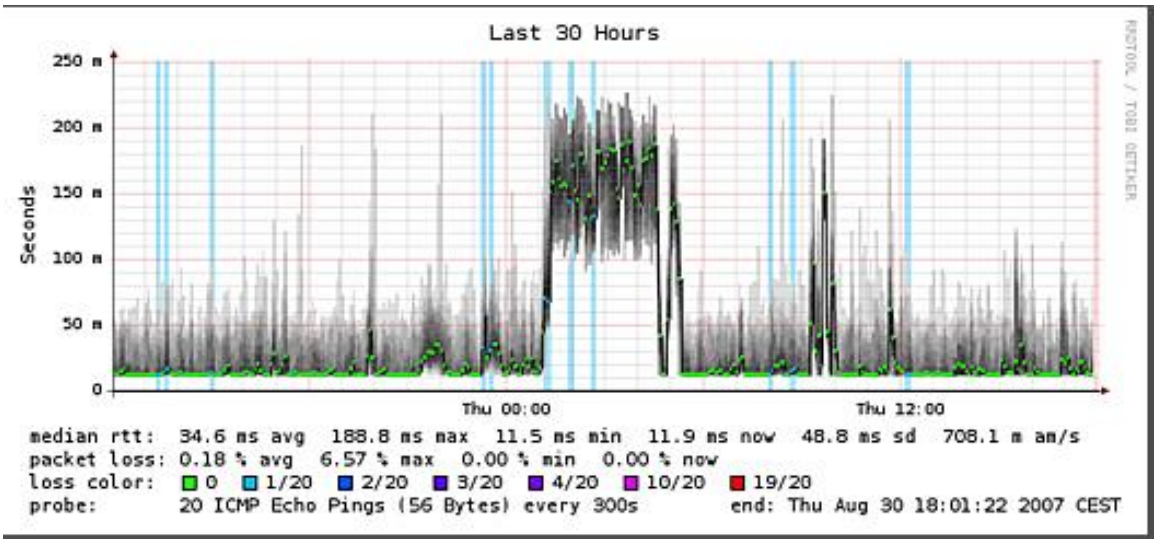
- Має досить зрозумілий та простий у використанні користувацький інтерфейс.
- Працює на основі TCP-запитів.
- Можна встановити дану програму для операційної системи Windows.

Недоліки:

- Немає докладного опису послідовності дій при встановленні даної програми, через, що виникають проблеми зі встановленням.
- Цей тест на основі браузера може показати помилкові затримки в результаті некоректності роботи браузера.
- З’являються перебої в роботі програми в результаті чого, відбуваються відображення некоректних даних.

2.2 SmokePing

SmokePing- це програма, яка вимірює затримки. Даний інструмент може вимірювати, зберігати і відтворювати затримку, розподілення затримки та втрату пакетів. SmokePing використовує RRDtool для відображення довгострокового сховища даних і створення красивих графіків, які відображають найбільш нову інформацію про стан кожного мережевого підключення.



«Рисунок 2 - Графік в програмі SmokePing для аналізу отриманих результатів»

Основні особливості програми:

- Вимірює затримку і її зміну.
- Широкий вибір зонтів, від простого пінгу до веб-запитів і протоколів, які налаштовуються.
- Розширена система сигналізації, яка спрацьовує по шаблонах затримки, які налаштовуються.
- Модель розгортання «ведучий або ведений» для паралельного виконання вимірів із кількох джерел.
- Графічна навігація на основі Ajax.
- Режим графіка при якому показуються спочатку найцікавіші графіки.
- Написано на Perl для полегшення покращень.
- Повністю задокументований.

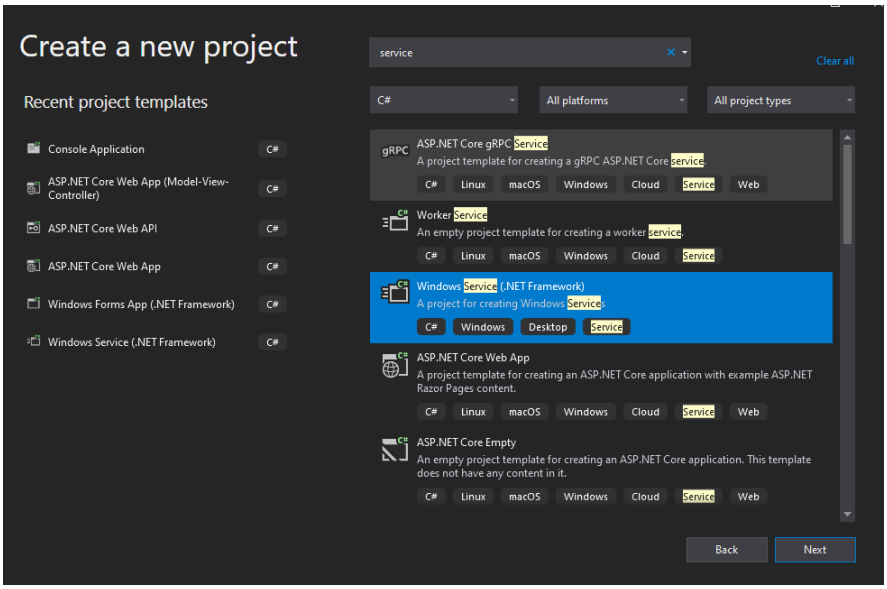
При тестуванні програми значних недоліків не було виявлено, тільки користувацький інтерфейс досить складний для початкового використання та потребує детального вивчення.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Створення та реєстрація Windows-служби

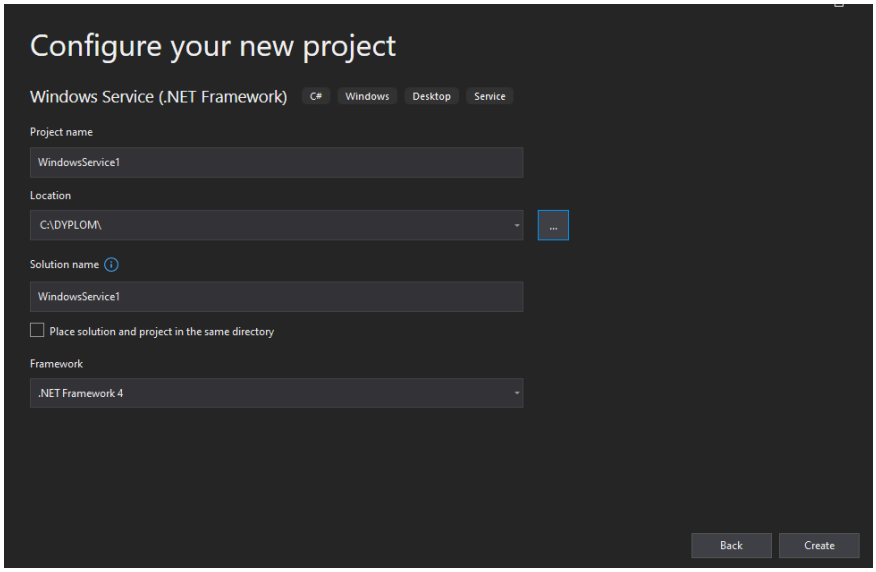
Для створення служби Windows у середовищі розробки Microsoft Visual Studio потрібно виконати такі кроки:

1. Створити новий проект Windows Service (.NET Framework).



«Рисунок 3 - Створення Windows Service»

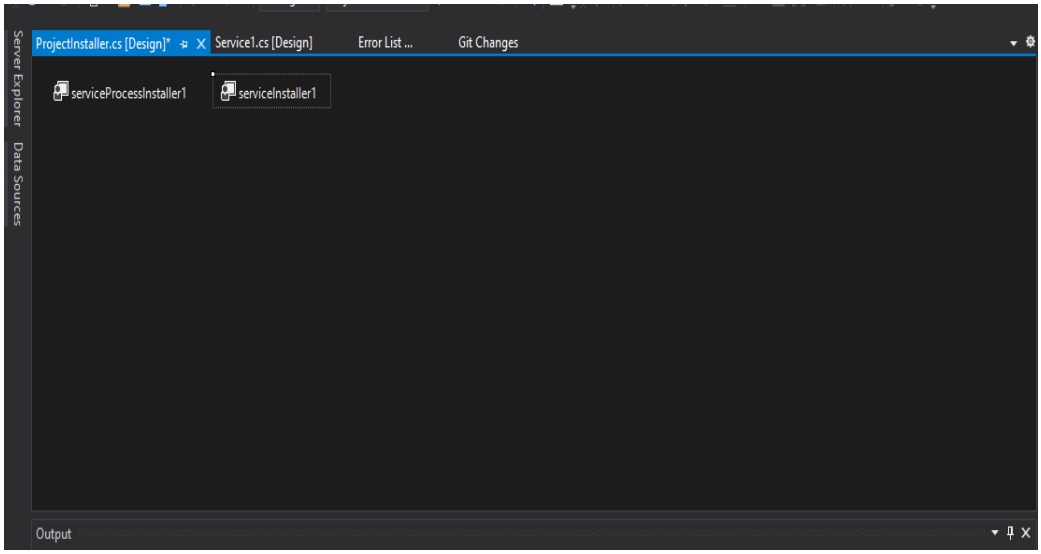
2. На наступному кроці потрібно вказати назву проекту та шлях розташування проекту на комп’ютері.



«Рисунок 4 - Вікно для вказання назви та місця розміщення»

3. Створення інсталятора Windows Service для реєстрації в диспетчері управління службами.

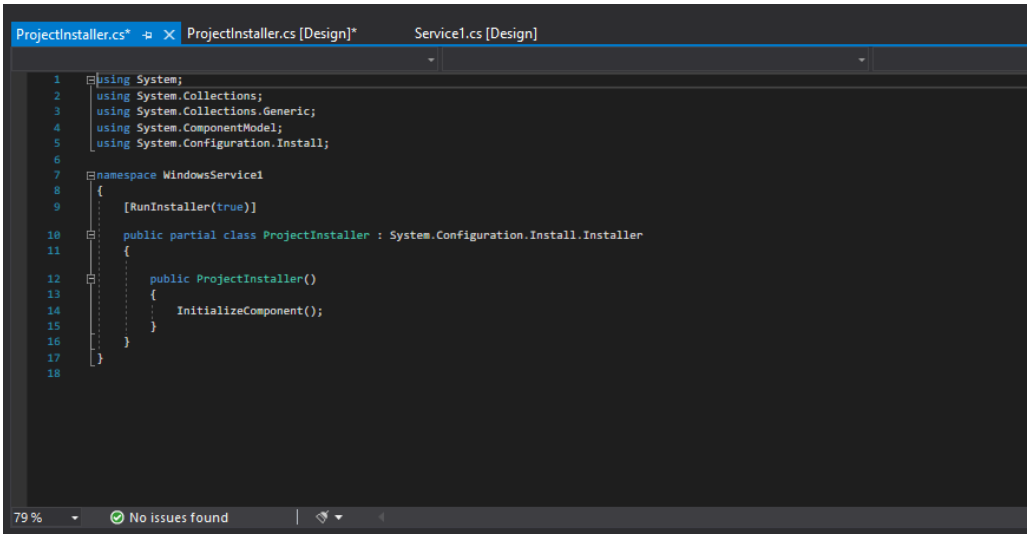
Необхідно натиснути правою кнопкою миші у вікні ProjectInstaller.cs та вибрати пункт “Add installer”.



«Рисунок 5 - Додавання до проекту інсталяторів»

4. Перехід до написання коду для створеної служби.

Щоб відкрити код потрібно у вікні ProjectInstaller.cs натиснути правою кнопкою миші та вибрати «View code», після виконання цієї дії відкриється вікно для написання коду.



« Рисунок 6 - Вікно для написання коду для Windows Service»

5. Розробка логіки програми.

Код програми Windows Service.

```
namespace WS2
{
public partial class Service1 : ServiceBase
{
public Service1()
{
InitializeComponent();
}
```

Метод, який виконується при старті Windows Service та забезпечує виконання функції для періодичного надсилання запитів на вузли.

```
protected override void OnStart(string[] args)
{
```

Шлях до файлу в якому зберігаються зовнішні вузли.

```
string path = @"C:\DYPLOM\ExternalIP.txt";
if (!File.Exists(path))
{
```

Якщо файл не існує, то створюємо файл та проводимо запис до файлу.

```
using (StreamWriter F = new StreamWriter("C:\\DYPLOM\\ExternalIP.txt", true))
{
```

Масив для збереження назв вузлів.

```
string[] hostname = { "mail.ukr.net" };
```

Масив для збереження портів.

```
int[] port = {80};
```

Циклічний запис у файл даних.

```
for (int i = 0; i < hostname.Length; i++){
```

Запис у файл вузлів та портів по яких будуть виконуватися запити.

```
F.WriteLine(hostname[i] + " " + port[i]);
}
}
}
```

Шлях до файлу в якому зберігаються внутрішні вузли.

```
string path2 = @"C:\DYPLOM\InternalIP.txt";
if (!File.Exists(path2))
{
```

Якщо файл не існує, то створюємо файл та розпочинаємо запис до даного файлу.

```
using (StreamWriter F = new StreamWriter(path2, true))
{
```

Запис у файл внутрішніх вузлів та портів по яких будуть виконуватися запити.

```
F.WriteLine("192.168.7.1" + " " + "22");
}
}
```

Створення екземпляру таймера, який буде спрацьовувати кожних 15хв.

```
Timer t = new Timer(900000);
```

Створення екземпляру таймера, який буде спрацьовувати кожну годину.

```
Timer t1 = new Timer(3600000);
```

Виконання методу, який відповідає за встановлення з’єднання та запис результату.

```
t.Elapsed += new ElapsedEventHandler(WriteLog);
```

Увімкнення таймера.

```
t.Enabled = true;
try
{
```

Якщо служба вдало запустилася, то виконання методу WriteTextLog()

```
WriteTextLog("Ok");
}
catch (Exception e){
```

Якщо виникли помилки при запуску служби, то запис їх в журнал.

```
WriteTextLog(e.Message);
}
```

Виконання методу, який відповідає за перевірку дати створення файлу та видалення його, якщо файл є застарілим.

```
t1.Elapsed += new ElapsedEventHandler(DeleteCheck);
```

Увімкнення таймера.

```
t1.Enabled = true;
}
```

Метод для виконання запитів та збереження отриманої інформації.

```
private static void WriteLog(object source, ElapsedEventArgs e)
{
```

Шлях до файлу в якому зберігаються зовнішні вузли.

```
string path = @"C:\DYPLOM\ExternalIP.txt";
```

Зчитування всіх рядків з файла в масив s, який має тип string.

```
string[] s = System.IO.File.ReadAllLines(path);
```

Шлях до файлу в якому зберігаються внутрішні вузли.

```
string path1 = @"C:\DYPLOM\InternalIP.txt";
```

Зчитування всіх рядків з файла в масив s1, який має тип string.

```
string[] s1 = System.IO.File.ReadAllLines(path1);
```

Визначення спільної довжини масивів s та s1.

```
int x = s.Length + s1.Length;
```

Створення нового масиву для збереження хостів довжиною x.

```
string[] hostname = new string[x] ;
```

Створення нового масиву для збереження портів довжиною x.

```
int[] port = new int[x];
```

Заповнення масивів зовнішніми вузлами та портами.

```
for (int i = 0; i < s.Length; i++)
{
```

Розділяємо кожен рядок масиву s по пробілу та записуємо отримані дані в новий масив типу string.

```
string[] words = s[i].Split(' ');
```

Записуємо в масив для вузлів перший елемент масиву words.

```
hostname[i] = words[0];
```

Запис в масив для портів попередньо сконвертованого значення до int порта.

```
port[i] = Convert.ToInt32(words[1]);
}
```

Заповнення масивів внутрішніми вузлами та портами.

```
for (int i = 0; i < s1.Length; i++)
{
```

Розділяємо кожен рядок масиву s1 по пробілу та записуємо отримані дані в новий масив типу string.

```
string[] words1 = s1[i].Split(' ');
```

Записуємо в масив для вузлів перший елемент масиву words1.

```
hostname[i+ s.Length] = words1[0];
```

Запис в масив для портів попередньо сконвертованого значення до int порта.

```
port[i + s.Length] = Convert.ToInt32(words1[1]);
}
```

Шлях для створення файлу в якому буде зберігатися інформація по отриманих запитах.

```
string sPath = @"C:\DYPLOM\WS2\FILES1\" + DateTime.Now.ToString("dd.MM.yyyy") + ".log";
for (int i = 0; i >= 0; i++)
{
```

Отримання IP-адрес з підсистеми DNS зв’язаних з іменем вузла.

```
IPAddress ipa = (IPAddress)Dns.GetHostAddresses(hostname[i])[0];
try
{
```

Створення змінної для визначення статусу підключення.

```
string status;
```

Створення об’єкта сокета, який використовує протокол TCP.

```
Socket sock = new Socket(System.Net.Sockets.AddressFamily.InterNetwork,
System.Net.Sockets.SocketType.Stream, System.Net.Sockets.ProtocolType.Tcp);
```

З’єднання з відповідним портом та вузлом.

```
sock.Connect(ipa, port[i]);
```

Перевірка на встановлення з’єднання.

```
if (sock.Connected == true)
{
```

Встановлення значення змінної в «Доступний».

```
status = "Доступний";
try
{
```

Відкриття файлового потоку та запис до створеного файлу отриманих даних.

```
using (StreamWriter F = new StreamWriter(sPath, true))
{
F.WriteLine(hostname[i] + " " + port[i] + " " + DateTime.Now.ToString("dd.MM.yyyy") + "
" + DateTime.Now.ToString("HH:mm:ss")+ " " +status);
}
```

Якщо процес внесення даних вдалий, то запис в файл значення “INSERTED!”

```
WriteTextLog("INSERTED!");
}
catch (Exception E)
{
```

Якщо дані не були записані, то запис в файл інформації про помилку.

```
WriteTextLog(E.Message);
```

```
}
}
```

Закриття з’єднання з відповідним портом та вузлом.

```
sock.Close();
}
```

Якщо з’єднання було невдалим, то виконуються наступні дії.

```
catch (System.Net.Sockets.SocketException ex)
{
string status;
```

Встановлення значення змінної в «Недоступний».

```
status = ("Недоступний");
try
{
```

Відкриття файлового потоку та запис до створеного файлу отриманих даних.

```
using (StreamWriter F = new StreamWriter(sPath, true)){
F.WriteLine(hostname[i] + " " + port[i] + " " + DateTime.Now.ToString("dd.MM.yyyy") + "
" + DateTime.Now.ToString("HH:mm:ss") + " " + status);
}
WriteTextLog("INSERTED!");
}
catch (Exception E)
{
WriteTextLog(E.Message);
}

}
}
}
```

Метод для запису у файл інформації при виникненні помилки та успішному записі інформації.

```
private static void WriteTextLog(string z)
{

using (StreamWriter F = new StreamWriter("C:\\DYPLOM\\WS2\\F10.log", true))
{
```

Запис у файл поточної дати та часу та отриманого повідомлення.

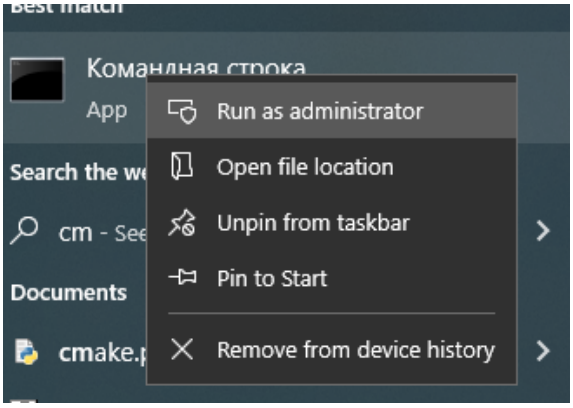
```
F.WriteLine(DateTime.Now + "" + z);
}
}
```

Метод для перевірки дати створення файлу та видалення його, якщо файл застарів(термін існування файлу більший ніж 365 днів).

```
private static void DeleteCheck(object source, ElapsedEventArgs e)
{
(from f in new DirectoryInfo("C:\\DYPLOM\\WS2\\FILES1").GetFiles()
where f.CreationTime < DateTime.Now.Subtract(TimeSpan.FromDays(365)) select
f).ToList().ForEach(f => f.Delete());
}
}
}
```

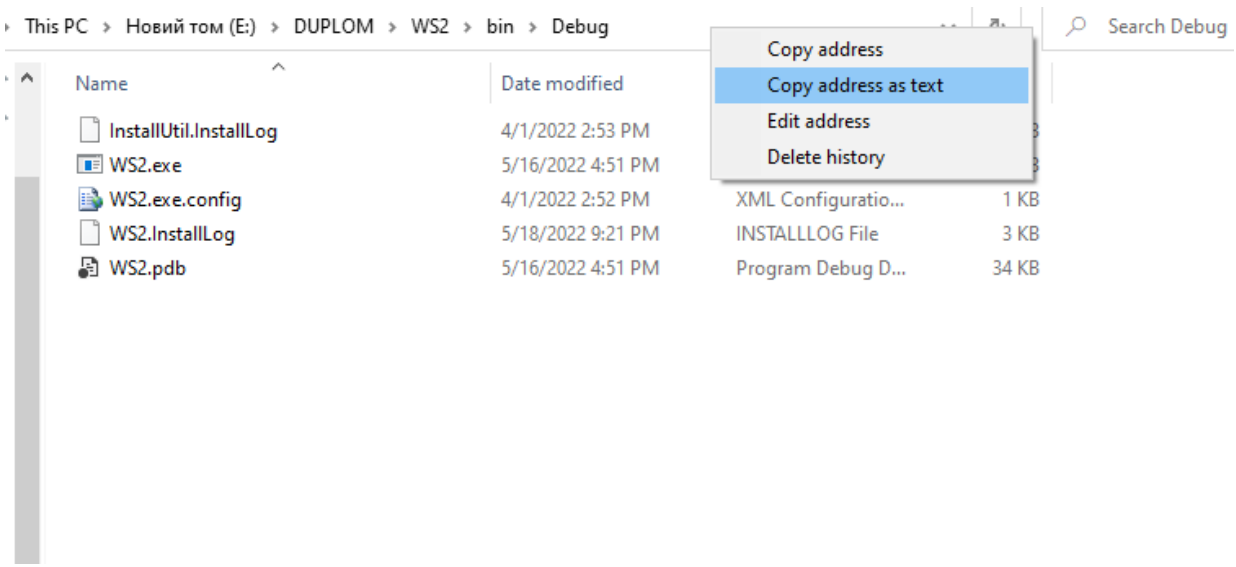
Після зміни логіки роботи служби та її успішному запуску в катологі C:\DYPLOM створюються текстові файли ExternalIP.txt (для збереження зовнішніх вузлів та портів на які будуть вілправлятися запити) та InternalIP.txt(для збереження внутрішніх вузлів та портів).

6. Реєстрація служби у вбудованому середовищі System Control Manager. Відкриття командного рядка в режимі Administrator.



« Рисунок 7 - Відкриття командного рядка в режимі Administrator»

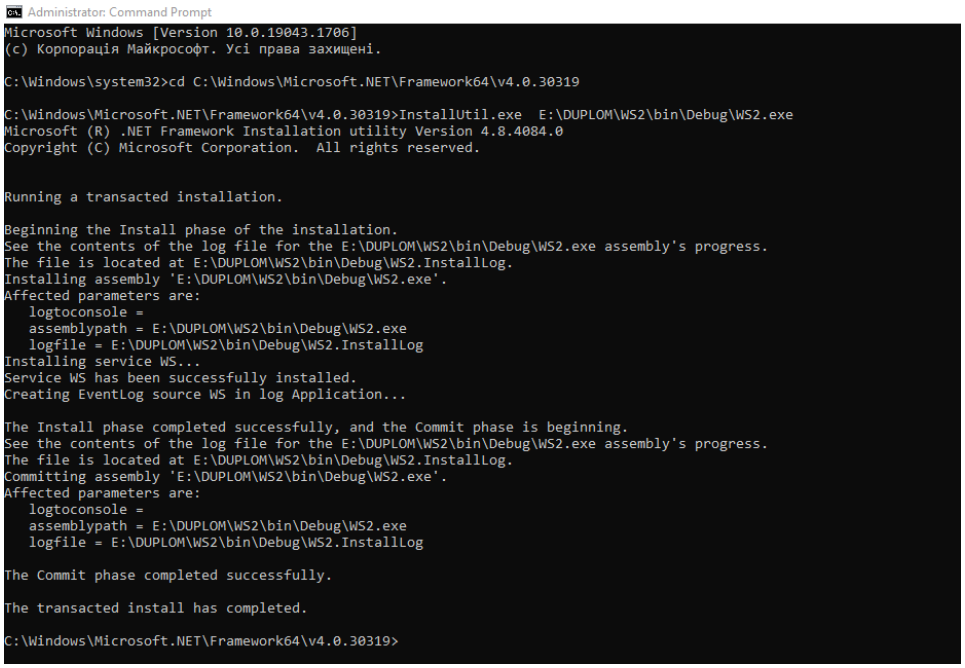
- З командного рядка потрібно перейти в каталог під назвою v4.0.30319, використавши команду:
  cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319
- Відкрити папку з проектом, перейти в каталог ./bin/Debug та скопіювати шлях до файлу.



«Рисунок 8 - Копіювання шляху до файлу з розширенням .exe»

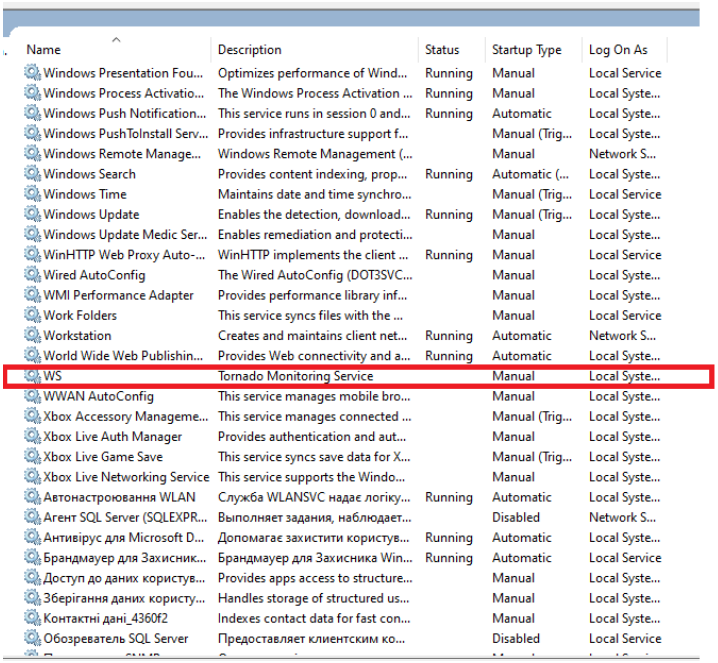
- Введення команди для інсталяції служби в командному рядку:

InstallUtil.exeE:\DUPLOM\WS2\bin\Debug\WS2.exe



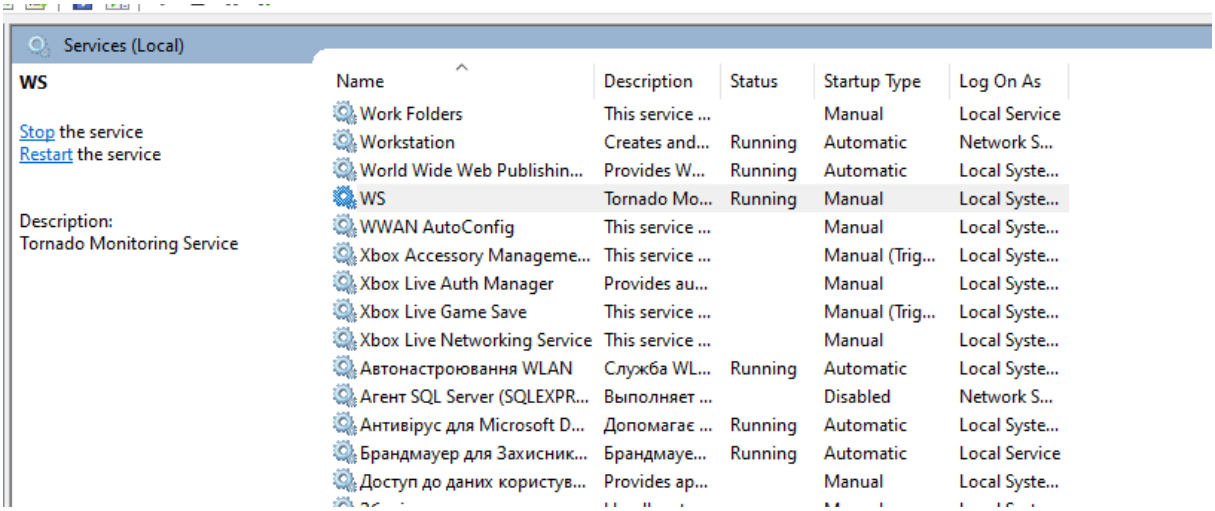
«Рисунок 9 - Інсталяція створеної служби»

- Перевірка успішного встановлення Windows Service у вбудованому середовищі System Control Manager.



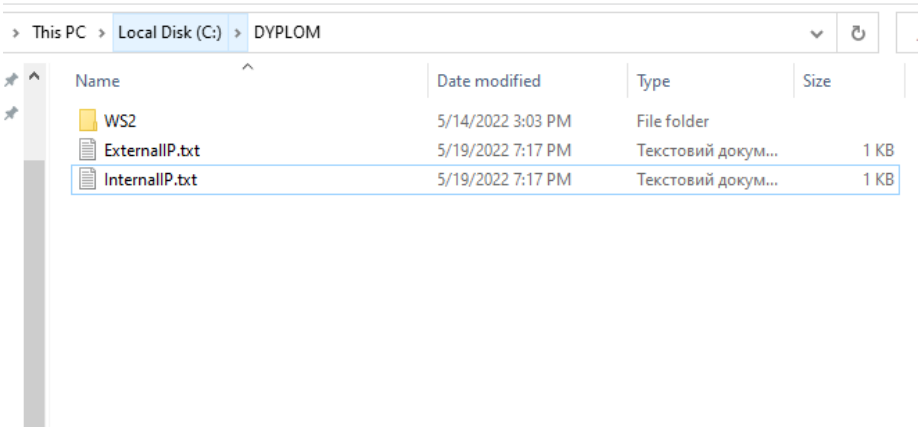
«Рисунок 10 - Проінстальована служба в System Control Manager»

7. Запуск служби та перевірка правильності роботи.



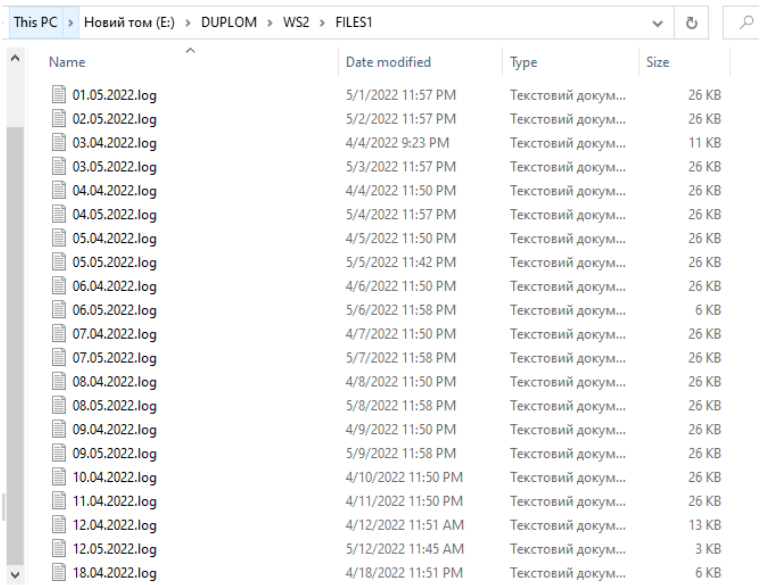
«Рисунок 11 -Запущена служба в System Control Manager»

При першому запуску перевіряється існування файлів ExternalIP.txt (даний файл потрібен для зовнішніх IP-адрес та портів) та InternalIP.txt (текстовий документ для внутрішніх адрес та портів), якщо даних файлів не знайдено, то вони створюються з даними, які вказані за замовчуванням.



«Рисунок 12 - Каталог C:\DYPLOM зі створеними файлами»

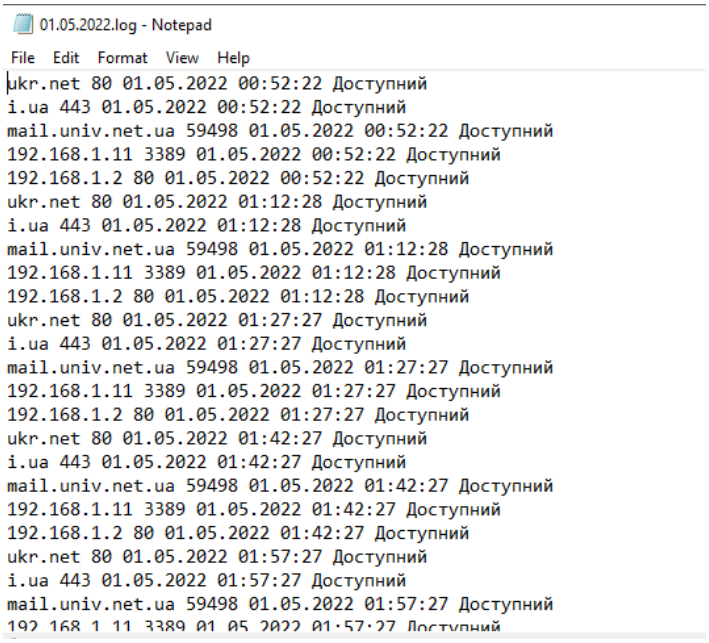
Windows Service працює таким чином, що кожних 15 хвилин будуть відправлені запити на вказані хости та порти і отриманий результат після з'єднання буде записаний у файл з поточною датою. Кожного дня у каталозі за шляхом C:\DYPLOM\WS2\FILES1 створюється новий файл для накопичення інформації.



«Рисунок 13 - Фрагмент каталогу із файлами для збереження результатів»

Після відправлення запиту на відповідний вузол результат записується у такому вигляді: назва хоста, назва порта, дата та час отриманої відповіді та статус

з’єднання («Доступний», якщо з’єднання було вдалим або «Недоступний», якщо з’єднання не відбулося).



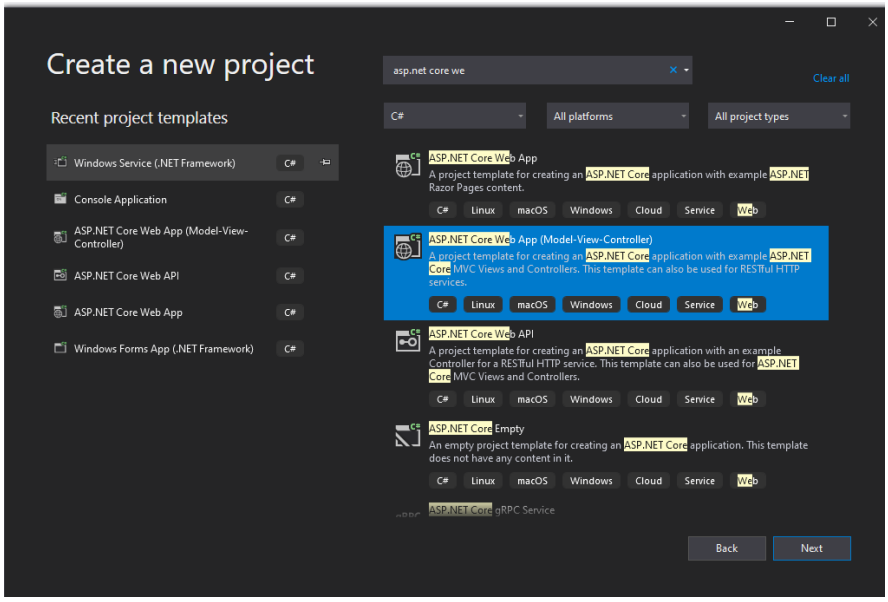
«Рисунок 14 - Фрагмент файлу з отриманою інформацією в результаті запитів»

3.2 Створення веб-сайту для графічного відображення отриманих статистичних даних

Для створення програми з веб-інтерфейсом для відображення статистичних даних було використано ASP.Net Core.

Для створення веб-сайту у Microsoft Visual Studio потрібно виконати такі кроки:

1. Створити проект ASP.Net Core MVC.



«Рисунок 15 - Створення ASP.NET Core App»

2. Створення класу для полів для роботи з накопиченою інформацією та управління зовнішніми та внутрішніми вузлами зі списку вибору для отримання запитуваних даних .

```
public class FileViewModel
{
```

Поле для роботи з назвою хоста.

```
public string HostName { get; set; }
```

Атрибут для дати, коли було отримано запис в текстовий файл.

```
public string Date { get; set; }
```

Властивість для визначення статусу з’єднання.

```
public string Status { get; set; }
```

Поле даних для відображення відсотка доступності вузла.

```
public float Score { get; set; }
```

Атрибут для вказання початкової дати в проміжку.

```
public string start1 { get; set; }
```

Властивість для вказання кінцевої дати з проміжку.

```
public string end1 { get; set; }
```

Поле для списку вибору, що відображає Id вказаного значення.

```
public string Id { get; set; }
```

Властивість, яка відображає назву вузла зі списку вибору.

```
public string Name{ get; set; }
```

Метод, який повертає масив вузлів, вказаних в файлах, які відповідають за внутрішні та зовнішні адреси.

```
public static string [] hosts()
{
```

Шлях до файлу в якому зберігаються зовнішні вузли та порти для запитів.

```
string path = @"C:\DYPLOM\ExternalIP.txt";
```

Збереження всіх рядків із файлу ExternalIP.txt в масив з типом даних string.

```
string[] s = System.IO.File.ReadAllLines(path);
```

Адреса за якою знаходиться файл для внутрішніх адрес та портів.

```
string path1 = @"C:\DYPLOM\InternalIP.txt";
```

Збереження рядків із файлу для внутрішніх вузлів та портів у масив даних string.

```
string[] s1 = System.IO.File.ReadAllLines(path1);
```

Визначення спільної довжини масивів зі збереженими рядками з файлів.

```
int x = s.Length + s1.Length;
```

Створення нового масиву типу string з довжиною x для збереження назв вузлів.

```
string[] hostname = new string[x];
```

Новий масив з типом int для збереження портів отриманих із файлів.

```
int[] port = new int[x];
```

Заповнення масивів для портів та вузлів із збереженого масиву з рядків для зовнішніх адрес.

```
for (int i = 0; i < s.Length; i++)
{
```

Розрізання рядків в масив з типом string.

```
string[] words = s[i].Split(' ');
```

Присвоєння поточному елементу масиву для вузлів першого елемента масиву з розрізаним рядком.

```
hostname[i] = words[0];
```

Заповнення даними, отриманими з файлу, масиву для портів.

```
port[i] = Convert.ToInt32(words[1]);
}
```

Заповнення масивів для портів та вузлів із збереженого масиву з рядків для внутрішніх адрес.

```
for (int i = 0; i < s1.Length; i++)
{
```

Заповнення масиву розрізаним рядком із масиву для рядків із внутрішніх адрес.

```
string[] words1 = s1[i].Split(' ');
```

Присвоєння поточному елементу масиву для вузлів першого елемента масиву з розрізаним рядком.

```
hostname[i + s.Length] = words1[0];
```

Заповнення даними, отриманими з файлу, масиву для портів.

```
port[i + s.Length] = Convert.ToInt32(words1[1]);
}
```

Повернення заповненого масиву.

```
return hostname;
}
```

Метод для заповнення списку з отриманих даних із файлів для відображення у списку вибору на сайті.

```
public static IEnumerable<FileViewModel> GetProduct()
{
```

Виклик методу для заповнення масиву даними.

```
string[] hostname = hosts();
```

Створення нового списку для збереження отриманої інформації.

```
List<FileViewModel> lst = new List<FileViewModel>();
```

Додавання записів в список, які включають в себе номер запису та назву вузла.

```
for (int i = 1; i < hostname.Length+1; i++)
{
lst.Add(new FileViewModel { Id = i.ToString(), Name = hostname[i-1] });
}
```

Додавання до списку назви «Усі зовнішні», для роботи з всіма зовнішніми адресами.

```
lst.Add(new FileViewModel { Id =(lst.Count+1).ToString() , Name = "Усі зовнішні" });
```

Додавання до списку назви «Усі внутрішні», для роботи з всіма внутрішніми адресами.

```
lst.Add(new FileViewModel { Id = (lst.Count+1).ToString(), Name = "Усі внутрішні" });
```

Додавання до списку назви «Усі», для відображення усіх вузлів по яких проводилися запити.

```
lst.Add(new FileViewModel { Id = (lst.Count+1).ToString(), Name = "Усі" });
```

Повернення заповненого списку.

```
return lst;
}
}
}
```

3. Розроблення логіки роботи контролера сайту для виведення накопичених даних та заповнення списку вибору.

```
namespace Site1.Controllers
{
public class HomeController : Controller
{
```

Змінні для додавання журналів в програму.

```
private readonly ILogger<HomeController> _logger;
private readonly IConfiguration _config;
```

Контейнер залежностей, який вбудований в ASP.Net Core

```
public HomeController(ILogger<HomeController> logger, IConfiguration config)
{
_logger = logger;
_config = config;
}
```

Головний метод для роботи з сайтом, приймає параметри для проміжку часу та вибору назви вузла для якого потрібно зробить запит.

```
public IActionResult Index(DateTime start, DateTime end, string select1,string
date)
{
```

Виклик методу для отримання рядків в таблицю для відображення на сторінці.

```
var Items = getAll(start, end, select1);
```

Перевірка встановленого значення в елементі input, який має тип date.

```
if (start.ToString("yyyy-MM-dd")=="0001-01-01")
```

```
{
```

Якщо значення задовольняє умову, то встановлюємо в input початкову та кінцеву дати в значення поточного дня.

```
FileViewModel rec = new FileViewModel
{
start1=DateTime.Now.ToString("yyyy-MM-dd"),
end1= DateTime.Now.ToString("yyyy-MM-dd")

};
```

Передача даних для відображення на сторінці сайту.

```
ViewData["Message"] = rec;
}
else
{
```

Збереження встановленого проміжку часу після перезавантаження сторінки.

```
FileViewModel rec = new FileViewModel
{

start1 = start.ToString("yyyy-MM-dd"),
end1 = end.ToString("yyyy-MM-dd")
};
```

Використовується для передачі даних для відображення на сторінці.

```
ViewData["Message"] = rec;
}
```

«Рисунок 16 - Відображення встановлення дати в елементах input»

Виклик методу для відображення назв вузлів для вибору в елементі вибору на сайті.

```
IEnumerable<FileViewModel> lst = FileViewModel.GetProduct();
```

Передача отриманих даних для відображення на сторінці.

```
ViewBag.ProductList = lst;

return View(Items);
}
```

Метод для запису в текстовий файл результатів опрацювання даних та повідомлень про помилку, якщо така виникає.

```
private static void WriteTextLog(string z)
{
using (StreamWriter F = new StreamWriter("C:\\DYPLOM\\F34.log", true))
{
F.WriteLine(DateTime.Now + "" + z);
}
}

public IActionResult Privacy()
{
return View();
}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
true)]
public IActionResult Error()
{
return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
}
```

Метод для опрацювання даних та додавання результатів у список для відображення у вигляді рядків таблиці на сторінці сайту. Даний метод приймає три параметри: початкову та кінцеву дату для проміжку за який має бути виведений результат та назву вузла для отримання опрацьованих даних.

```
private List<FileViewModel> getAll(DateTime? start1, DateTime? end1, string
select1)
{
```

Конструкція, яка отримує всі файли у вказаній директорії.

```
DirectoryInfo DirInfo = new DirectoryInfo(@"C:\DYPLOM\WS2\FILES1");
```

Змінна для вказання початкової дати для проміжку.

```
var StartDate = Convert.ToDateTime(start1);
```

Змінна для кінцевої дати проміжку.

```
var EndDate = Convert.ToDateTime(end1);
```

Створеннння списку для проміжного запису результатів отриманих записів.

```
List<FileViewModel> lstFiles = new List<FileViewModel>();
```

Список для формування остаточного вигляду рядків для відображення в таблиці.

```
List<FileViewModel> lstFiles8 = new List<FileViewModel>();
```

Список для збереження відсотків доступності за вказаний проміжок часу для вибраного хоста.

```
List<Int32> lstFiles1 = new List<Int32>();
```

Виклик методу для отримання списку даних з елемента вибору.

```
IEnumerable<FileViewModel> lst = FileViewModel.GetProduct();
```

Отримання назви вузла за вказаним номерним знаком.

```
var k1 = from m in lst where m.Id == select1 select m.Name;
string select2;
```

Створення масиву для запису всіх хостів, які потрібні для виводу інформації.

```
string[] hostinside = new string[] { };
```

Цикл, який проходиться по всіх отриманих іменах зі списку.

```
foreach (string d in k1)
{
select2 = d;
```

Якщо отриманий рядок з елементу вибору є «Усі внутрішні», то заповнюємо масив даними з файлу для збереження внутрішніх адрес та портів.

```
if ( select2== "Усі внутрішні")
{
string path1 = @"C:\DYPLOM\InternalIP.txt";
string[] s1 = System.IO.File.ReadAllLines(path1);
string[] hostname = new string[s1.Length];
for (int i = 0; i < s1.Length; i++)
{
string[] words = s1[i].Split(' ');
hostname[i] = words[0];
}
hostinside= hostname;
}
```

Якщо результат дорівнює «Усі зовнішні», то заповнюємо масив даними з файлу для зовнішніх вузлів та портів.

```
else if (select2 == "Усі зовнішні")
{
string path1 = @"C:\DYPLOM\ExternalIP.txt";
```

```
string[] s1 = System.IO.File.ReadAllLines(path1);
string[] hostname = new string[s1.Length];
for (int i = 0; i < s1.Length; i++)
{
string[] words = s1[i].Split(' ');
hostname[i] = words[0];
}
hostinside = hostname;
}
```

Якщо отриманий результат з елемента вибору є «Усі», то викликаємо метод hosts() з класу FileViewModel та отримуємо всі вузли з файлів для зовнішніх та внутрішніх адрес.

```
else if (select2== "Yci")
{
hostinside = FileViewModel.hosts();

}
```

Якщо отримана назва не задовольняє попередні умови, то додаємо отримане значення з елемента selectв масив для відображення результатів для вказаного вузла.

```
else
{
hostinside = new string[] { select2 };
}
}
```

Циклічне проходження по масиву з отриманими назвами вузлів.

```
foreach (string v in hostinside)
{
```

Якщо початкова та кінцева дати збігаються, то отримуємо список файлів в змінну з вказаної директорії.

```
if (StartDate == EndDate)
{
try
{
var files = Directory.GetFiles(@"C:\DYPLOM\WS2\FILES1")
.Where(x => new FileInfo(x).CreationTime.Date == StartDate);
```

Проходження по значеннях змінної files.

```
foreach (var k in files)
```

```
{
```

Запис в змінну шляху до файлу.

```
string path = k.ToString();
```

Запис всіх рядків вказаного файлу в масив даних типу string.

```
string[] s = System.IO.File.ReadAllLines(path);
```

Змінні для визначення відсотка доступності певного вузла.

```
float l = 0;
float b = 0;
```

Змінні для встановлення отриманої назви хоста, дати та статусу відповідно.

```
string a = "";
string c = "";
string d = "";
```

Цикл для проходження по всіх рядках отриманих з файлу, їх розділення та запис в новий масив.

```
foreach (var i in s)
{
string[] words = i.Split(' ');
```

Перевірка на знаходження вузла в масиві даних та виконання дій для вираховування відсотка доступності.

```
if (words[0] == v)
{
l++;
a = words[0];
c = words[2];
if (words[4] == "Доступний")
{
b++;
d = words[4];
}
}
}
int score = Convert.ToInt32((b / l) * 100);
```

Додавання даних отриманих в результаті перевірок в список для виконання подальших дій для них.

```
if (score > 0)
{
```

```
        lstFiles.Add(new FileViewModel { HostName = a, Date = c, Status = d, Score = score
});
        }
        else
        {
        lstFiles.Add(new FileViewModel { HostName = a, Date = c, Status = "Недоступний",
Score = score });
        }
        }
```

Якщо не виникало помилок під час проходження по алгоритму, то виклик відповідного методу та запис рядка про успіх виконання.

```
        WriteTextLog("good1");
        }
        catch (Exception e)
        {
        WriteTextLog(e.Message);
        }
        }
        else
        {
        try
        {
```

Якщо вказаний період часу, то тримання в змінну всіх файлів з вказаного проміжку.

```
        var files = from f in DirInfo.EnumerateFiles()
        where f.CreationTime.Date >= StartDate && f.CreationTime.Date <= EndDate
        select f;
```

Змінні для отримання результатів обробки даних, а саме назви вузла, дати, статусу та відсотка доступності.

```
        string a = "";
        string c = "";
        string d = "";
        int score = 0;
```

Проходження по отриманих файлах за вказаний період.

```
        foreach (var k in files)
        {
```

Обробка даних та вираховування відсотка доступності вказаного вузла.

```
        string path = k.ToString();
        string[] s = System.IO.File.ReadAllLines(path);
```

```
float l = 0;
float b = 0;
foreach (var i in s)
{
string[] words = i.Split(' ');
if (words[0] == v)
{
l++;
a = words[0];
c = words[2];

if (words[4] == "Доступний")
{
b++;
d = words[4];
}
}
}
score = Convert.ToInt32((b / l) * 100);
lstFiles1.Add(score);
```

Додавання отриманих результатів у список для подальшої обробки даних.

```
if (score > 0 && score <= 100)
{
lstFiles.Add(new FileViewModel { HostName = a, Date = c, Status = d, Score =
score});
}
else if (score == 0)
{
lstFiles.Add(new FileViewModel { HostName = a, Date = c, Status = "Недоступний",
Score = score});
}

}

WriteTextLog("Good2");


}
catch (Exception e)
{
WriteTextLog(e.Message);
}
```

```
}
}
```

Алгоритм для об’єднання рядків списку для виведення результуючих даних для кожного вузла за вказаний період часу з загальним відсотком доступності.

Список для збереження назв вузлів отриманих з проміжних результатів обробки даних.

```
List<string> lstFiles3 = new List<string>();
```

Проходження по списку отриманому раніше.

```
foreach (var g in lstFiles)
{
```

Запис в список назв хостів без повторів.

```
bool vic = lstFiles3.Contains(g.HostName);
if(vic==false)
{
lstFiles3.Add(g.HostName) ;
}
}
```

Алгоритм для знаходження загального відсотка до вузла за вказаний період.

```
foreach (string h in lstFiles3)
{
float p = 0;
float z = 0;
foreach (var e in lstFiles)
{
if (e.HostName == h)
{
p += e.Score;
z++;
}
}
p = p / z;
```

Запис у список отриманих даних з урахування певних вимог для подальшого виведення у вигляді рядків таблиці на сторінці розробленого сайту.

```
if (p == 0)
{
```

Якщо відсоток дорівнює нулю, то встановлення статусу в значення «Недоступний».

```
if (start1 == end1)
{
```

Якщо початкова та кінцева дати співпадають, то в колонці з назвою «Дата» встановлення одного значення дати.

```
lstFiles8.Add(new FileViewModel { HostName = h, Date = start1.Value.ToString("dd.MM.yyyy"), Status = "Недоступний", Score = Convert.ToInt32(p) });
}
else
{
```

Якщо вказаний проміжок, то в колонці «Дата» вказується період часу за який отримано результат.

```
lstFiles8.Add(new FileViewModel { HostName = h, Date = start1.Value.ToString("dd.MM.yyyy") + "-" + end1.Value.ToString("dd.MM.yyyy"), Status = "Недоступний", Score = Convert.ToInt32(p) });
}
}
else
{
```

Перевірка на співпадіння дат, якщо це відбулося, то вказання в колонці «Дата» одного значення вибраної дати.

```
if (start1 == end1)
{
lstFiles8.Add(new FileViewModel { HostName = h, Date = start1.Value.ToString("dd.MM.yyyy"), Status = "Доступний", Score = Convert.ToInt32(p) });
}
```

Якщо дати є різними, то вказання в колонці «Дата» проміжку часу за який отриманий кінцевий результат.

```
else
{
lstFiles8.Add(new FileViewModel { HostName = h, Date = start1.Value.ToString("dd.MM.yyyy") + "-" + end1.Value.ToString("dd.MM.yyyy"), Status = "Доступний", Score = Convert.ToInt32(p) });
}
}
```

```
        }
        return lstFiles8;
        }
        }
        }
```

4. Розробка коду для відображення графічного інтерфейсу для сайту та виведення накопичених даних у вигляді рядків таблиці.

```
@model IEnumerable<Site1.Models.FileViewModel>
@{
ViewData["Title"] = "Index";
}
@{
```

Змінна для збереження встановлених дат.

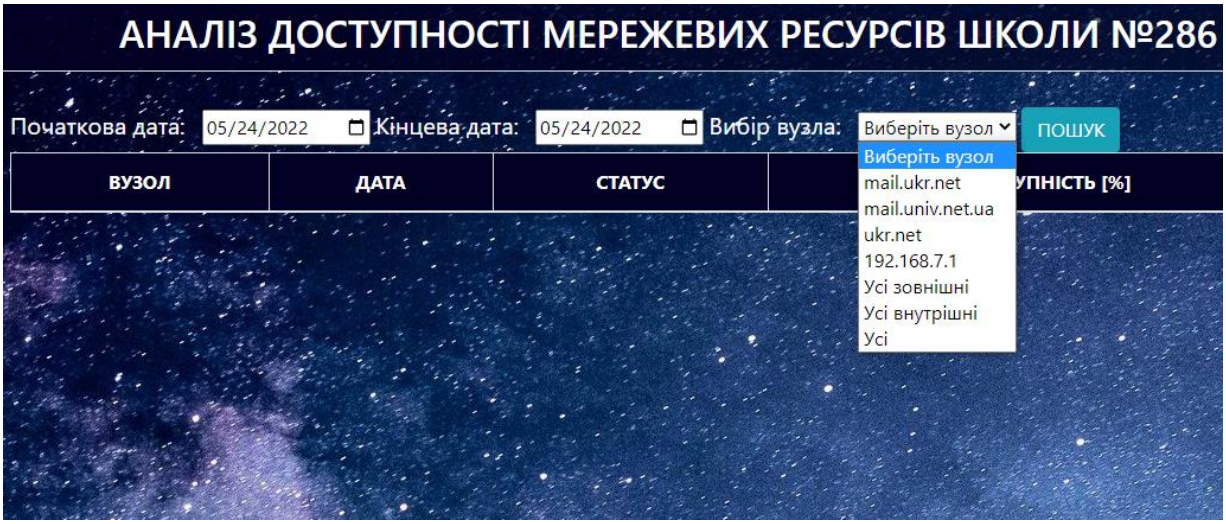
```
var data = (FileViewModel)ViewData["Message"];
}
```

Використання метода Get для отримання введених даних та обробки їх в контролері.

```
@using (Html.BeginForm("Index", "Home", FormMethod.Get))
{
```

Створення форми для введення даних, щоб в подальшому отримувати результат запитів. Дана форма включає два input типу dateдля визначення проміжку часу, список вибору вузлів та кнопку для обробки запиту.

```
<form >
<span style="color: white; font-size: 20px;margin-right:10px;">Початкова дата:</span>
<input type="date"value="@data.start1" name="start" style=margin-right: 10px; font-size:
15px;" />
<span style="color: white; font-size: 20px; margin-right: 10px;">Кінцева дата:</span>
<input type="date"value="@data.end1" style= margin-right: 10px; font-size: 15px;" name="end"
/>
<span style="color: white; font-size: 20px; margin-right: 10px;">Вибір вузла:</span>
@Html.DropDownList("select1",new SelectList(ViewBag.ProductList,"Id","Name",0), "Виберіть
вузол")
<input type="submit" id="button" name="submit"class="btn btn-info" value="ПОШУК" />
</form>
}
```



«Рисунок 17 - Розроблена форма для обробки накопичених даних»

Формування таблиці для отриманих результатів.

```
<div style="text-align:center" class="table-responsive table">
<table class="table table-bordered ">
```

Створення рядка із заголовками для таблиці та вказання їм стилю.

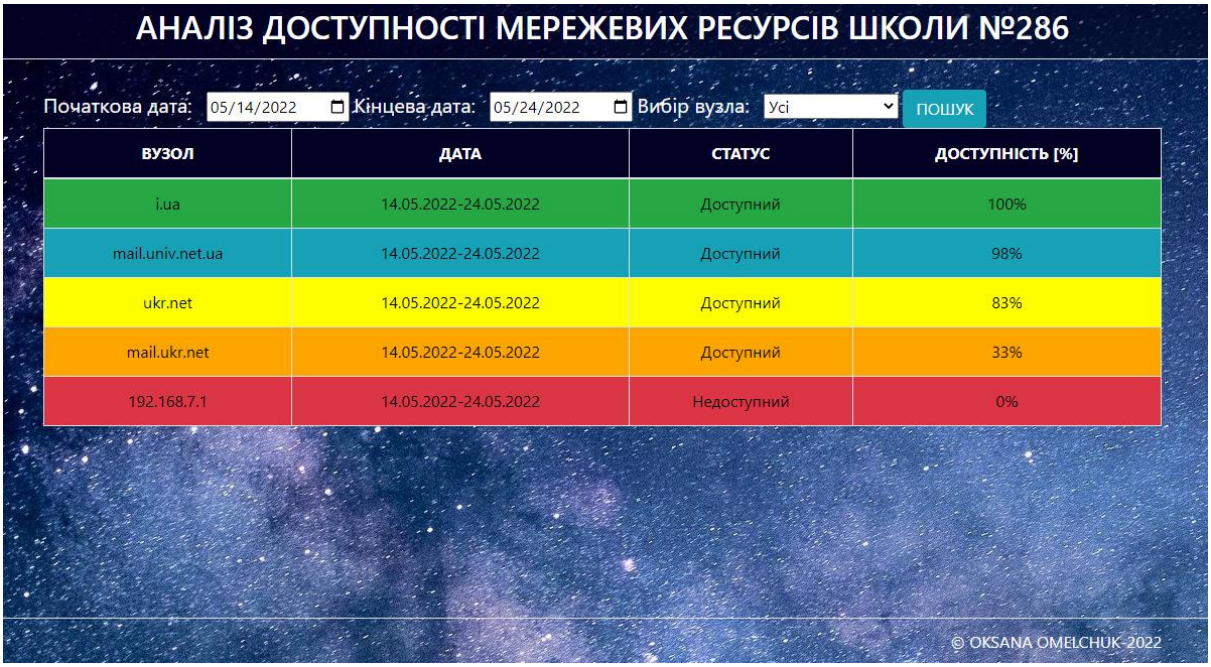
```
<thead>
<tr style="background: linear-gradient(90deg, rgba(2,0,36,1) 100%,
rgba(3,6,29,0.4990371148459384) 100%, rgba(0,212,255,1) 100%);color:white; ">
<th>ВУЗОЛ</th>
<th>ДАТА</th>
<th>СТАТУС</th>
<th>ДОСТУПНІСТЬ [%]</th>
</tr>
</thead>
```

Заповнення таблиці даними. З метою кращого сприйняття отриманих результатів було прийнято рішення щодо виділення рядків різними кольорами в результаті отриманого відсотка доступності вузла за вказаний період часу. Якщо відсоток становить 0, то виділення рядка червоним кольором, якщо значення від 1 до 50, то колір рядка помаранчевий, якщо від 50 до 95, то колір жовтий, від 95 до 100 – блакитний та якщо значення дорівнює 100, то колір зелений.

```
@foreach (FileViewModel file in Model)
{
@if (file.Score == 0)
```

```
{
<tr class="bg-danger" style="font-size:16px; color:black;">
<td>@file.HostName</td>
<td> @file.Date</td>
<td> @file.Status</td>
<td> @file.Score%</td>
</tr>
}
@if (file.Score >= 1 && file.Score < 50)
{
<trstyle="background-color:orange; font-size:16px; color:black;">
<td>@file.HostName</td>
<td> @file.Date</td>
<td> @file.Status</td>
<td> @file.Score%</td>
</tr>
}
@if (file.Score >= 50 && file.Score < 95)
{
<tr style="background-color:yellow; font-size:16px; color:black;">
<td>@file.HostName</td>
<td> @file.Date</td>
<td> @file.Status</td>
<td> @file.Score%</td>
</tr>
}
@if (file.Score >= 95 && file.Score < 100)
{
<tr class="bg-info" style="font-size:16px; color:black;">
<td>@file.HostName</td>
<td> @file.Date</td>
<td> @file.Status</td>
<td> @file.Score%</td>
</tr>
}
@if (file.Score ==100)
{
<tr class="bg-success " style="font-size:16px; color:black;">
<td>@file.HostName</td>
<td> @file.Date</td>
```

```
<td> @file.Status</td>
<td> @file.Score%</td>
</tr>
}
}
</table>
</div>
```



«Рисунок 18 - Розроблений сайт з результатами обробки накопичених даних.»

3.3 Етапи публікації сайту за допомогою Internet Information Service (IIS)

1. Для віддаленої публікації сайту потрібно встановити застосунок TCP-Proxy та додати налаштування для з’єднання в файл LinkConfig.xml.

У даному файлі вказується переадресація TCP-сокета(IP-адреси та порта) для віддаленого хостингу сайту.

«Рисунок 24 - Фрагмент файлу LinkConfig.xml з доданими налаштуваннями»

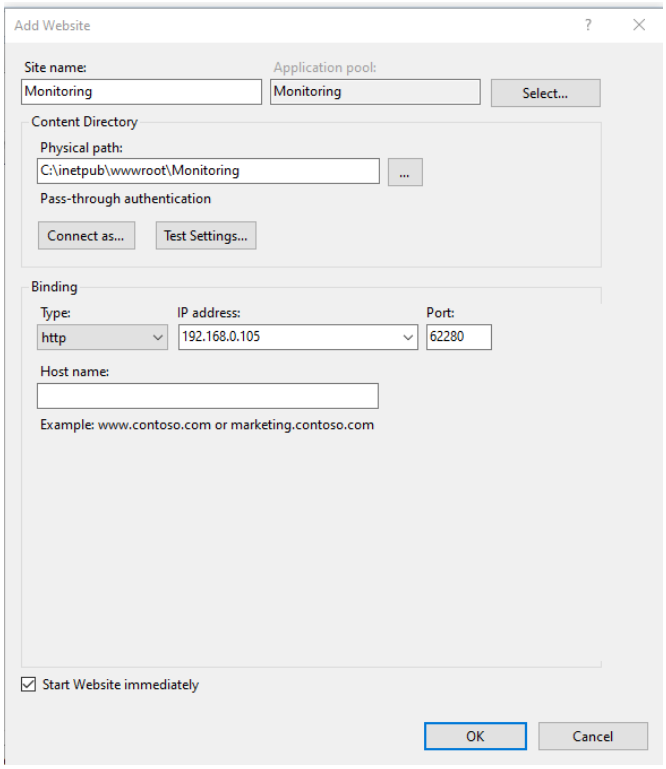
2. Встановлення додаткового пакету для публікації сайту.

Оскільки веб-програми розроблений на платформі ASP.Net Core, то потрібно встановити додатковий пакет для успішного хостингу. Даний пакет включає в себе такі компоненти, як: середовище для ASP.Net Core, бібліотеку та модуль для цієї платформи.

3. Створення сайту та виконання підготовчих дій для його публікації у Internet Information Service.

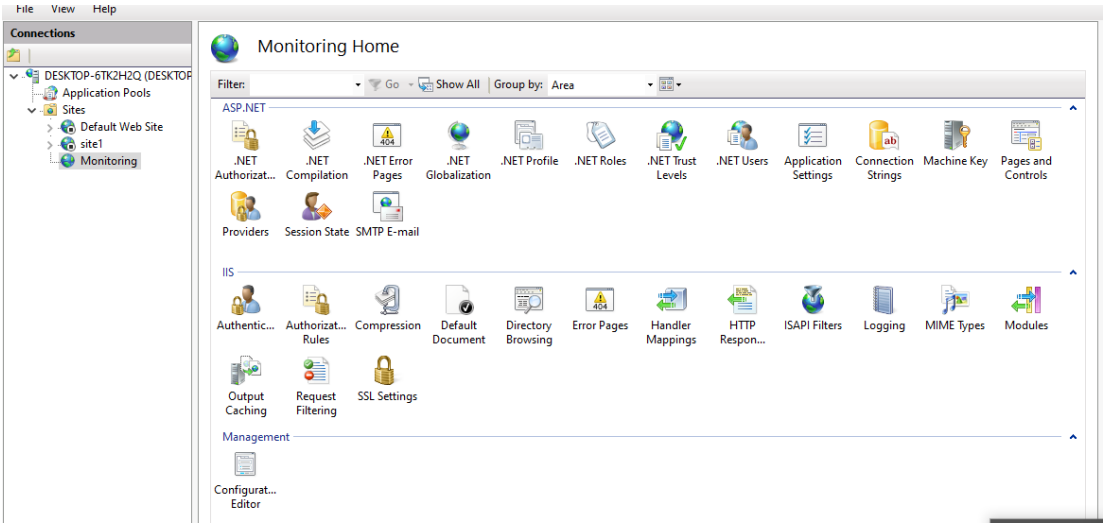
- Створення нового сайту у IIS.

Для створення нової програми з веб-інтерфейсом потрібно вказати назву сайту, фізичний шлях розміщення файлів, IP-адресу та порт за якими буде розміщено сайт.



«Рисунок 19 - Вікно створення нового сайту»

- Перевірка успішного додавання сайту в IIS.

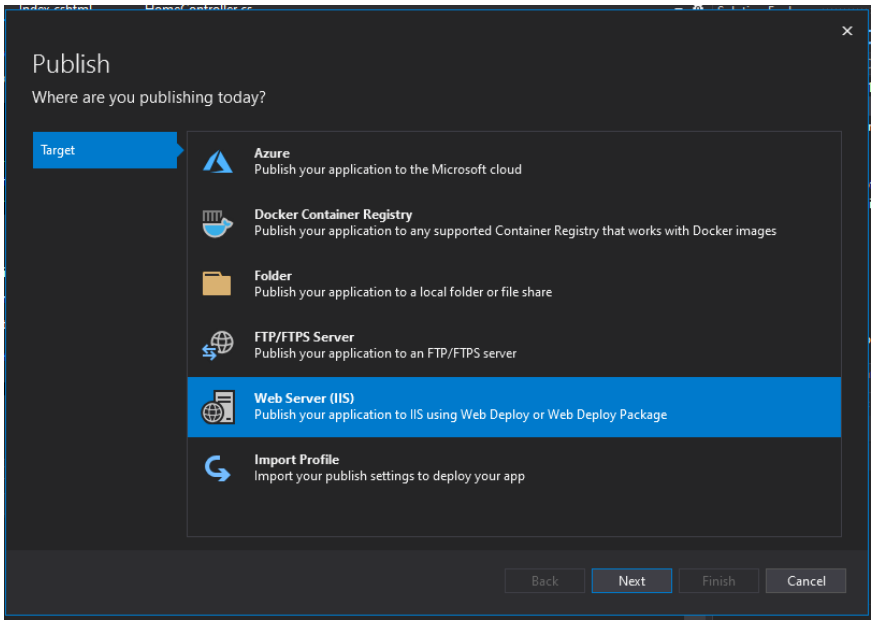


«Рисунок 20 - Вікно IIS з доданим сайтом»

4. Підготовка файлів створеного сайту у Microsoft Visual Studio для подальшої публікації.

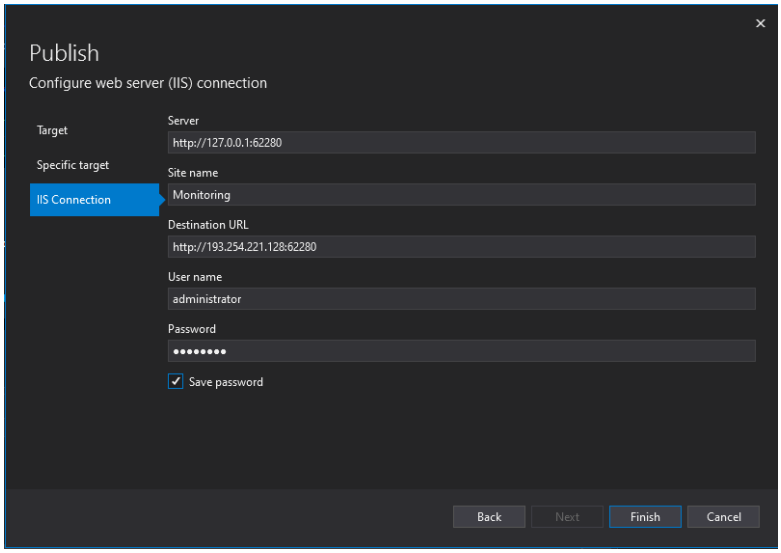
У Visual Studio потрібно відкрити проект та перейти у вкладку Build/Publish.

Далі необхідно натиснути кнопку New та вибрати пункт Web Server(IIS).



«Рисунок 21 - Вікно з вибором Publish»

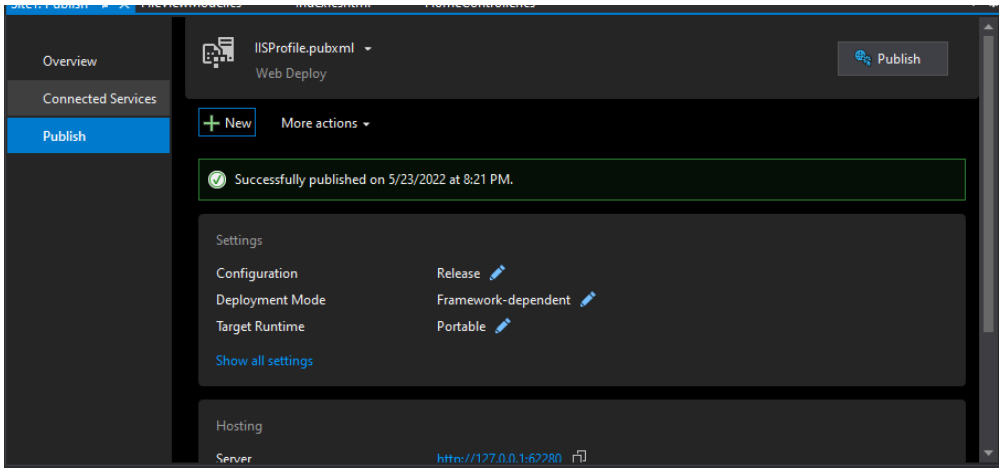
Вибір формату публікації Web Deploy та заповнення інформації для публікації сайту.



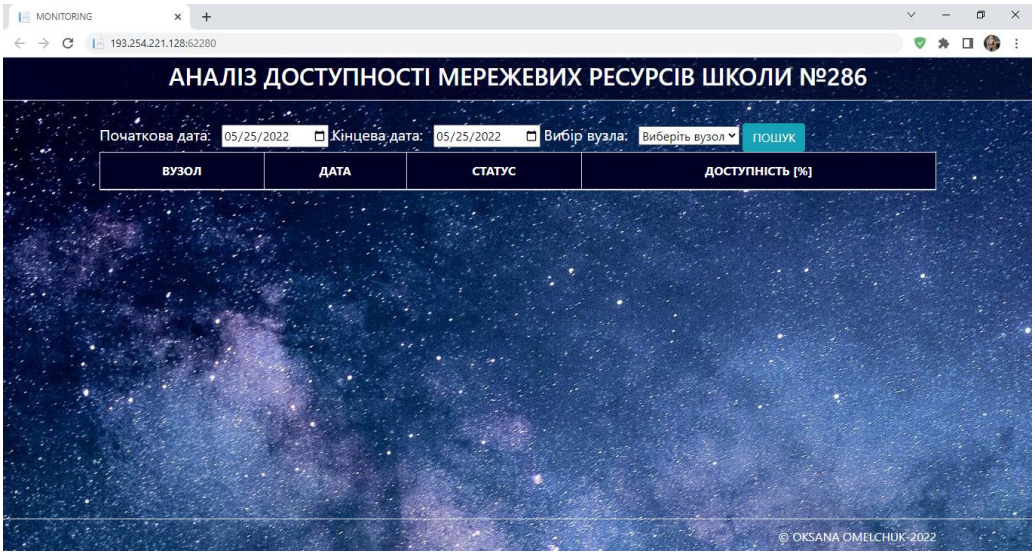
«Рисунок 22 - IIS Connection»

У даному вікні вказується адреса сервера, ім’я веб-програми у IIS, адреса призначення, ім’я користувача та пароль. У результаті проведення тестування публікації було виявлено, що доступ до хостингу в IIS має тільки один користувач administrator, який за замовчуванням є заблокованим.

Після натиснення кнопки Publish розроблений сайт є успішно опублікованим.



«Рисунок 23 - Відображення успішної публікації сайту»



«Рисунок 24 - Успішно опублікований сайт»

ВИСНОВКИ

Аналіз технічного завдання, сформульованого адміністрацією школи I-III ступенів №286 показав, що Інтернет-провайдер надає доступ до мережі Інтернет надійно і стабільно, але періодично працівники школи скаржаться на те, що у конкретний період часу вони «зовсім не змогли приєднатись до ресурсів Інтернету». Це дозволяє зробити висновок, що для кожного користувача локальної мережі школи в заданий момент часу доступ до мережі Інтернет має фактично лише два стійких стани: 1 - повністю працює або 0 - зовсім не працює.

Для того, щоб підтвердити або спростувати скарги працівників школи, а також об'єктивно оцінити рівень доступності ресурсів необхідно і достатньо для кожного вказаного ресурсу обчислити відсоток часу, коли ресурс перебуває у стані 1 по відношенню до усього заданого періоду. В роботі продемонстровано, що найбільш зручно вирішити цю технічну задачу можна за допомогою програмного комплексу, що складається з двох частин: моніторингової системної служби Windows та веб-програми платформи ASP.NET MVC.

Windows-служба працює цілодобово, періодично перевіряє доступність вказаних ресурсів і записує інформацію в текстовий журнал. Зроблено оцінку, що найкращий період визначення доступності складає 15 хвилин. Натомість веб-програма працює тільки в результаті запиту користувача, вона аналізує створений службою журнал, підраховує рівень доступності ресурсу і виводить результат у вигляді веб-сторінки.

Для точного визначення слабкої ненадійної ділянки локальної мережі може бути доцільним навмисно залишити в робочому стані додаткові вузли локальної мережі і визначити рівень доступності цих вузлів за вказаний період часу. Нарешті, для великої розгалуженої локальної мережі є можливість розгорнути моніторингову службу в кількох екземплярах у різних сегментах та підмережах закладу освіти, що в результаті дозволить здобути максимально повну і цілісну інформацію про доступність вузлів і слабкі ділянки локальної мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to Windows Service Applications [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/dotnet/framework/windows-services/introduction-to-windows-service-applications
2. SQL Server Management Studio (SSMS) [Електронний ресурс]. URL: https://docs.microsoft.com/en-gb/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15
3. TCP/IP [Електронний ресурс]. URL: https://www.techtarget.com/searchnetworking/definition/TCP-IP
4. Protocolul TCP/IP [Електронний ресурс]. URL: https://www.competentedigitale.ro/internet/internet_TCP_IP.php
5. How to: Create a Socket [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/dotnet/framework/network-programming/how-to-create-a-socket
6. Introduction to ASP.NET Core [Електронний ресурс]. URL:https://docs.microsoft.com/en-gb/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0
7. Overview of ASP.NET Core MVC [Електронний ресурс]. URL: https://docs.microsoft.com/en-gb/aspnet/core/mvc/overview?view=aspnetcore-6.0
8. IIS Web Server Overview [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/iis/get-started/introduction-to-iis/iis-web-server-overview
9. Transact-SQL Reference (Database Engine) [Електронний ресурс]. URL: https://docs.microsoft.com/en-gb/sql/t-sql/language-reference?view=sql-server-ver15
10. Data types (Transact-SQL) [Електронний ресурс]. URL: https://docs.microsoft.com/en-gb/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15

11. How to: Install and uninstall Windows services [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/dotnet/framework/windows-services/how-to-install-and-uninstall-services

12. Publish an ASP.NET Core app to IIS [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/aspnet/core/tutorials/publish-to-iis?view=aspnetcore-6.0&tabs=visual-studio

13. What is LINQ? [Електронний ресурс]. URL: https://www.tutorialsteacher.com/linq/what-is-linq

14. Language Integrated Query (LINQ) (C#) [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/

15. Tag Helpers in forms in ASP.NET Core [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/working-with-forms?view=aspnetcore-6.0

16. Introduction to Razor Pages in ASP.NET Core [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-6.0&tabs=visual-studio