

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему:

«Проект інформаційної вебсистеми
на базі рішень Microsoft»

Виконав: студент 4-го курсу групи КІ-21-1
спеціальності 123 «Комп'ютерна інженерія»

 Андрій ЛАТАШ

Керівник: Доцент кафедри КІ та КБ, к.т.н.

 Олексій ШЕЛУХА

Рецензент: Доцент кафедри КІ та КБ, к.т.н.

 Олександр ПІРОГ

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій ЄФІМЕНКО

3 березня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Латаш Андрій Андрійович

Тема роботи: «Проект інформаційної вебсистеми на базі рішень Microsoft»

затверджена Наказом університету від 3 березня 2025 р. №79

Термін здачі студентом закінченої роботи 9 червня 2025 р.

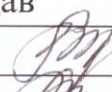
Вихідні дані роботи:

- процеси функціонування вебстеку на основі вебсервера IIS із використанням технологій ASP.NET Core;
- методи та засоби забезпечення високої доступності та відмовостійкості вебзастосунку, зокрема механізми моніторингу та балансування навантаження;
- засоби, методи та технології, що застосовуються для розробки, розгортання, адміністрування та підтримки веборієнтованих систем у середовищі операційних систем Windows із використанням програмної платформи Microsoft.

Обладнання та програмне забезпечення:

1. Oracle VirtualBox
2. Windows Server 2019
3. Microsoft IIS
4. ASP.NET Core
5. Microsoft SQL Server

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Воротніков В.В.	22.03.2025 	22.03.2025 
2	Воротніков В.В.	05.04.2025 	05.04.2025 
3	Воротніков В.В.	01.05.2025 	01.05.2025 

Студент  Андрій ЛАТАШ
Керівник  Олексій ШЕЛУХА

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	22.03.2025	Виконано
2	Аналіз завдання та уточнення вимог для його виконання	27.03.2025	Виконано
3	Вибір оптимальних технологій	01.04.2025	Виконано
4	Аналіз наукових джерел	07.04.2025	Виконано
5	Інсталяція та конфігурація компонентів стеку	16.04.2025	Виконано
6	Створення бази даних	26.04.2025	Виконано
7	Розробка серверної частини	16.05.2025	Виконано
8	Налаштування інфраструктури вебсервера	23.05.2025	Виконано
9	Тестування функціональності	27.05.2025	Виконано
10	Оформлення пояснювальної записки	09.06.2025	Виконано

Студент  Андрій ЛАТАШ
Керівник  Олексій ШЕЛУХА

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра на тему «Проект інформаційної вебсистеми на базі рішень Microsoft» складається з вступу, трьох розділів основного тексту, висновків, списку використаної літератури та п'яти додатків. Текстова частина викладена на 80 сторінках друкованого тексту. Пояснювальна записка має 9 сторінок додатків. Список використаних джерел містить 31 найменувань і займає 3 сторінки. В роботі наведено 64 рисунки, 5 таблиць. Загальний обсяг роботи – 93 сторінки.

Метою кваліфікаційної роботи бакалавра є розробка інформаційної вебсистеми на базі рішень Microsoft.

Перший розділ присвячений загальному огляду сучасних технологій та рішень, які використовуються для створення вебсистем. Проаналізовано основні компоненти вебінфраструктури та обґрунтовано вибір технологічної платформи для реалізації проєкту.

Другий розділ зосереджено на створенні серверного середовища для розгортання вебзастосунку. Описано етапи налаштування операційної системи, встановлення необхідного програмного забезпечення та підготовки середовища для розміщення сайту.

Третій розділ описує процес розгортання вебзастосунку на сервері, а також впровадження інструментів для забезпечення стабільної та надійної роботи системи. Було реалізовано механізми балансування навантаження, моніторингу та маршрутизації. Виконано перевірку працездатності вебсистеми. Проведено тестування функціональності, доступності, стабільності роботи сайту та ефективності застосованих рішень щодо безпеки та стійкості до навантажень.

У висновку відображено підсумки виконаної роботи та розглянуто перспективи подальшого розвитку.

Ключові слова: ОПЕРАЦІЙНА СИСТЕМА, ВЕБСАЙТ, ВЕБСЕРВЕР, БАЗА ДАНИХ, БАЛАНСУВАННЯ НАВАНТАЖЕННЯ, МОНІТОРИНГ.

					ККІК<.КР.Б – 123 – 9 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Проект інформаційної вебсистеми на базі рішень Microsoft	Літ.	Арк.	Аркушів
Розроб.		Латаш А.А.		20.08				
Керівник		Шелуха О.О.		20.08			4	93
Реценз.		Пірог О.В.		20.08		ФІКТ, Гр. КІ-21-1		
Н. контр.		Шелуха О.О.		20.08				
Затверд.		Єфіменко А.А.		20.08				

ANNOTATION

The explanatory note to the bachelor's qualification work titled "Project of an Information Web System Based on Microsoft Solutions" consists of an introduction, four chapters, conclusions, a list of references, and appendices. The textual part is presented on 80 pages of printed text. The explanatory note includes 9 pages of appendices. The list of references contains 31 sources and occupies 3 pages. The work includes 64 figures and 5 tables. The total volume of the work is 93 pages.

The purpose of the bachelor's qualification work is the development of an information web system based on Microsoft technologies.

The first chapter is dedicated to a general overview of modern technologies and solutions used for building web systems. The main components of web infrastructure are analyzed, and the technological platform chosen for the implementation of the project is justified.

The second chapter focuses on the creation of the server environment for deploying the web application. It describes the steps of configuring the operating system, installing the necessary software, and preparing the environment for hosting the website.

The third chapter covers the deployment of the web application on the server, as well as the implementation of tools to ensure the stability and reliability of the system. Load balancing, monitoring, and routing mechanisms were implemented. The operability of the web system was verified. Functionality, availability, system stability, and the effectiveness of the applied security and load-resistance solutions were tested.

The conclusion presents a summary of the completed work.

Keywords: OPERATING SYSTEM, WEBSITE, WEBSERVER, DATABASE, LOAD BALANCING, MONITORING.

		Латаш А.А.			ККИКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП	8
РОЗДІЛ 1 АНАЛІЗ КЛЮЧОВИХ КОМПОНЕНТІВ ВЕБСТЕКУ	10
1.1 Огляд та порівняння сучасних вебсерверів	10
1.2 Порівняння серверних операційних систем	14
1.3 Вибір технологій для написання серверної частини	17
1.4 Вибір системи управління базами даних	21
1.5 Постановка технічного завдання	24
Висновки до першого розділу	25
РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА ПІДГОТОВКА СЕРВЕРНОЇ	
ІНФРАСТРУКТУРИ	27
2.1 Проєктування архітектури серверної інфраструктури.....	27
2.2 Підготовка серверної частини.....	28
2.3 Встановлення Ms SQL Server та створення бази даних	39
2.4 Реалізація програмного забезпечення серверної частини.....	49
Висновки до другого розділу	57
РОЗДІЛ 3 РОЗГОРТАННЯ САЙТУ НА ВЕБСЕРВЕРІ, НАЛАШТУВАННЯ	
МОНІТОРИНГУ ТА БАЛАНСУВАННЯ НАВАНТАЖЕННЯ	58
3.1 Встановлення та налаштування Application Request Routing (ARR)	58
3.2 Розгортання сайту на вебсервері	66
3.3 Тестування роботи вебсервера.....	71
Висновки до третього розділу	78
ВИСНОВКИ	80
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	81
ДОДАТКИ.....	84

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ARR - Application Request Routing

IIS – Internet Information Services

MS SQL - Microsoft SQL Server

VM – Virtual Machine

ОС – операційна система

СУБД – система управління базами даних

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Сучасний розвиток інформаційних технологій значно впливає на обмін інформацією та взаємодію між користувачами, роблячи вебтехнології важливою складовою сучасних інформаційних систем. Необхідність створення надійних, безпечних і ефективних вебсистем, здатних працювати під високим навантаженням і забезпечувати безперервний доступ користувачам, є однією з ключових вимог сучасної інформаційної інфраструктури. Вебзастосунки відіграють важливу роль у суспільстві, автоматизуючи різноманітні процеси та покращуючи взаємодію між користувачами і інформаційними ресурсами.

Метою кваліфікаційної роботи є дослідження та практична реалізація процесу створення і налаштування вебсервера на базі Internet Information Services (IIS), розгортання вебзастосунків, створених із використанням платформи ASP.NET Core та забезпечення їх стабільної і безпечної роботи у віртуалізованому середовищі. Проєкт спрямований на здобуття навичок адміністрування серверної інфраструктури, налаштування вебінфраструктури і впровадження механізмів моніторингу, балансування навантаження та захисту даних.

Для досягнення поставленої мети було визначено наступні завдання:

1. Провести аналіз проблематики та предметної області, що пов'язана з вебсерверами та хмарною інфраструктурою.
2. Розробити архітектуру та здійснити проєктування інформаційної вебсистеми на базі рішень Microsoft.
3. Реалізувати розгортання сайту на вебсервері, налаштувати моніторинг та балансування навантаження.
4. Провести тестування роботи вебсервера та вебзастосунку, перевірити роботу систем моніторингу та балансування навантаження.

Об'єктом дослідження є процес розгортання та налаштування вебзастосунків і серверної інфраструктури на базі технологій Microsoft, зокрема IIS та платформи ASP.NET Core.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Предметом дослідження є методи та засоби налаштування вебсервера IIS, розгортання вебзастосунків на платформі ASP.NET Core, а також використання інструментів моніторингу, балансування навантаження та забезпечення безпеки веб-систем.

У процесі виконання кваліфікаційної роботи було розроблено проект інформаційної вебсистеми з використанням IIS та платформи ASP.NET Core, що забезпечує стабільну та безпечну роботу вебзастосунку з підтримкою моніторингу і балансування навантаження.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ КЛЮЧОВИХ КОМПОНЕНТІВ ВЕБСТЕКУ

1.1 Огляд та порівняння сучасних вебсерверів

На сучасному етапі розвитку інформаційних технологій, коли цифрові сервіси, вебзастосунки та онлайн-платформи стали невіддільною частиною повсякденного життя, вебсервери відіграють ключову роль у забезпеченні стабільної, безпечної та швидкої взаємодії між користувачем і системою. Вони виконують важливі функції щодо обробки запитів, керування підключеннями, маршрутизації трафіку та масштабування застосунків. У зв'язку зі зростанням навантажень на вебінфраструктуру, дедалі більше уваги приділяється вибору оптимального вебсервера, його налаштуванню, безпеці та відмовостійкості.

На сьогодні найпопулярнішими вебсерверами (рис.1.1) є Nginx, Apache, Cloudflare Server, LiteSpeed та Microsoft IIS [1]. Вони використовуються для обробки HTTP-запитів, розгортання вебзастосунків і забезпечення доступу до вебсайтів. Ці сервери становлять основу сучасного інтернету, кожен має свої переваги для різних типів проєктів.

	2024 1 Jun	2024 1 Jul	2024 1 Aug	2024 1 Sep	2024 1 Oct	2024 1 Nov	2024 1 Dec	2025 1 Jan	2025 1 Feb	2025 1 Mar	2025 1 Apr	2025 1 May	2025 1 Jun
Nginx	34.2%	34.1%	34.0%	33.9%	33.8%	33.8%	33.8%	33.9%	33.9%	33.8%	33.8%	33.8%	33.8%
Apache	29.8%	29.5%	29.1%	29.0%	28.8%	28.7%	28.3%	27.5%	27.0%	26.8%	26.6%	26.4%	26.2%
Cloudflare Server	21.9%	22.2%	22.5%	22.6%	22.7%	22.7%	22.6%	22.9%	23.1%	23.3%	23.3%	23.6%	23.8%
LiteSpeed	13.2%	13.4%	13.5%	13.7%	13.8%	14.0%	14.0%	14.1%	14.3%	14.5%	14.6%	14.6%	14.6%
Node.js	3.3%	3.3%	3.4%	3.5%	3.5%	3.6%	3.8%	3.9%	4.1%	4.2%	4.3%	4.4%	4.5%
Microsoft-IIS	4.7%	4.6%	4.5%	4.5%	4.4%	4.4%	4.3%	4.2%	4.2%	4.1%	4.1%	4.0%	3.9%
Envoy	0.4%	0.4%	0.4%	0.5%	0.4%	0.4%	0.9%	1.2%	1.2%	1.2%	1.2%	1.3%	1.4%
Google Servers	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%
Caddy	0.2%	0.2%	0.2%	0.2%	0.2%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%
IdeaWebServer	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
Tengine	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
Kestrel	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%

Рисунок 1.1 – Рейтинг найпопулярніших вебсерверів [1]

Internet Information Services (IIS) – це вебсервер від Microsoft, який працює на операційних системах Windows. Він використовується для хостингу вебсайтів, веб-

застосунків та служб, підтримуючи як статичний, так і динамічний контент. У рейтингу найпопулярніших вебсерверів він посідає шосте місце, після Nginx, Apache, Cloudflare Server, LiteSpeed та Node.js.

Серед основних можливостей IIS варто виділити підтримку широкого спектру протоколів, включаючи HTTP, HTTPS, FTP, FTPS, SMTP, NNTP, а також новіші версії HTTP/2 і HTTP/3. Наявність модульної архітектури, починаючи з версії 7.0, дозволяє гнучко керувати встановленими компонентами, що знижує потенційну поверхню атаки та підвищує загальну продуктивність системи. Тісна інтеграція з платформою ASP.NET спрощує розгортання та адміністрування вебзастосунків, а підтримка додаткових технологій, таких як PHP, CGI, FastCGI і ISAPI, розширює можливості створення динамічного контенту. Особливу увагу приділено безпеці: IIS підтримує кілька методів автентифікації – базову, Windows-автентифікацію, а також автентифікацію за допомогою сертифікатів, і дозволяє використовувати TLS для захищеного з'єднання.

Серед переваг IIS можна відзначити його глибоку інтеграцію з Windows Server і Active Directory, що значно полегшує управління користувачами, правами доступу та політиками безпеки. Наявність зручного графічного інтерфейсу IIS Manager забезпечує комфортне конфігурування та моніторинг роботи вебсерверу. Завдяки підтримці масштабування і можливості налаштовувати кілька робочих процесів, IIS дозволяє ефективно розподіляти навантаження та використовувати ресурси системи. Також важливо, що останні версії сервера підтримують сучасні вебстандарти, включаючи HTTP/2, HTTP/3, WebSocket та інші технології, що забезпечує його актуальність у сучасному вебсередовищі [2].

Application Request Routing (ARR) – це потужне розширення для IIS, яке розширює стандартні можливості вебсервера, перетворюючи його на зворотний проксі та балансувальник навантаження. Його основне призначення – приймати HTTP-запити від клієнтів і перенаправляти їх на інші сервери в бекенді, тобто виконувати роль посередника між клієнтом і вебзастосунками, що розгорнуті на різних хостах.

ARR активно використовується у великих інфраструктурах, де один сервер IIS виступає вхідною точкою (gateway), що розподіляє навантаження між кількома

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

серверами застосунків. Він дозволяє здійснювати балансування навантаження з урахуванням стану серверів, обираючи найменш завантажені або найшвидші вузли. Це особливо корисно для забезпечення високої доступності та масштабованості вебсервісів.

Головні функції Application Request Routing охоплюють низку важливих можливостей, які роблять його ефективним інструментом для роботи з вебтрафіком. По-перше, ARR виконує роль зворотного проксі-сервера, приймаючи вхідні запити і перенаправляючи їх на відповідні бекенд-сервери. Він також забезпечує балансування навантаження між кількома серверами, використовуючи різні алгоритми, зокрема кругове опитування, розподіл за продуктивністю або з урахуванням стану доступності серверів. Однією з корисних функцій є можливість маршрутизації запитів на основі URL-адрес, заголовків, cookies та інших параметрів. ARR підтримує кешування відповідей, що дозволяє зменшити навантаження на сервери, обробляючи частину запитів без повторного звернення до бекенду. Крім того, він виконує SSL offloading, тобто знімає навантаження з додатків, розшифровуючи HTTPS-з'єднання на своєму рівні. Ще однією важливою функцією є інтеграція з Web Farm Framework, що забезпечує автоматизоване розгортання вебферм, моніторинг стану серверів і синхронізацію контенту між ними [3].

Безпека в IIS (Internet Information Services) є однією з головних переваг цього вебсервера. Вона реалізується на кількох рівнях-від базових механізмів автентифікації до гнучкої системи дозволів і захисту трафіку через SSL/TLS.

IIS підтримує різні типи автентифікації: анонімну, базову, формну, автентифікацію Windows. Це дозволяє гнучко обмежувати доступ до окремих ресурсів або застосунків, особливо якщо сервер працює у зв'язці з Active Directory. Крім цього, IIS дозволяє задавати детальні правила авторизації, які визначають, хто може звертатися до певних частин сайту, на основі ролей або груп користувачів.

Для додаткового захисту Microsoft пропонує URLScan і Request Filtering, які допомагають блокувати підозрілі запити ще до того, як вони будуть оброблені застосунком [4].

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтеграція з іншими продуктами Microsoft, IIS має тісний зв'язок із Microsoft SQL Server (MS SQL), що спрощує розгортання вебзастосунків, які зберігають і обробляють дані. Зокрема, вебдодатки на ASP.NET можуть легко працювати з базами даних SQL Server, а IIS забезпечує зручне середовище для їхнього хостингу. Крім того, IIS інтегрується з Windows Server, PowerShell (для автоматизації налаштувань), System Center (для моніторингу), Azure (для хмарного розгортання), а також з Visual Studio, що полегшує розробку, налагодження і публікацію застосунків безпосередньо на сервер. Така глибока взаємодія робить IIS ключовим елементом у побудові масштабованих, безпечних та інтегрованих рішень у середовищі Microsoft.

Серед недоліків IIS можна відзначити обмеження платформи, оскільки він функціонує виключно на операційних системах Windows, що звужує можливості вибору хостинг-середовищ. Незважаючи на підтримку сторонніх технологій, таких як PHP чи CGI, рівень інтеграції та гнучкість у використанні можуть поступатися іншим вебсерверам, зокрема Apache або Nginx. Також варто враховувати вартість ліцензування: розгортання IIS зазвичай передбачає використання Windows Server, що може бути значно дорожчим за безкоштовні альтернативи.

Серед найпопулярніших вебсерверів сьогодні лідирують Nginx, Apache та Cloudflare Server. Кожен з них має свої особливості та переваги, які визначають їх використання в різних середовищах.

Nginx став особливо популярним завдяки своїй високій продуктивності та здатності обробляти тисячі одночасних з'єднань з мінімальним споживанням ресурсів. Його архітектура на базі подій дозволяє ефективно працювати під великим навантаженням, що робить його ідеальним вибором для сайтів з високим трафіком. Nginx часто використовується як зворотний проксі або балансувальник навантаження перед іншими серверами чи додатками, включаючи ті, що працюють на Node.js або [5].

Apache, хоча й старіший за Nginx, досі залишається одним з найпоширеніших вебсерверів. Його перевага-це гнучкість у конфігурації та багатий набір модулів,

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

що дозволяє налаштувати сервер практично під будь-які потреби. Apache добре інтегрується з різними системами керування контентом, зокрема WordPress, і часто використовується в поєднанні з Linux [6].

Cloudflare Server не є типовим вебсервером у класичному розумінні, оскільки він працює в межах глобальної інфраструктури CDN (Content Delivery Network). Його головна функція-забезпечення швидкої доставки контенту, кешування та захист від DdoS-атак. Це серверна платформа, яка приймає запити на глобальному рівні та передає їх до оригінальних серверів (origin servers), таких як Nginx або Apache [7].

Якщо порівнювати ці сервери з Microsoft IIS, то слід зазначити, що IIS має найтіснішу інтеграцію з Windows-екосистемою. Це робить його найкращим вибором для застосунків, розроблених на .NET або для корпоративного середовища, де використовується Active Directory. Водночас, Nginx і Apache краще підходять для високонавантажених публічних сайтів, особливо на платформі Linux. Вони є більш універсальними з точки зору підтримки різних мов програмування та хостинг-середовищ. Cloudflare Server ж взагалі працює як глобальний захисний і оптимізуючий шар над іншими серверами, тому безпосередньо з IIS не конкурує, а скоріше доповнює його.

Таким чином, IIS має свої переваги в корпоративному середовищі, але поступається гнучкістю та продуктивністю в умовах великого навантаження, де краще себе проявляють Nginx і Apache.

1.2 Порівняння серверних операційних систем

У сучасному цифровому середовищі серверні операційні системи відіграють ключову роль у забезпеченні функціонування корпоративних мереж, вебресурсів, віртуалізованих середовищ і хмарних платформ. Вибір ОС визначає рівень надійності, безпеки, продуктивності та зручності адміністрування IT-інфраструктури. Існує кілька основних серверних операційних систем, які широко використовуються в сучасних IT-інфраструктурах. Серед них найбільшу популярність мають

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Linux, Windows Server, Unix та macOS Server. Кожна з них має свої особливості, переваги й сфери застосування.

Windows Server-це серверна операційна система, розроблена корпорацією Microsoft, яка служить основою для побудови надійної корпоративної інфраструктури. Вона підтримує широкий спектр ролей і функцій, таких як доменна служба Active Directory, файлові та принт-сервери, вебсервер (IIS), віртуалізація, засоби безпеки, віддалене керування та інше. Завдяки тісній інтеграції з іншими продуктами Microsoft, ця ОС часто є основою для корпоративного середовища на базі технологій Windows.

Windows Server створено для забезпечення централізованого управління комп'ютерними мережами. Його використовують у державних установах, підприємствах, банках, навчальних закладах-всюди, де потрібно об'єднати багато комп'ютерів, користувачів, даних і сервісів в єдину керовану систему.

Windows Server має низку ключових компонентів, які забезпечують його багатofункціональність та ефективність у корпоративному середовищі. Одним із центральних елементів є Active Directory (AD) - система, що дозволяє централізовано керувати обліковими записами користувачів і комп'ютерів, контролювати доступ до ресурсів та впроваджувати політики безпеки. Завдяки цьому адміністратор може управляти всією мережею з єдиного центру.

Ще одним важливим інструментом є Group Policy (GPO), яка дозволяє задавати правила для користувачів і комп'ютерів у межах домену. За допомогою GPO можна, наприклад, обмежити доступ до певних функцій системи, заборонити встановлення програм або налаштувати стандартний вигляд робочого столу.

Для віртуалізації в Windows Server інтегровано Hyper-V-гіпервізор, що дозволяє запускати кілька віртуальних машин на одному фізичному сервері. Це дає змогу значно зекономити ресурси та гнучко керувати серверною інфраструктурою [8].

Windows Server-це потужна, надійна платформа для побудови IT-інфраструктури будь-якого масштабу. Її головна перевага-глибока інтеграція з іншими продуктами Microsoft, централізоване управління та високий рівень безпеки. Вона

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

особливо популярна в корпоративному середовищі, де потрібна підтримка доменів, політик безпеки, віддаленого доступу та віртуалізації.

Linux є однією з найпотужніших і найгнучкіших платформ для серверного використання. Її популярність зумовлена низкою факторів, серед яких-відкритий вихідний код, високий рівень безпеки, масштабованість та гнучкість налаштувань. Linux є основою для більшості вебресурсів, хостинг-платформ, серверних інфраструктур, кластерних систем та хмарних середовищ.

На відміну від комерційних ОС, Linux доступна безкоштовно (у більшості дистрибутивів), що дозволяє суттєво знизити витрати на розгортання та обслуговування серверної інфраструктури. Водночас, завдяки відкритому коду, адміністратори мають змогу детально налаштовувати систему під власні потреби, змінювати її компоненти та адаптувати функціональність.

Дистрибутиви Linux - це різні версії операційної системи, створені для різних цілей. Наприклад, Ubuntu Server часто використовується завдяки простоті в налаштуванні та великій спільноті; Debian - за стабільність; CentOS/AlmaLinux і RHEL - у корпоративному середовищі за підтримку, довгострокові оновлення та високу надійність.

Linux підтримує широкий спектр серверного ПЗ - таких як Apache, Nginx, MySQL, PostgreSQL, Docker, Kubernetes, Samba, а також різноманітні засоби моніторингу, резервного копіювання та автоматизації (Ansible, Puppet, Zabbix тощо). Це дає змогу побудувати повноцінну серверну інфраструктуру без прив'язки до певного вендора.

Сервери на базі Linux відзначаються високою стабільністю в роботі - багато систем можуть безперервно функціонувати роками без потреби перезавантаження. Також Linux традиційно вважається безпечнішою ОС завдяки модульності, прозорості коду та активному виявленню вразливостей спільнотою [9].

Unix - це родина операційних систем, які є прабатьками Linux, але використовуються переважно у великих корпоративних середовищах та в галузях, де важлива безперебійна робота систем (наприклад, банківська справа, телекомунікації). До прикладу, Solaris (від Oracle) або AIX (від IBM) - це комерційні UNIX-системи,

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

що відзначаються надійністю, масштабованістю і високим рівнем безпеки, але мають високу вартість підтримки та менш гнучкі умови ліцензування [10].

macOS Server - це серверне розширення для macOS, яке орієнтоване здебільшого на освітні установи та малі офіси, що використовують екосистему Apple. З роками Apple значно скоротила функціональність цього продукту, тому його використання нині досить обмежене [11].

Windows Server вирізняється тісною інтеграцією з іншими продуктами Microsoft, простим графічним інтерфейсом і широкими можливостями для керування мережею через Active Directory, Group Policy та PowerShell. Він ідеально підходить для середовищ, де використовуються .NET-застосунки, MS SQL, Exchange Server та інші фірмові сервіси Microsoft.

Натомість Linux більше підходить для розробників, хостинг-провайдерів і компаній, які віддають перевагу відкритому коду, автоматизації через командний рядок і роботі з вебтехнологіями. Linux забезпечує більшу гнучкість, стабільність і безкоштовну ліцензію, але вимагає вищого рівня технічних знань.

Unix-системи залишаються актуальними для високонавантажених корпоративних систем із високими вимогами до безпеки та стабільності, хоча їх використання знижується через складність адміністрування та високу вартість.

Отже, Windows Server найкраще підходить для корпоративних середовищ із уже сформованою інфраструктурою Microsoft. У той час як Linux - гнучке, надійне й економне рішення, особливо для інтернет-серверів і хмарних середовищ. Unix лишається вибором для специфічних, критично важливих систем із особливими вимогами.

1.3 Вибір технологій для написання серверної частини

Оскільки для реалізації серверної частини було обрано операційну систему Windows Server та вебсервер IIS, логічним рішенням є використання вебплатформи від Microsoft – ASP.NET. Ця платформа забезпечує глибоку інтеграцію з інфраструктурою Windows, високу продуктивність, надійність і підтримку сучасних стандартів веброзробки. Крім того, ASP.NET добре поєднується з іншими продуктами

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Microsoft, такими як SQL Server, що дозволяє створити єдине, злагоджене середовище для розробки та розгортання вебзастосунків.

ASP.NET – це фреймворк для створення вебзастосунків, який входить до складу платформи .NET і дозволяє розробляти як прості сайти, так і складні корпоративні системи. Його основною мовою програмування є C#, яка забезпечує високу швидкодію, чітку структуру коду та надійність.

Платформа підтримує різні архітектурні підходи – MVC (Model-View-Controller), Web API, Razor Pages та Blazor, що дозволяє розробникам обирати найбільш зручний варіант реалізації в залежності від задачі. Завдяки ASP.NET Core платформа стала повністю кросплатформеною, тобто тепер її можна запускати не лише на Windows, а й на Linux чи macOS.

Ще однією перевагою ASP.NET є вбудовані засоби безпеки, такі як захист від XSS, CSRF, підтримка шифрування, автентифікації та авторизації через стандартні механізми (JWT, OAuth, OpenID Connect). У поєднанні з IIS це дозволяє ефективно контролювати доступ до ресурсів і забезпечити захищену взаємодію між клієнтом і сервером.

Розробка з використанням ASP.NET активно підтримується за допомогою середовища Visual Studio, яке надає інструменти для зручного дебагу, розгортання, роботи з базами даних та інтеграції зі службами Azure. Завдяки цьому процес розробки суттєво пришвидшується і стає більш контрольованим на всіх етапах.

Таким чином, ASP.NET – це надійне, масштабоване та безпечне рішення для реалізації серверної частини застосунку в середовищі Windows Server та IIS.

Окрім фреймворку від Microsoft, у веброзробці широко використовуються й інші популярні мови програмування та платформи, кожна з яких має свої переваги й характерні особливості [\[12\]](#).

Однією з найбільш популярних є Node.js – це JavaScript-платформа для серверної частини, яка базується на неблокуючій моделі обробки запитів. Вона відзначається високою продуктивністю в умовах великої кількості одночасних підключень, тому часто використовується у реальному часі – чатах, онлайн-іграх,

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

стрімінгу. Node.js має широку екосистему npm-пакетів, проте для великих, структурованих проєктів її підтримка менш строгих стандартів може ускладнювати масштабування та підтримку [13].

Ще одним поширеним рішенням є Python з фреймворками Django та Flask. Python вирізняється простотою синтаксису та швидкістю розробки. Django надає усі інструменти для створення вебзастосунків, включаючи ORM, панель адміністратора та автентифікацію. Це ідеальне середовище для MVP, прототипів і стартапів. Проте за продуктивністю ASP.NET, як правило, перевершує Python-рішення, особливо в навантажених системах [14].

Також варто згадати PHP, особливо з фреймворком Laravel. Це один із найстаріших інструментів для веброзробки, який залишається актуальним завдяки простоті, доступності хостингів та швидкому старту. Проте ASP.NET перевершує PHP у побудові масштабованих, безпечних і добре структурованих застосунків [15].

У середовищі ASP.NET робота з базами даних є ключовою складовою серверної логіки, адже більшість застосунків зберігають, обробляють і відображають дані. ASP.NET підтримує різні способи взаємодії з базами даних, зокрема через ORM-фреймворки (Object-Relational Mapping), прямі SQL-запити або сучасні API-підходи.

Найпоширенішим фреймворком для роботи з базами даних у .NET є Entity Framework (EF). Це потужний ORM, який дозволяє розробнику працювати з даними у вигляді об'єктів без потреби писати сирі SQL-запити. EF підтримує як Code First (створення структури бази даних на основі класів коду), так і Database First (створення моделей на основі вже наявної бази). Це спрощує розробку, дозволяє уникати помилок при роботі з SQL, а також забезпечує міграції – автоматизоване оновлення структури бази при зміні моделі [16].

Ще одним популярним фреймворком є Dapper – мікро-ORM, створений розробниками Stack Overflow. На відміну від EF, Dapper не приховує SQL-запити – він дозволяє писати їх напрямку, але значно полегшує прив'язку результатів запитів до C# об'єктів. Завдяки цьому Dapper забезпечує високу продуктивність і контроль,

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

що робить його ідеальним варіантом для систем із великим навантаженням або де потрібна максимальна швидкість виконання запитів [17].

Крім класичної роботи з реляційними БД (зокрема MS SQL Server), у середовищі ASP.NET активно застосовуються й GraphQL-API. На відміну від REST, де кожна кінцева точка відповідає за фіксований набір даних, GraphQL дозволяє клієнту самому формувати запити й отримувати лише потрібні поля. У .NET є підтримка GraphQL через бібліотеки, такі як HotChocolate, що інтегруються з Entity Framework або іншими джерелами даних. Це дає змогу створювати гнучкі, зручні для фронтенду API, особливо коли структура даних складна або залежить від динамічних умов.

Продуктивність – одна з ключових переваг ASP.NET, особливо його сучасної версії ASP.NET Core, яка була створена з нуля для досягнення високої швидкості, масштабованості та ефективності використання ресурсів. ASP.NET Core побудовано на кросплатформеному високопродуктивному вебсервері Kestrel, який працює напрямку поверх сокетів без додаткових шарів абстракції. Це дозволяє значно знизити затримки обробки запитів і краще використовувати апаратні ресурси. Також, це середовище орієнтоване на асинхронне програмування (async/await), що дозволяє ефективно обслуговувати тисячі одночасних запитів без блокування потоків. У порівнянні з традиційними синхронними моделями це дає значний виграш у продуктивності на високому навантаженні.

Отже, ASP.NET є сучасною, потужною та надійною вебплатформою від Microsoft, яка дозволяє ефективно створювати вебзастосунки, API та сервіси будь-якої складності. Завдяки глибокій інтеграції з інфраструктурою Windows Server та IIS, ASP.NET є ідеальним вибором для корпоративних рішень, особливо у середовищах, де вже використовуються продукти Microsoft, такі як SQL Server, Active Directory та Azure. Платформа вирізняється високою продуктивністю, безпекою, широкими можливостями масштабування та зручністю для розробників. Підтримка асинхронного програмування, розвинене середовище для тестування, розширювана архітектура на основі middleware, а також активна спільнота і

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

постійна підтримка з боку Microsoft роблять ASP.NET одним із найкращих варіантів для розробки сучасних вебрішень.

1.4 Вибір системи управління базами даних

Одним із ключових компонентів будь-якого вебзастосунку є система управління базами даних (СУБД), яка відповідає за зберігання, обробку та управління інформацією. Вибір відповідної СУБД напряму впливає на швидкодію, масштабованість, надійність та зручність підтримки програмного продукту. Сучасні СУБД дозволяють працювати з великими обсягами даних, забезпечують паралельну обробку запитів, резервне копіювання, захист даних та багато інших критично важливих функцій.

Існує два основні типи СУБД: реляційні (SQL) та нереляційні (NoSQL).

Реляційні СУБД, такі як Microsoft SQL Server, PostgreSQL, MySQL, зберігають дані у вигляді таблиць і використовують мову запитів SQL для взаємодії з ними. Вони добре підходять для структурованих даних, що мають чітко визначені зв'язки.

Нереляційні СУБД, такі як MongoDB, Redis або Cassandra, використовуються для зберігання неструктурованих або слабо структурованих даних, і часто застосовуються у високонавантажених або розподілених системах, таких як великі вебпортали, системи аналітики, чат-додатки тощо.

Серед найпоширеніших систем управління базами даних варто виділити PostgreSQL, MySQL та MongoDB - кожна з них має свої переваги, архітектурні особливості та сфери застосування.

PostgreSQL - це потужна об'єктно-реляційна СУБД з відкритим кодом, відома своєю розширюваністю, відповідністю SQL-стандартам і високою надійністю. Вона активно використовується в проєктах, де потрібна складна логіка збереження даних, підтримка нестандартних типів даних, а також у системах з високим навантаженням. PostgreSQL добре масштабується і підходить як для невеликих стартапів, так і для великих корпоративних рішень [\[18\]](#).

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

MySQL - ще одна популярна реляційна СУБД, відома своєю швидкістю роботи, простотою налаштування та широкою підтримкою хостинг-провайдерів. MySQL часто використовується в поєднанні з PHP у LAMP-стосі та підходить для динамічних сайтів, CMS, інтернет-магазинів. Проте, в деяких випадках їй бракує гнучкості та функціональності, яку пропонують більш потужні СУБД, зокрема PostgreSQL чи MS SQL [\[19\]](#).

MongoDB - це представник нереляційних СУБД (NoSQL), яка зберігає дані у вигляді JSON-подібних документів. Вона ідеально підходить для застосунків з гнучкою структурою даних, високою швидкістю запису й масштабуванням у хмарі. MongoDB широко використовується в сучасних вебдодатках, особливо тоді, коли структура даних змінюється динамічно або коли потрібно зберігати великі обсяги неструктурованої інформації [\[20\]](#).

На відміну від вищезгаданих рішень, Microsoft SQL Server пропонує глибоку інтеграцію з екосистемою Microsoft, має потужні засоби адміністрування, високий рівень безпеки, розвинені можливості аналітики, тригери, збережені процедури та масштабування. Особливо це зручно у поєднанні з ASP.NET, IIS, Windows Server, а також із Visual Studio як основним середовищем розробки.

Microsoft SQL Server (MS SQL) - це потужна реляційна система управління базами даних (СУБД), розроблена корпорацією Microsoft, яка широко використовується в корпоративних та державних інфраструктурах. Вона надає широкі можливості для роботи з даними: зберігання, обробка, захист, реплікація, резервне копіювання, аналітика тощо. Залежно від потреб і масштабів проєкту Microsoft пропонує кілька редакцій SQL Server, кожна з яких має свою специфіку, набір функцій і ліцензійні умови [\[21\]](#).

Microsoft SQL Server Enterprise Edition - це найповніша та найпотужніша редакція SQL Server, яка орієнтована на великі підприємства, корпорації та критично важливі системи. Вона підтримує необмежену кількість ядер, до 524 петабайт даних і максимальну кількість оперативної пам'яті, доступної в системі. До ключових можливостей належать кластеризація з високою доступністю (Always On

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

Availability Groups), масштабована реплікація, підтримка In-Memory OLTP (для високопродуктивної обробки транзакцій у пам'яті), розширена безпека (Transparent Data Encryption, row-level security), а також повна інтеграція з аналітичними службами (SQL Server Analysis Services) та службами машинного навчання. Ця версія ідеально підходить для побудови великих data warehouse, фінансових систем, ERP/CRM та сервісів, які мають підвищені вимоги до надійності, масштабування та безпеки.

Microsoft SQL Server Standard Edition - це більш доступна версія, яка забезпечує ключовий функціонал для більшості середніх і малих організацій. Вона підтримує до 24 ядер і має обмеження на використання оперативної пам'яті (до 128 ГБ для рушія бази даних). Незважаючи на обмеження, вона надає широкий набір функцій, зокрема: базове резервне копіювання, журналювання, відновлення, підтримку Reporting Services та Integration Services, підтримку OLTP і прості варіанти кластеризації. Ця версія підходить для типових корпоративних систем, сайтів, облікових програм і середніх баз даних, де не потрібні найвищі рівні продуктивності.

Microsoft SQL Server Web Edition спеціально призначена для вебхостинг-провайдерів та розробників вебзастосунків. Вона є оптимізованим варіантом за доступною ціною для розміщення сайтів, інтернет-магазинів, CMS-систем тощо. Її функціонал обмежений, зокрема в області бізнес-аналітики, кластеризації, індексації великих обсягів даних і інтеграції з зовнішніми аналітичними системами. Однак, для вебпрограм із помірним навантаженням вона забезпечує достатній рівень функціональності й продуктивності.

Microsoft SQL Server Express Edition - це безкоштовна, легка версія SQL Server, яка ідеально підходить для навчальних цілей, локальних застосунків або невеликих стартапів. Вона має суттєві обмеження: максимальний розмір бази даних - 10 ГБ, підтримка одного процесора та до 1 ГБ оперативної пам'яті для рушія. Відсутня підтримка аналітичних сервісів, високої доступності, автоматизованого резервного копіювання тощо. Однак Express дозволяє створювати повноцінні бази

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

даних і взаємодіяти з ними через ті ж самі інструменти (SSMS, T-SQL), що й старші версії.

Microsoft SQL Server Developer Edition є повністю функціональною копією Enterprise, але призначена виключно для розробки, тестування й вивчення. Вона безкоштовна, але не може використовуватися в продуктивному середовищі. Її головна перевага - це можливість ознайомлення з повним функціоналом SQL Server без ліцензійних витрат, що особливо корисно для студентів, викладачів і розробників, які працюють над прототипами чи навчальними проєктами.

У підсумку, Microsoft SQL Server - це багаторівнева платформа, яка пропонує рішення як для простих застосунків, так і для масштабних підприємств. У межах даного проєкту було обрано Microsoft SQL Server Express Edition як систему управління базами даних. Ця редакція є оптимальним вибором для невеликих вебзастосунків завдяки своїй безкоштовності, простоті розгортання та сумісності з іншими продуктами Microsoft, зокрема ASP.NET та IIS. Express Edition забезпечує базову функціональність для зберігання та обробки даних користувачів, автентифікації, а також інших сутностей, необхідних для роботи серверної частини. Незважаючи на певні обмеження, такі як обсяг бази даних до 10 ГБ, вона повністю задовольняє вимоги проєкту на поточному етапі розробки [22].

1.5 Постановка технічного завдання

В умовах стрімкого розвитку інформаційних технологій спостерігається зростання попиту на стабільні, безпечні та масштабовані вебсистеми. З огляду на збільшення обсягу цифрових сервісів, кількість користувачів та рівень очікуваної доступності, особливу актуальність набувають питання надійного хостингу, балансування навантаження та централізованого моніторингу. Сучасні вебсервери повинні не лише обробляти запити, а й гарантувати безперервність доступу, обробку великого трафіку та захищеність даних користувачів.

Особливо важливою є інтеграція таких систем у корпоративне середовище, де використовується інфраструктура Microsoft - зокрема, Windows Server, IIS, SQL

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Server та ASP.NET Core. Рішення на базі цієї екосистеми дозволяють гнучко масштабувати сервіси, використовувати інструменти централізованого адміністрування та забезпечувати високий рівень продуктивності.

Отже, було сформульовано наступну мету роботи – розробка проєкту інформаційної вебсистеми на базі рішень Microsoft, що дозволить створити функціональне середовище для розгортання вебдодатків із підтримкою балансування навантаження, моніторингу та резервування.

Для досягнення поставленої мети було визначено наступні завдання:

5. Провести аналіз проблематики та предметної області, що пов’язана з веб-серверами та хмарною інфраструктурою.

6. Розробити архітектуру та здійснити проєктування інформаційної вебсистеми на базі рішень Microsoft.

7. Реалізувати розгортання сайту на вебсервері, налаштувати моніторинг та балансування навантаження.

8. Провести тестування роботи вебсервера та вебзастосунку, перевірити роботу систем моніторингу та балансування навантаження.

Виконання зазначених завдань дозволить комплексно вирішити проблему створення сучасної вебінфраструктури, здатної забезпечити безперервну та надійну роботу вебсистеми в умовах високого навантаження. Реалізація функцій балансування трафіку, централізованого моніторингу та підтримки масштабування сприятиме підвищенню стабільності, продуктивності сервісу, що є критично важливим у контексті сучасних вимог до доступності інформаційних ресурсів.

Висновки до першого розділу

У цьому розділі також було здійснено аналіз і обґрунтовано вибір технологій для реалізації серверної частини вебзастосунку. Основною платформою розробки було обрано ASP.NET, що забезпечує високу продуктивність, масштабованість і глибоку інтеграцію з екосистемою Microsoft. Цей вибір логічно впливає з того, що основою інфраструктури є Windows Server та вебсервер IIS, з якими ASP.NET працює найбільш ефективно.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

Для зберігання та обробки даних було обрано Microsoft SQL Server Express Edition, який забезпечує необхідний функціонал для роботи з базами даних у межах проєкту. Його використання дозволяє реалізувати надійну систему зберігання інформації про користувачів та інші сутності, необхідні для функціонування сервісу. Було розглянуто та порівняно альтернативні рішення, зокрема PostgreSQL, MySQL та MongoDB. Вибір MS SQL зумовлено потребою в сумісності з іншими компонентами платформи, зручністю адміністрування та підтримкою складних реляційних структур.

Розробка серверної логіки передбачає використання таких фреймворків, як Dapper для доступу до бази даних, а також GraphQL для ефективної побудови API. Застосування цих інструментів дозволяє реалізувати як просту, так і гнучку взаємодію між клієнтською частиною та сервером, що є важливою вимогою до сучасних вебзастосунків.

Окрім побудови бекенду та бази даних, подальші етапи проєкту включатимуть реалізацію балансування навантаження, що дозволить розподіляти запити між кількома інстанціями серверів для підвищення надійності та масштабованості системи. Також необхідно буде налаштувати моніторинг та логування, щоб своєчасно виявляти збої, слідкувати за станом сервера та реагувати на потенційні загрози.

Таким чином, у результаті аналізу було обрано оптимальний стек технологій, який забезпечить ефективну, безпечну та масштабовану реалізацію серверної частини вебзастосунку.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА ПІДГОТОВКА СЕРВЕРНОЇ ІНФРАСТРУКТУРИ

2.1 Проєктування архітектури серверної інфраструктури

Проєктування серверної інфраструктури є важливим етапом у створенні надійної, масштабованої та ефективної веб-системи. У рамках даного проєкту розроблено базову архітектуру серверної інфраструктури з використанням засобів Microsoft, зокрема веб-сервера IIS (Internet Information Services), модуля Application Request Routing (ARR) та платформи ASP.NET. Уся інфраструктура базується на клієнт-серверній моделі із впровадженням балансування навантаження між кількома вузлами.

На зображеній схемі (рис.2.1) показано загальну архітектуру системи. Клієнт надсилає HTTP-запити до балансувальника навантаження, який реалізовано за допомогою ARR у середовищі IIS. Балансувальник, у свою чергу, розподіляє трафік між двома незалежними веб-серверами, кожен з яких виконує роль хосту для ASP.NET-додатку. Обидва екземпляри додатку мають однакову логіку та підключені до централізованої бази даних, що реалізована на основі Microsoft SQL Server.

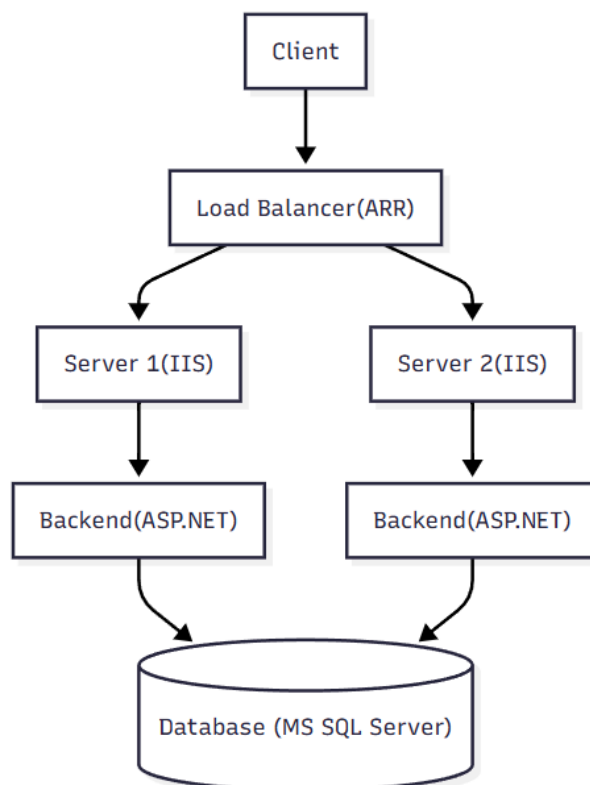


Рисунок 2.1 – схема архітектури серверної інфраструктури

Такий підхід дозволяє не лише забезпечити високу доступність сервісу, але й досягти рівномірного розподілу навантаження між вузлами, зменшити час відгуку системи та уникнути перевантаження окремих компонентів. Крім того, централізоване зберігання даних гарантує узгодженість інформації незалежно від того, через який сервер відбувається обробка запиту.

Обрана архітектурна модель відповідає сучасним принципам побудови високонавантажених веб-застосунків, є масштабованою та легко адаптується до змін у трафіку або до додавання нових серверів без порушення цілісності сервісу.

2.2 Підготовка серверної частини

2.2.1 Встановлення системи Windows Server

Початковим етапом у процесі розгортання серверної частини вебзастосунку є встановлення серверної операційної системи (рис.2.2).

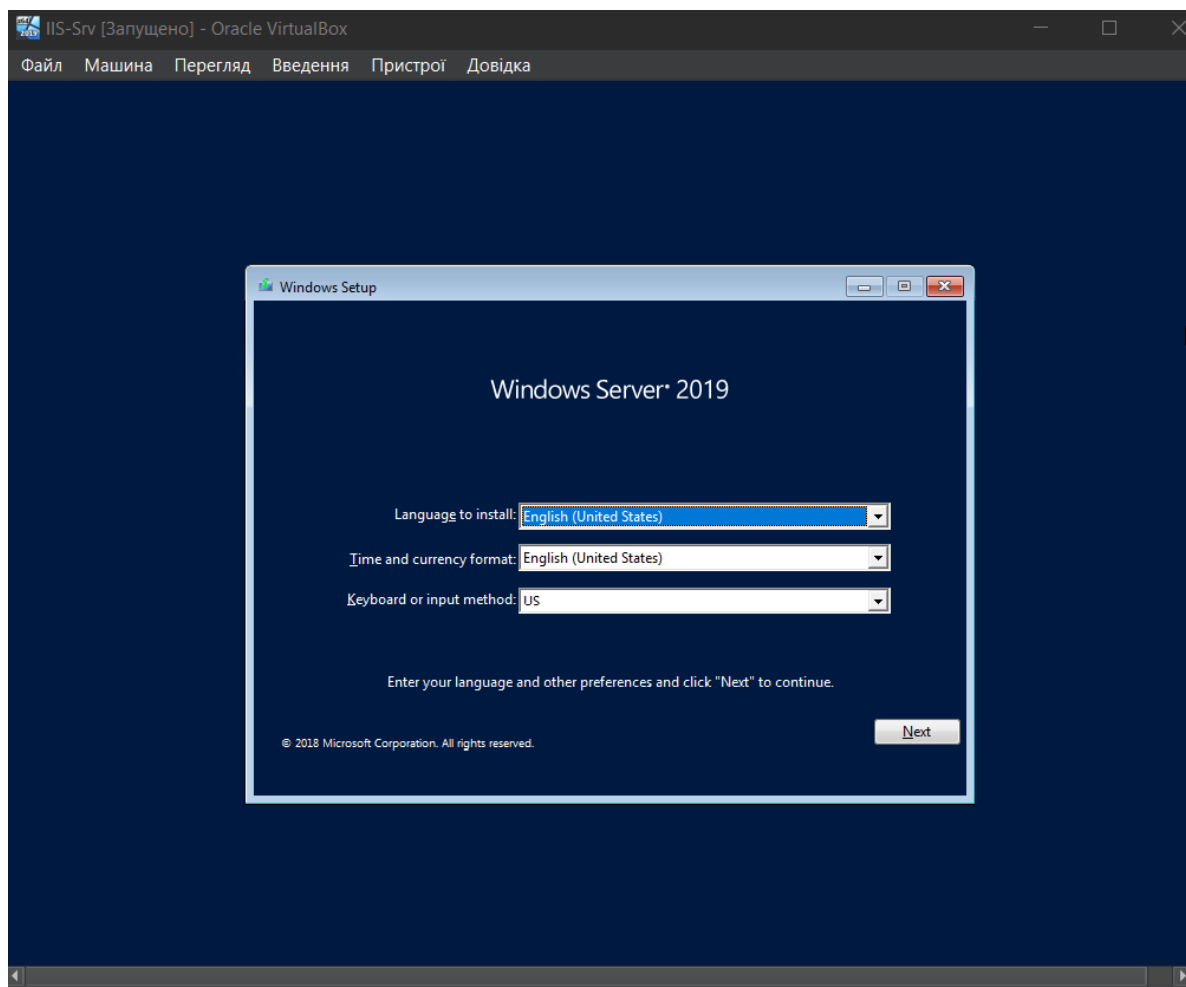


Рисунок 2.2 – Початок встановлення Windows Server

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

У межах цього проєкту для побудови інфраструктури обрано Windows Server 2019 - стабільну та функціональну платформу, що ідеально підходить для розгортання корпоративних сервісів, зокрема вебсервера IIS та бази даних SQL Server. Встановлення відбувається у віртуальному середовищі VirtualBox, що дозволяє ізолювати сервер від основної системи та зручно тестувати конфігурації.

Під час встановлення Windows Server необхідно обрати одну з доступних редакцій операційної системи (рис.2.3). Серед чотирьох запропонованих варіантів було обрано Windows Server 2019 Standard (Desktop Experience).

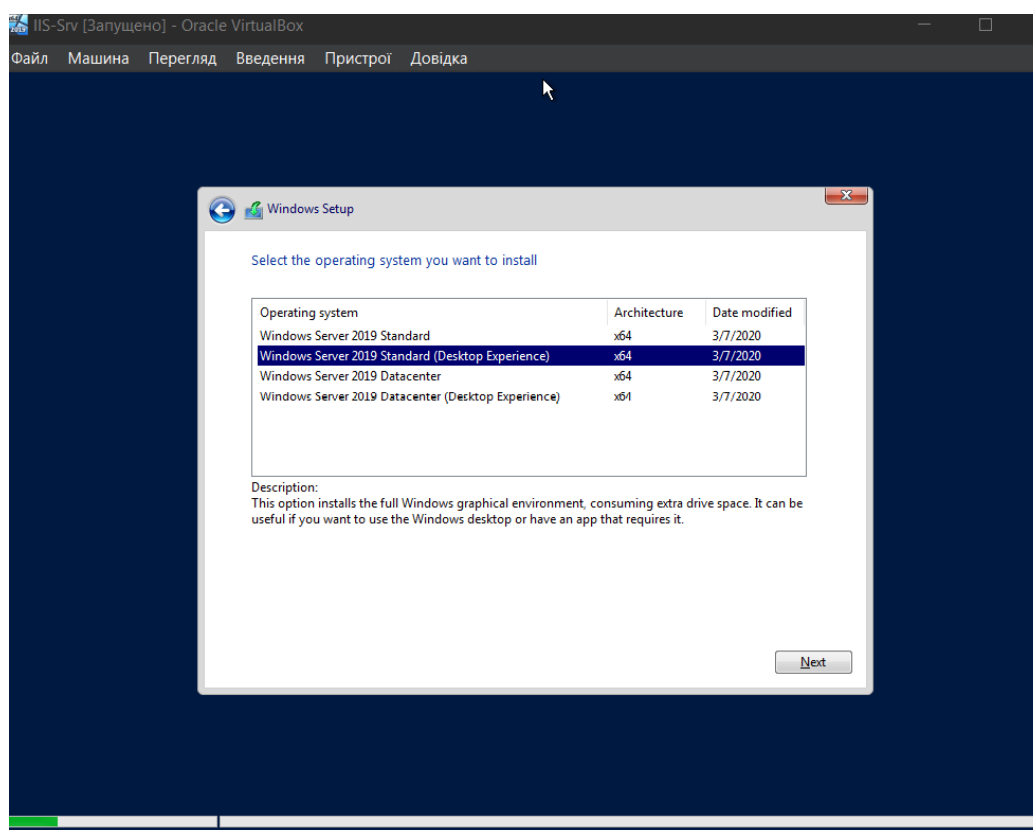


Рисунок 2.3 – Вибрана операційна система Windows Server 2019 Standard (Desktop Experience)

Windows Server 2019 Standard (Desktop Experience) - Це редакція Windows Server, яка поєднує базову функціональність серверної операційної системи з класичним графічним інтерфейсом Windows. Вона орієнтована на малі та середні організації, які потребують стабільного, керованого середовища для запуску критично важливих служб, таких як файлові сервери, контролери домену, вебсервери (IIS), системи резервного копіювання та інші базові інфраструктурні ролі.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

Ця редакція має графічну оболонку, що дозволяє керувати системою без необхідності використовувати командний рядок або PowerShell, що є особливо зручним для новачків або в освітніх цілях. Вона підтримує віртуалізацію і дає змогу створювати до двох віртуальних машин на одному фізичному сервері без додаткових ліцензій. Також реалізована підтримка широкого спектра ролей і служб, включно з Active Directory, DNS, DHCP, IIS, Hyper-V, файловими службами та іншими. У сфері безпеки передбачено базовий функціонал, включаючи Windows Defender, BitLocker, шифрування дисків та підтримку групових політик (GPO). Проте ця редакція має певні обмеження: вона не підтримує розширені функції, такі як Software-defined networking, Shielded VMs або Storage Spaces Direct. Це найкращий варіант для розгортання внутрішніх корпоративних сервісів, серверів додатків або вебплатформ, де немає потреби в надвисокій масштабованості або складних сценаріях віртуалізації.

Windows Server 2019 Datacenter (Desktop Experience) - Ця редакція призначена для великих підприємств, дата-центрів або хмарних середовищ, які мають високі вимоги до масштабованості, віртуалізації, зберігання даних та безпеки. Вона також має повноцінний графічний інтерфейс, як і Standard (Desktop Experience), але відрізняється набором підтримуваних технологій.

Однією з головних переваг є можливість запуску необмеженої кількості віртуальних машин, що робить її ідеальною для віртуалізованих середовищ із великою кількістю VM на одному фізичному сервері. Ця редакція підтримує розширені засоби віртуалізації, зокрема Shielded Virtual Machines, які забезпечують захист вмісту віртуальних машин навіть від адміністратора хост-системи. За допомогою технології Storage Spaces Direct можна створювати високонадійні й масштабовані кластери зберігання даних без потреби у зовнішньому сховищі (SAN). Крім того, реалізована підтримка мережевої віртуалізації через Software-Defined Networking (SDN), включаючи мікросегментацію, внутрішнє балансування навантаження та гнучке керування мережевими ресурсами. Для критично важливих сервісів передбачена підтримка реплікації та побудови високодоступних кластерів, що забезпечує надійність і безперервність роботи [23].

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

Ця версія є ідеальною для побудови великомасштабних інфраструктур, хмарних рішень або критичних додатків, які повинні працювати безперервно та мати високий рівень захисту.

Табл.1. Порівняння основних редакцій

Характеристика	Standard (Desktop Experience)	Datacenter (Desktop Experience)
Графічна оболонка	Так	Так
Віртуальні машини	До 2	Необмежено
Захищені ВМ (Shielded VMs)	Ні	Так
Storage Spaces Direct (Прямі простори зберігання)	Ні	Так
Реплікація та кластеризація	Базова	Повна

Під час встановлення обираємо тип інсталяції Custom (рис.2.4)

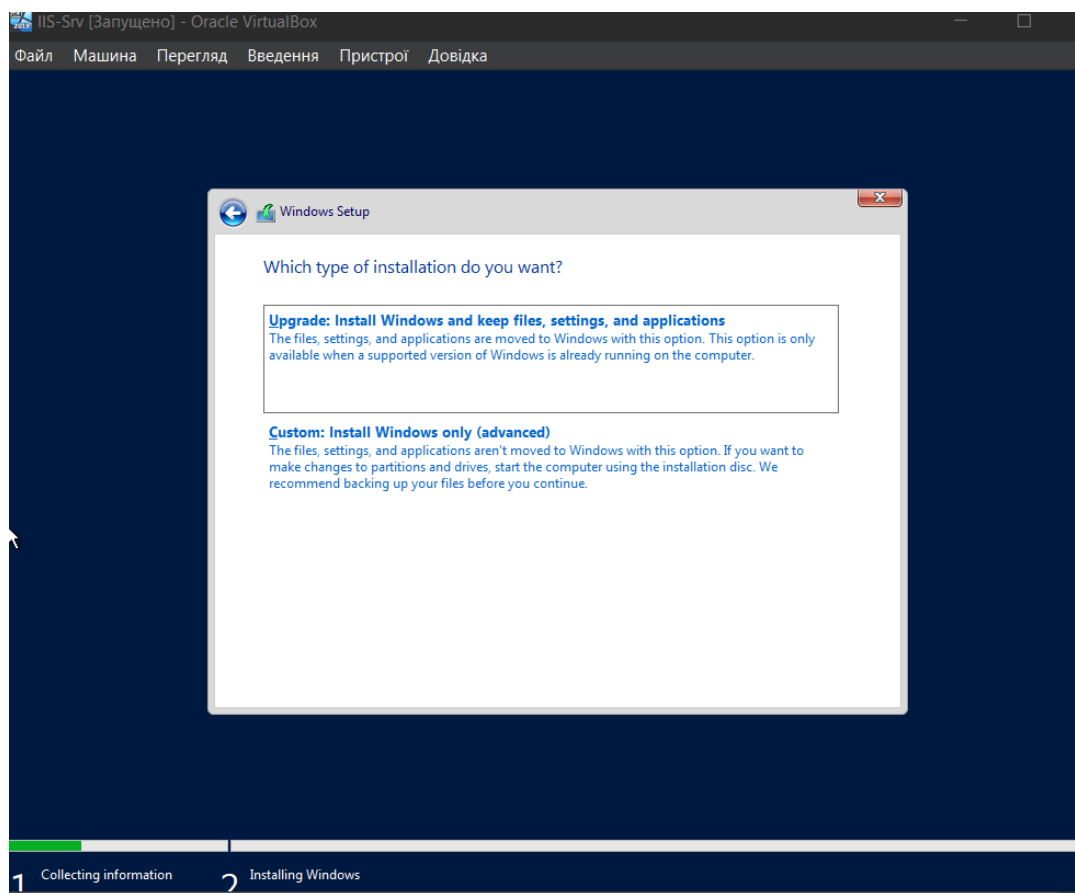


Рисунок 2.4 – Вибір типу інсталяції – Custom

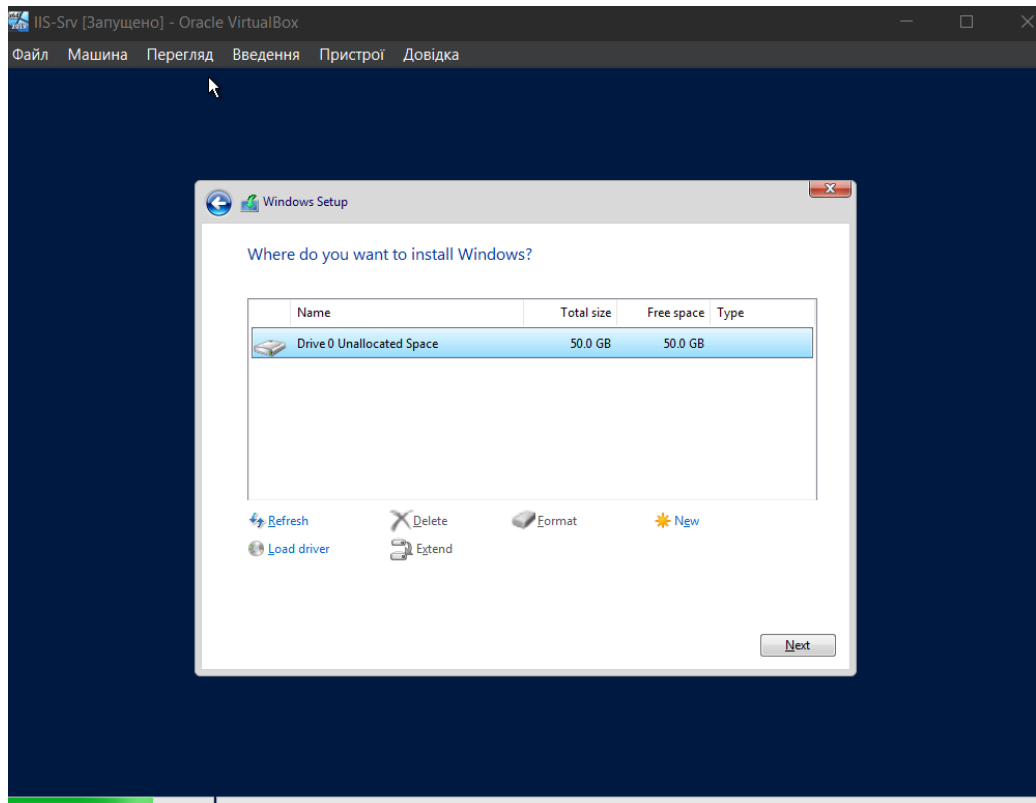


Рисунок 2.5 – Дискові розділи залишаємо без змін

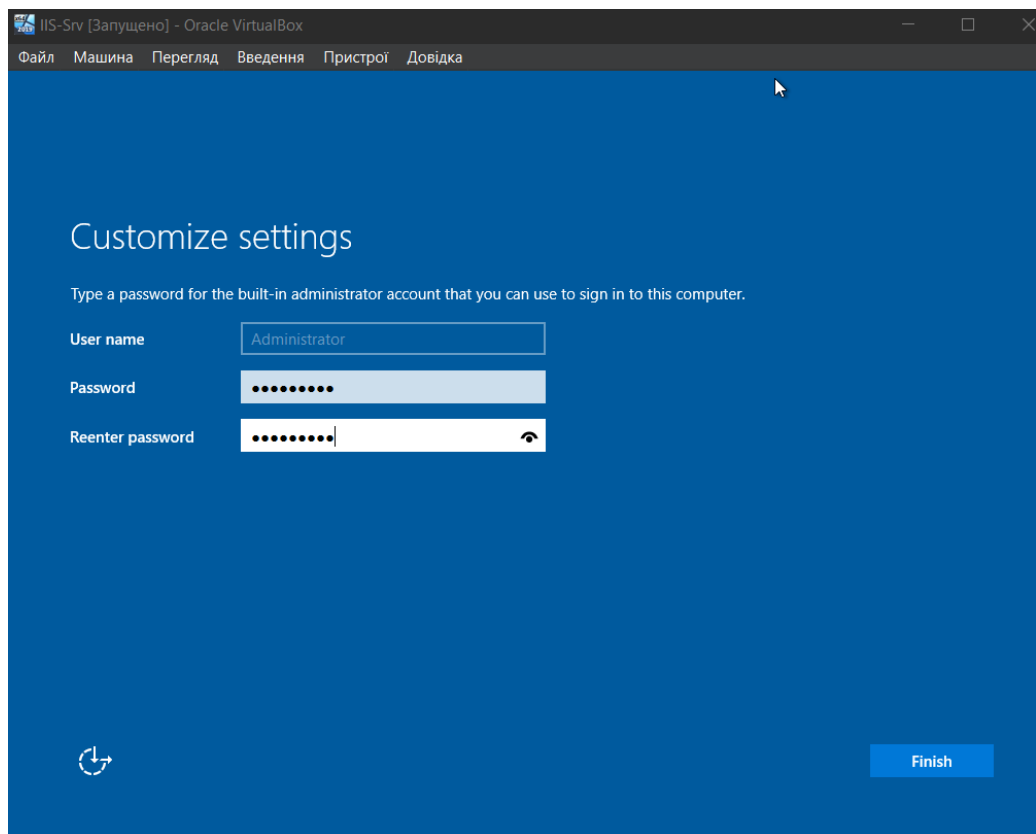


Рисунок 2.6 – Створення паролю для Administrator

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

У цьому підрозділі було здійснено встановлення операційної системи Windows Server 2019, що є основою для побудови серверної частини проєкту. Для забезпечення графічного інтерфейсу та широких функціональних можливостей було обрано редакцію Windows Server 2019 Standard (Desktop Experience). Цей варіант оптимально підходить для середніх і великих інфраструктур, забезпечує підтримку ключових компонентів, таких як Active Directory, IIS, Hyper-V та інші. Було створено пароль для користувача Administrator. Цей користувач є головним адміністратором сервера.

2.2.2 Додавання ролей Windows Server

Після успішного встановлення операційної системи Windows Server наступним кроком є конфігурація її функціональних можливостей шляхом додавання необхідних ролей. Ролі Windows Server - це набір служб, які забезпечують виконання певних завдань у серверному середовищі, таких як керування користувачами, надання вебсервісів, віртуалізація, зберігання даних тощо (рис.2.7, 2.8).

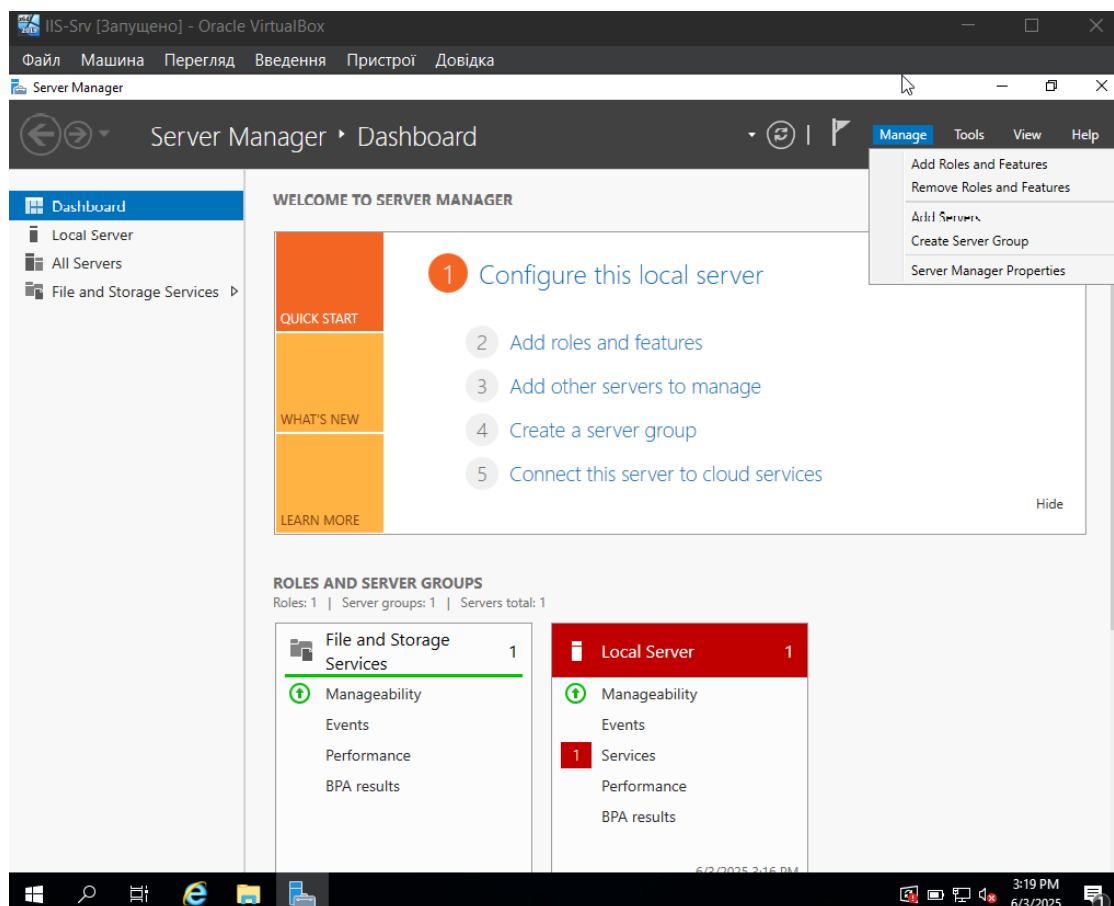


Рисунок 2.7 – Server Manager

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

У рамках цього проєкту ключовою роллю є вебсервер IIS (Internet Information Services), який використовується для розгортання серверної частини вебзастосу- нку. Додавання ролей дозволяє адаптувати сервер до конкретних потреб проєкту та забезпечити необхідну функціональність для подальшої розробки, налаштування та експлуатації вебінфраструктури (рис.2.10 – 2.17).

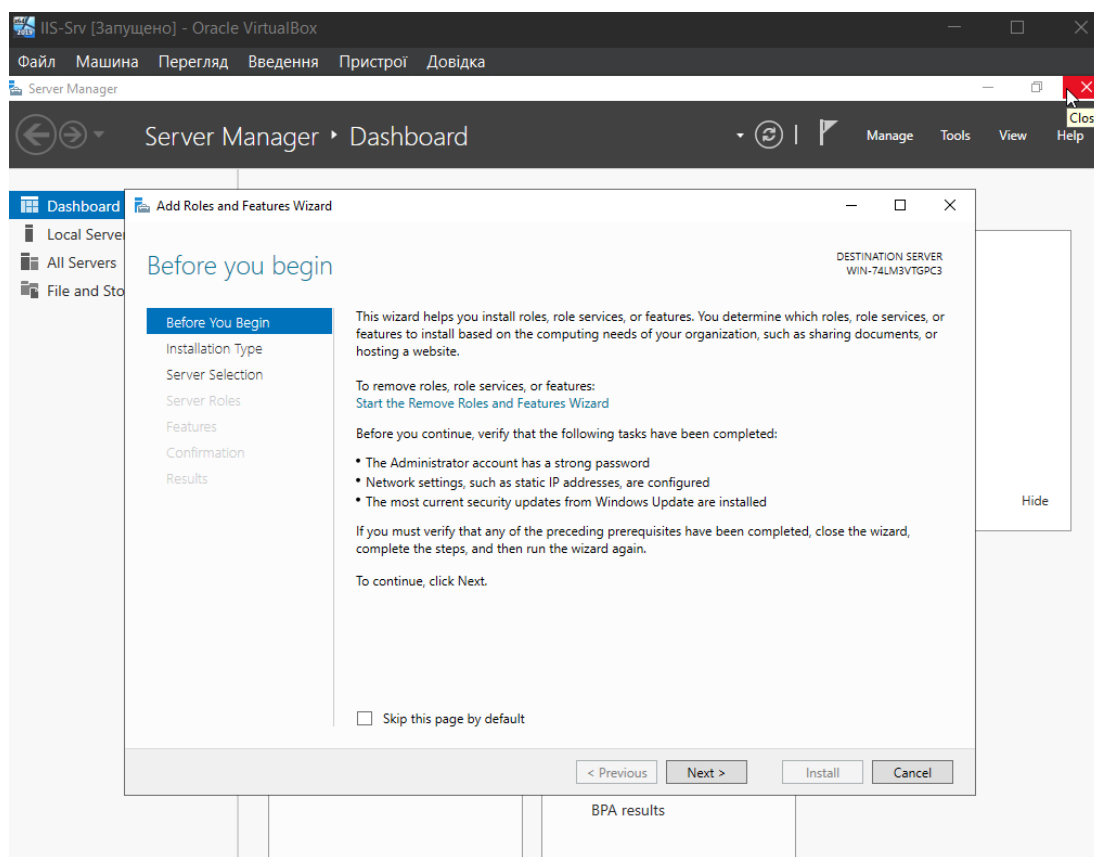


Рисунок 2.8 – Додавання ролей

На цьому кроці (рис.2.9) пропонується обрати один із двох варіантів:

Role-based or feature-based installation - Цей варіант використовується най- частіше і саме його слід обирати для типових сценаріїв, коли потрібно встановити ролі (наприклад, вебсервер IIS) або функції на окремому сервері. Це дозволяє на- лаштувати один фізичний або віртуальний сервер, додаючи йому нові ролі, сервіси ролей або компоненти Windows.

Remote Desktop Services installation. Цей варіант призначений для встанов- лення компонентів інфраструктури віддаленого робочого столу (VDI). Він викори- стовується у випадках, коли потрібно організувати віртуальні робочі місця для ко- ристувачів з централізованим адмініструванням [24] .

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

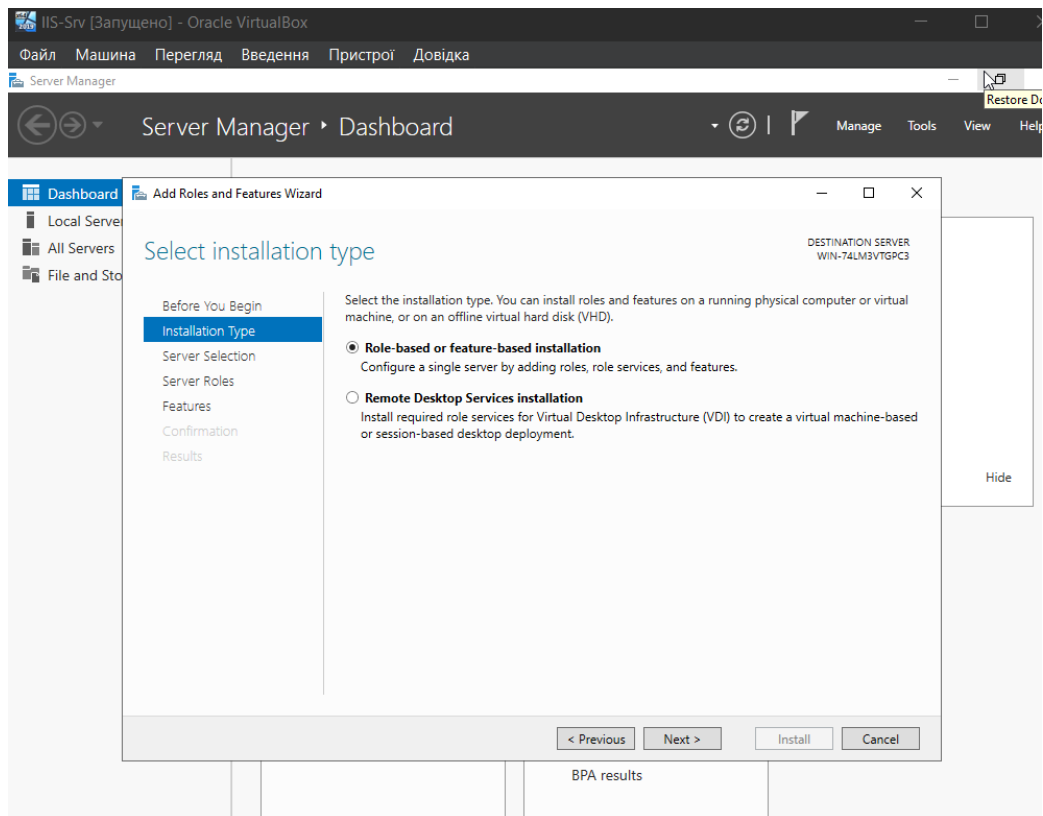


Рисунок 2.9 – Обираємо тип встановлення

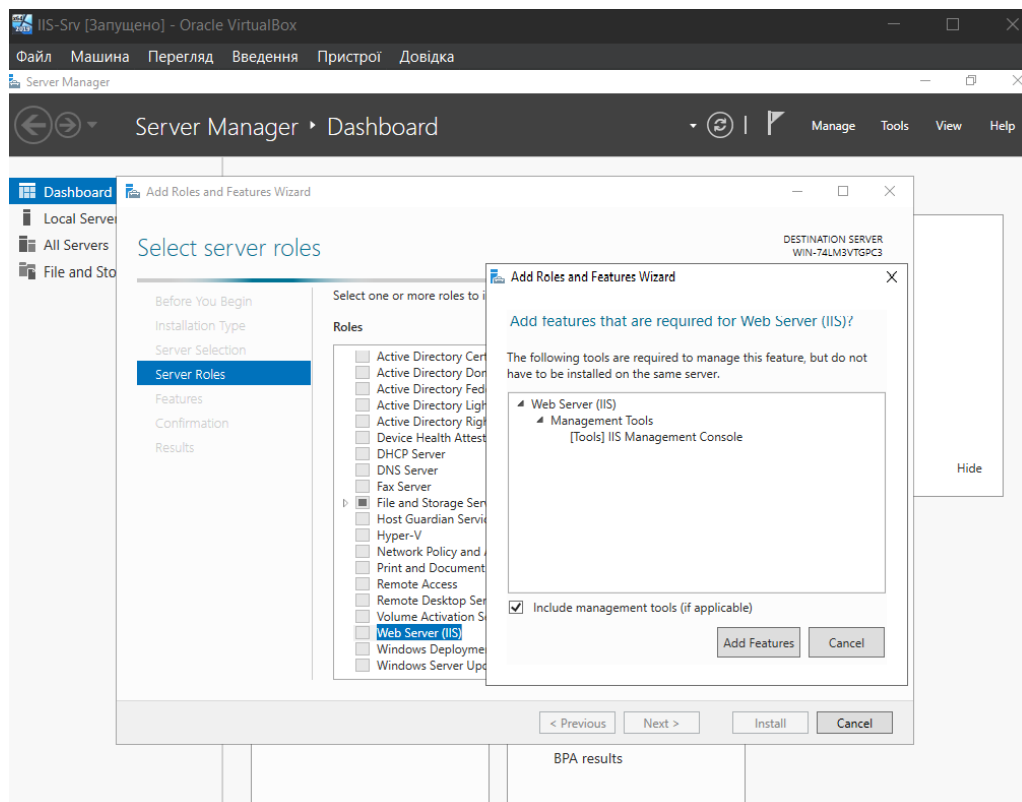


Рисунок 2.10 – Додаємо роль IIS

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

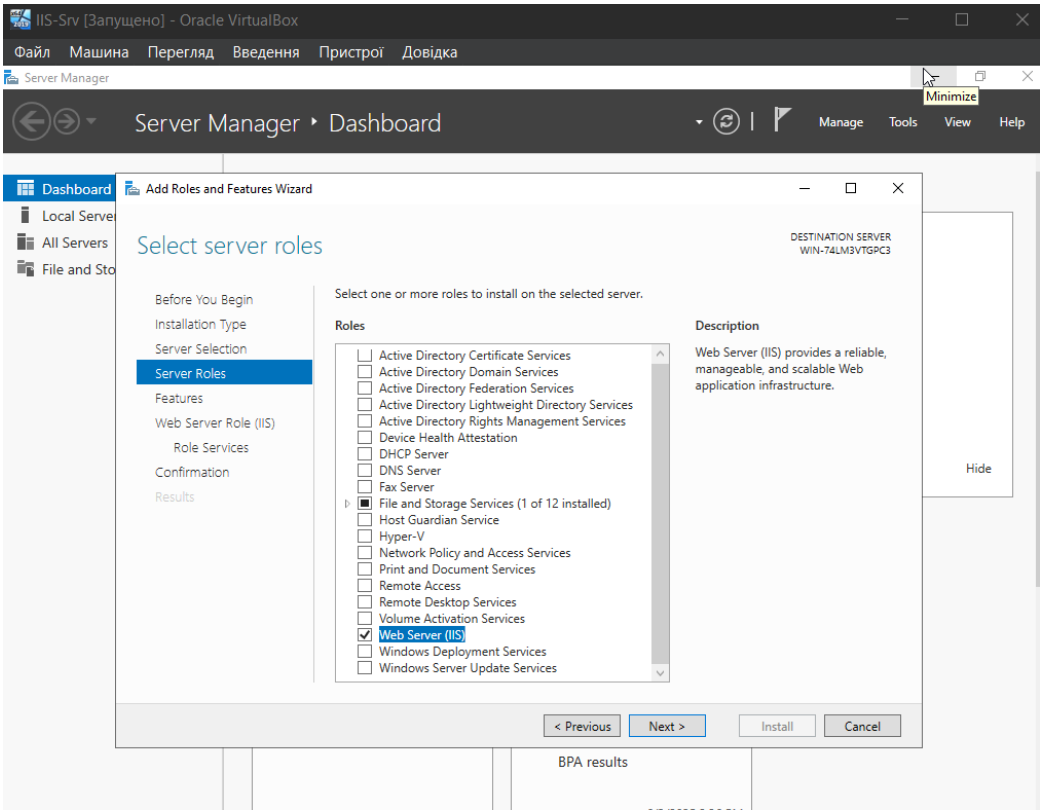


Рисунок 2.11 – Додаємо роль IIS

Обираємо потрібні компоненти

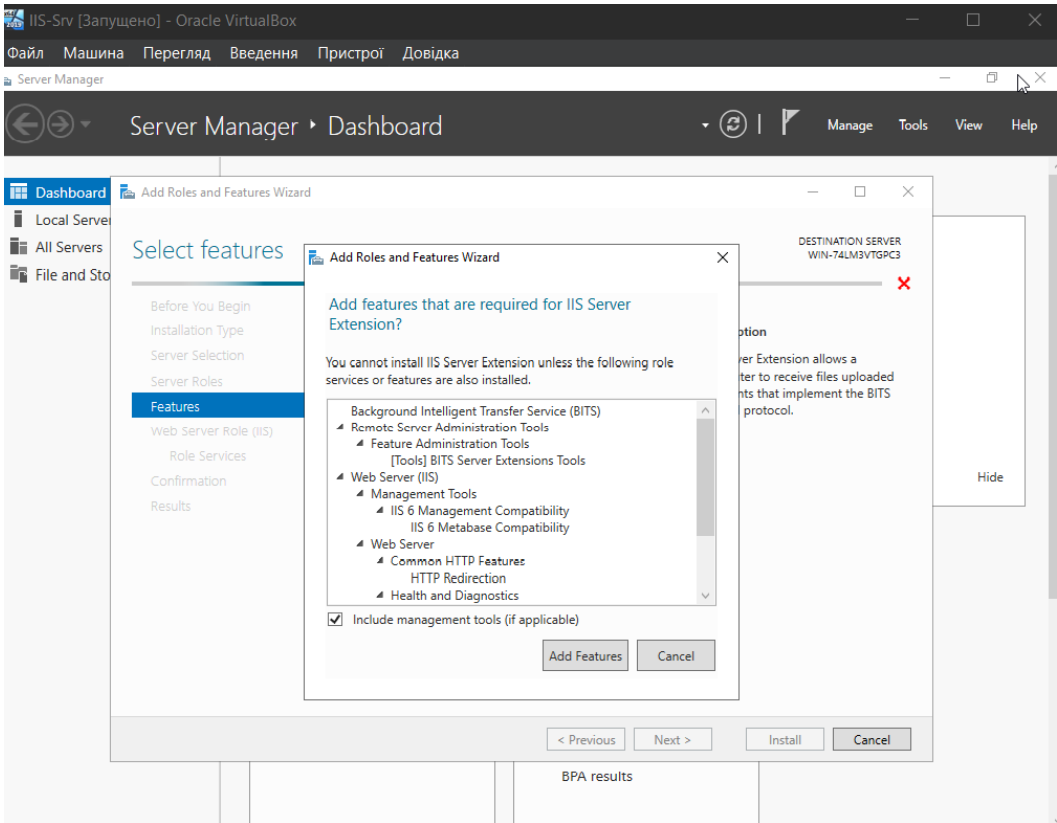


Рисунок 2.12 – Додавання компонентів

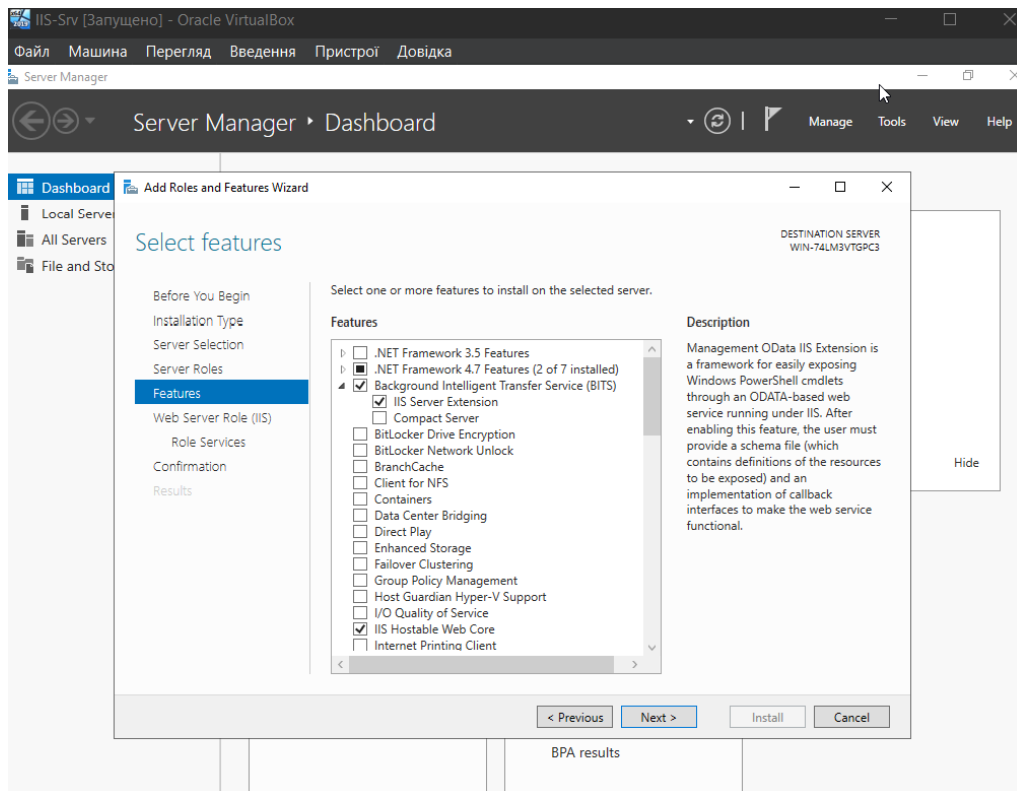


Рисунок 2.13 – Додавання компонентів

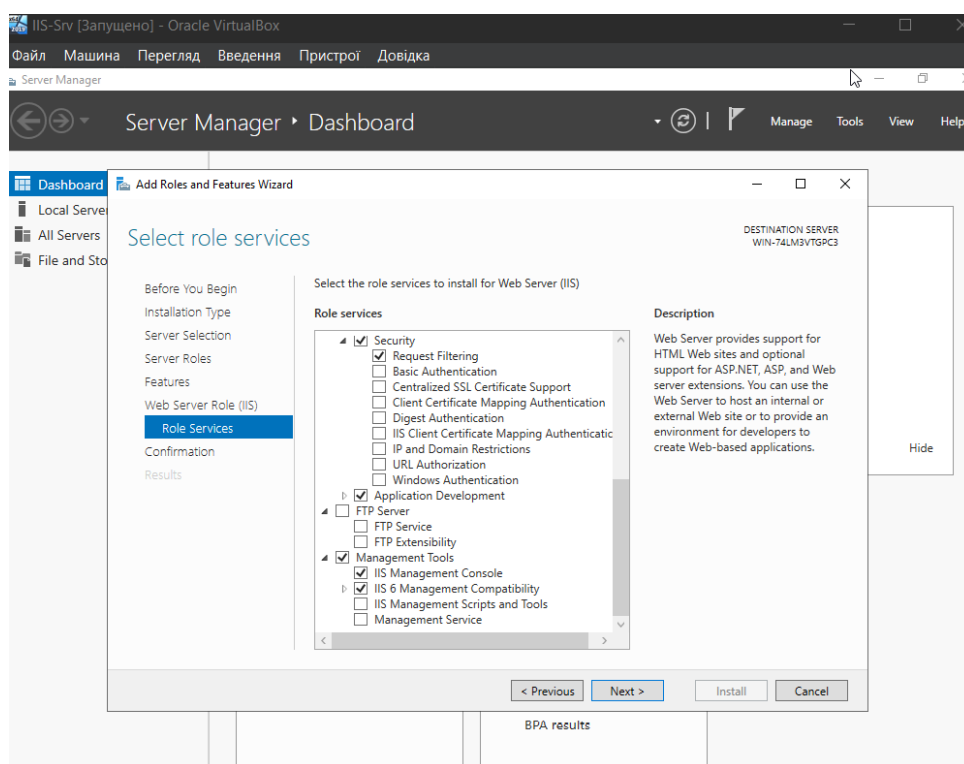


Рисунок 2.14 – Додавання ролі IIS

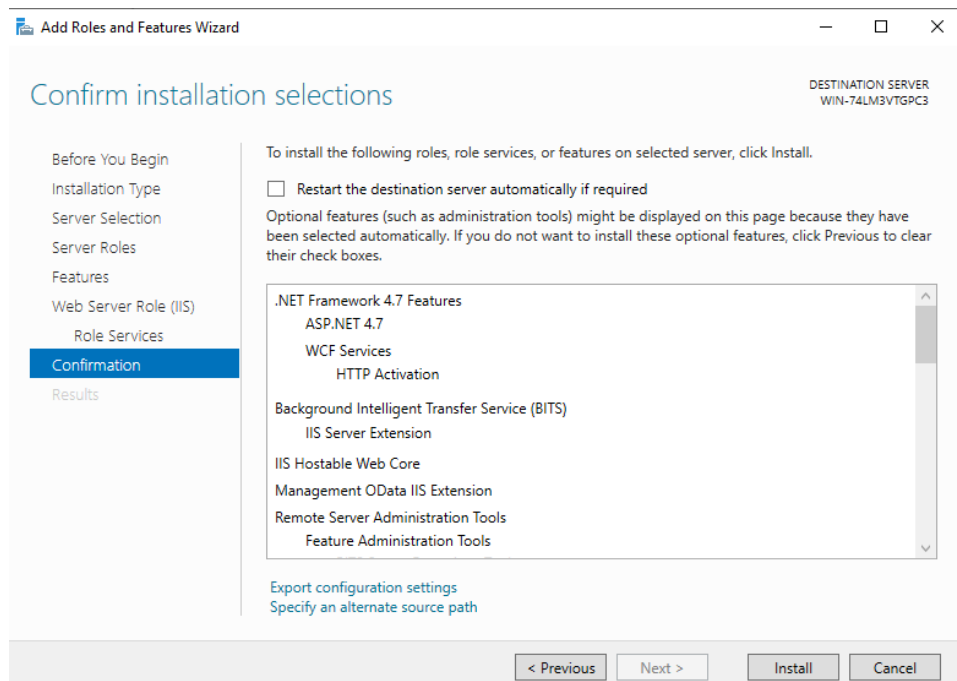


Рисунок 2.15 – Перегляд доданих компонентів

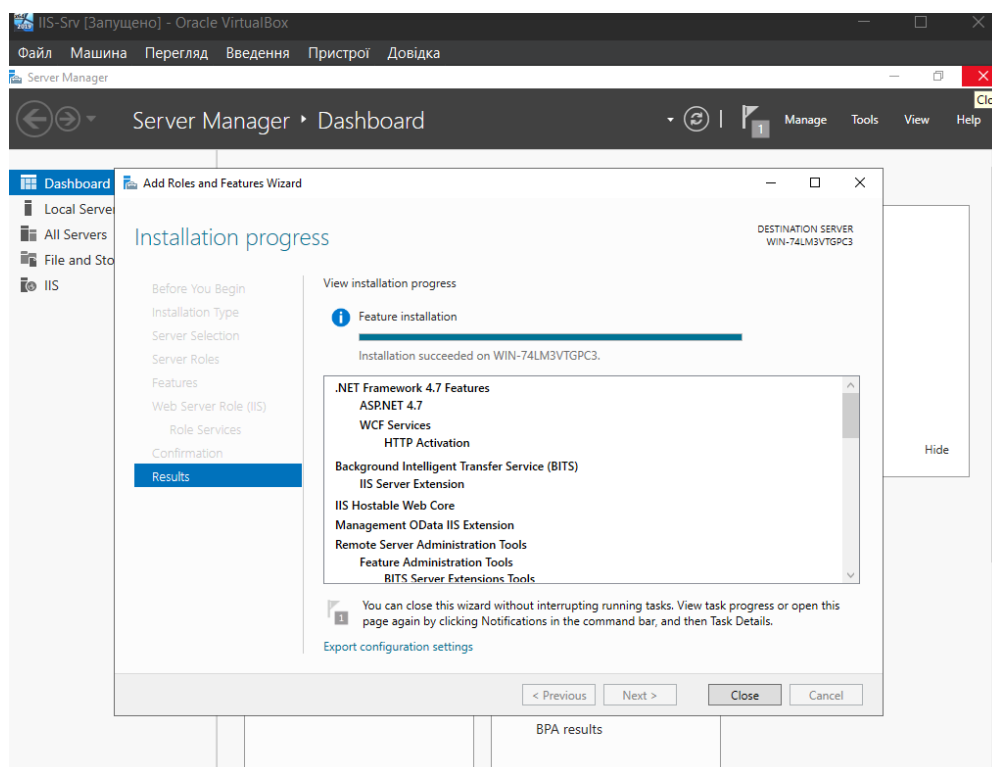


Рисунок 2.16 – Компоненти успішно встановлено

У результаті виконання налаштування ролей Windows Server на сервер було успішно додано роль Internet Information Services (IIS) - вбудованого вебсервера, який дозволяє публікувати вебзастосунки, API та вебсайти. У менеджер IIS видно базову структуру сайту Default Web Site, що свідчить про правильну роботу сервісу.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

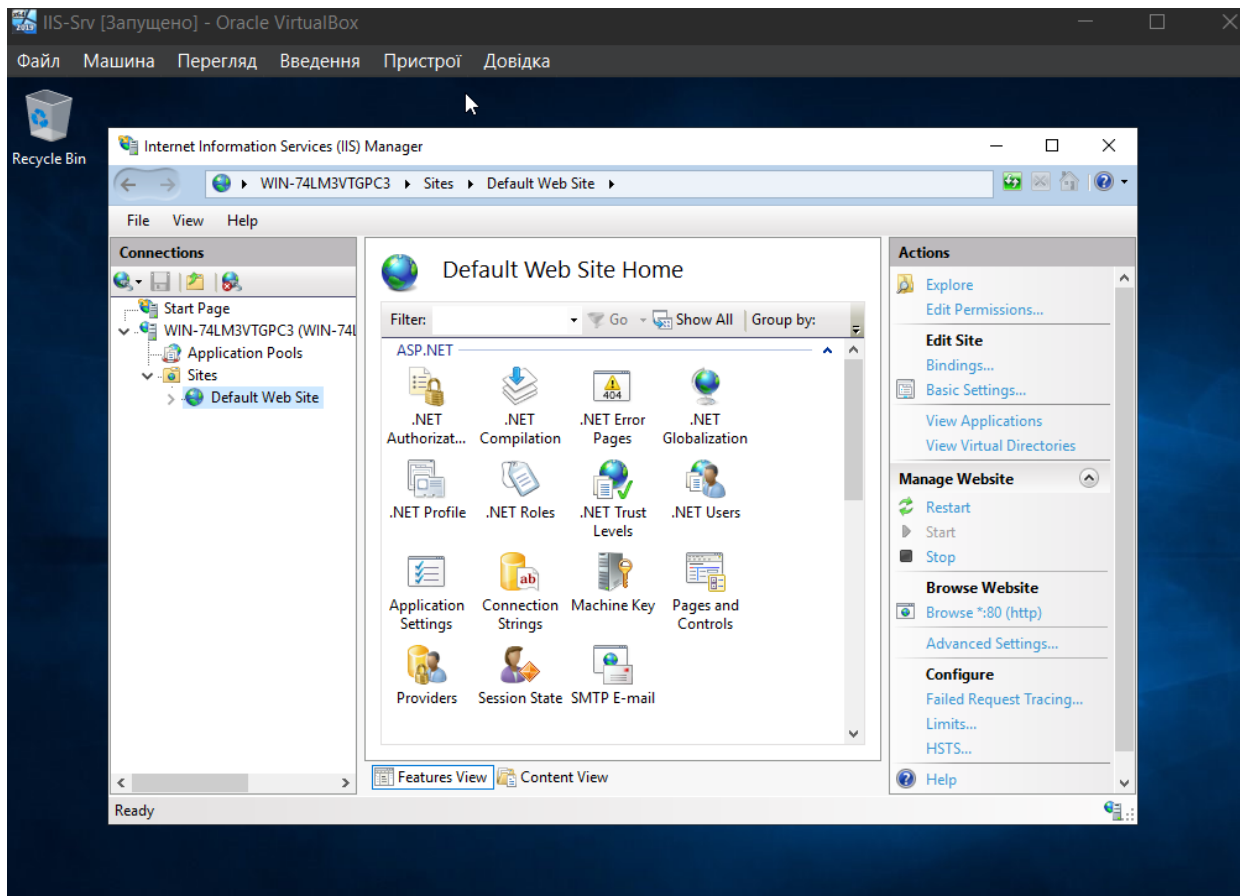


Рисунок 2.17 – Огляд IIS

За допомогою команди Power Shell можна перевірити запущені вебсервери (рис.2.18).

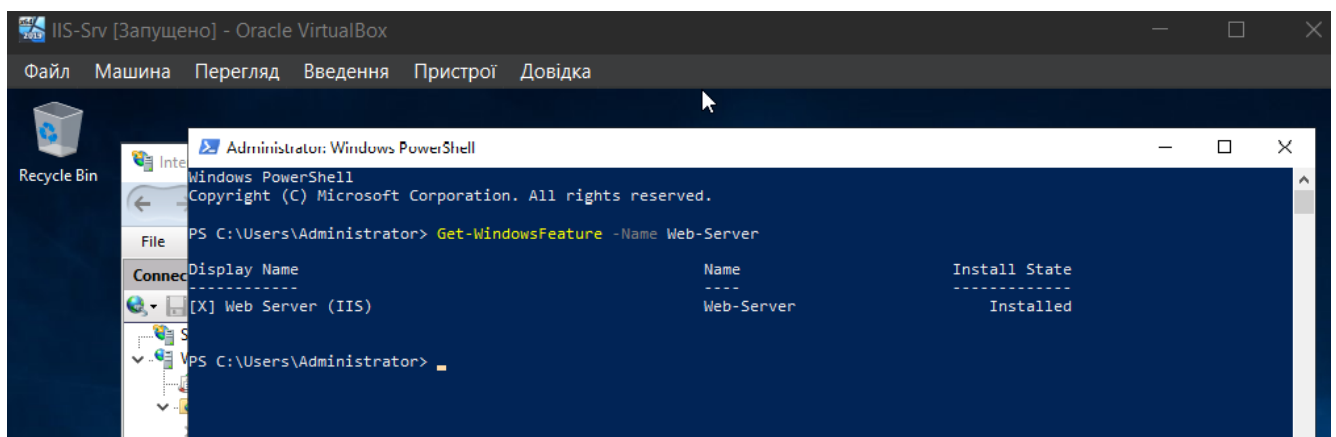


Рисунок 2.18 – Перегляд запущених вебсерверів

2.3 Встановлення Ms SQL Server та створення бази даних

Для забезпечення повноцінної роботи вебсайту необхідно впровадити систему управління базами даних (СУБД), яка дозволяє ефективно зберігати,

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

обробляти та отримувати дані. Найбільш типовим вибором для інтеграції з вебсервером IIS є Microsoft SQL Server. У рамках цього проєкту буде виконано встановлення MS SQL Server 2022 Express - безкоштовної редакції, яка є достатньо функціональною для невеликих і середніх вебзастосунків (рис.2.19 – 2.25). У подальшому буде здійснене базове налаштування сервера та створення необхідної структури бази даних для зберігання інформації про користувачів та інші елементи вебзастосунку.

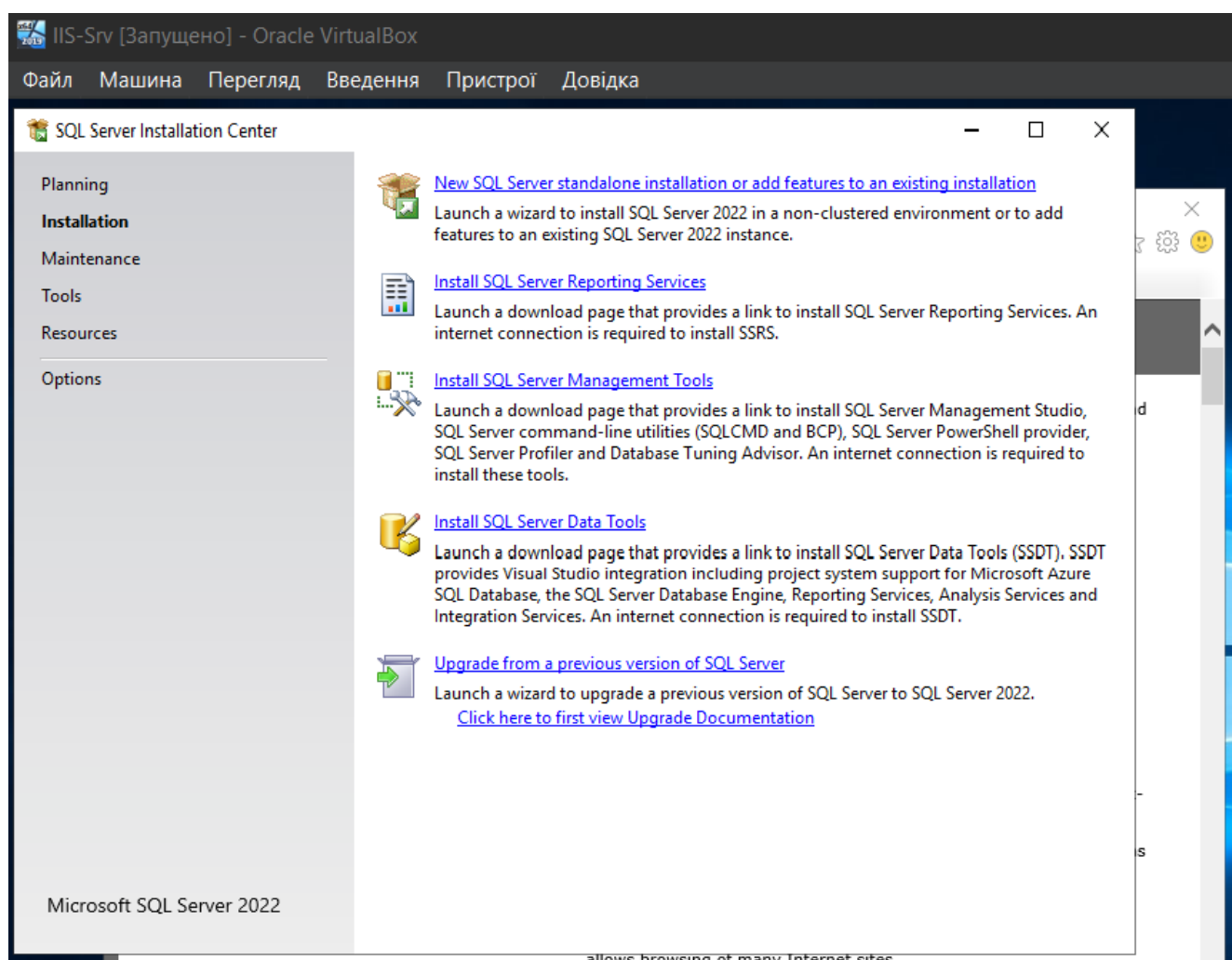


Рисунок 2.19 – Встановлення SQL Server

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

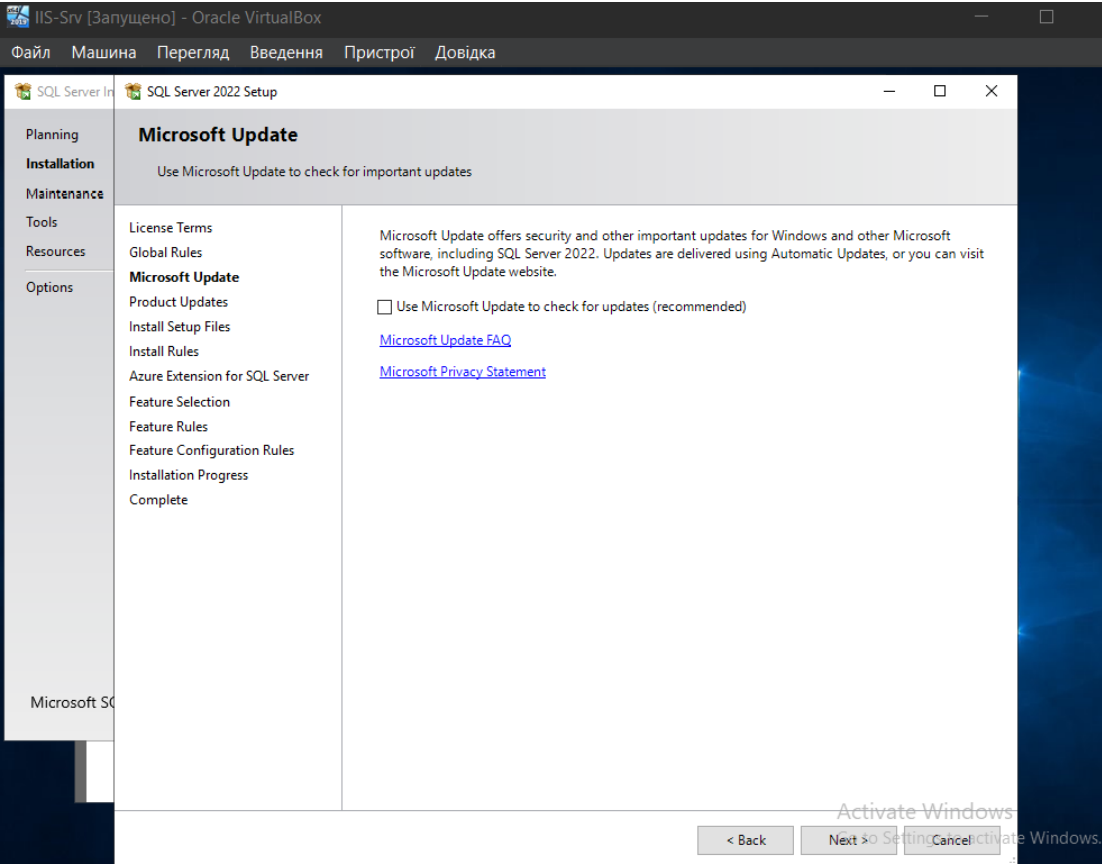


Рисунок 2.20 – Відмовляємося від перевірки оновлень

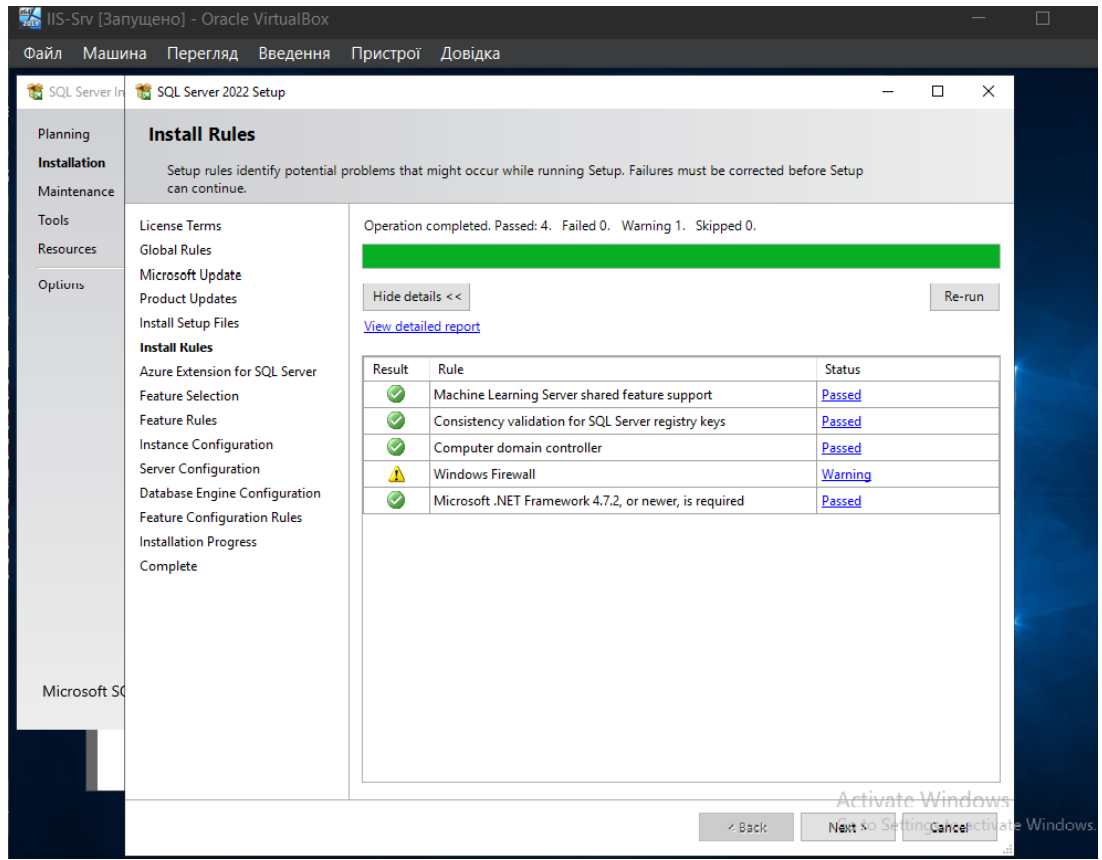


Рисунок 2.21 – Сторінка встановлення правил

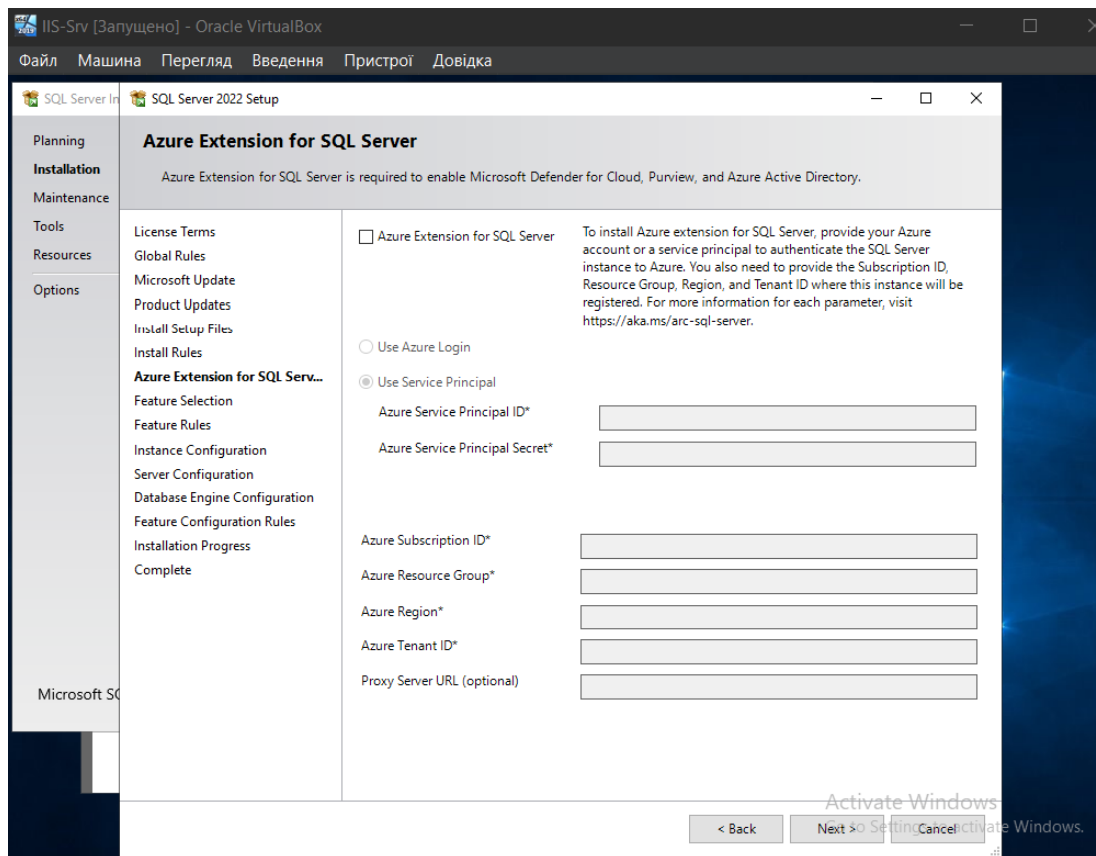


Рисунок 2.22 – Відмовляємося від розширення Azure

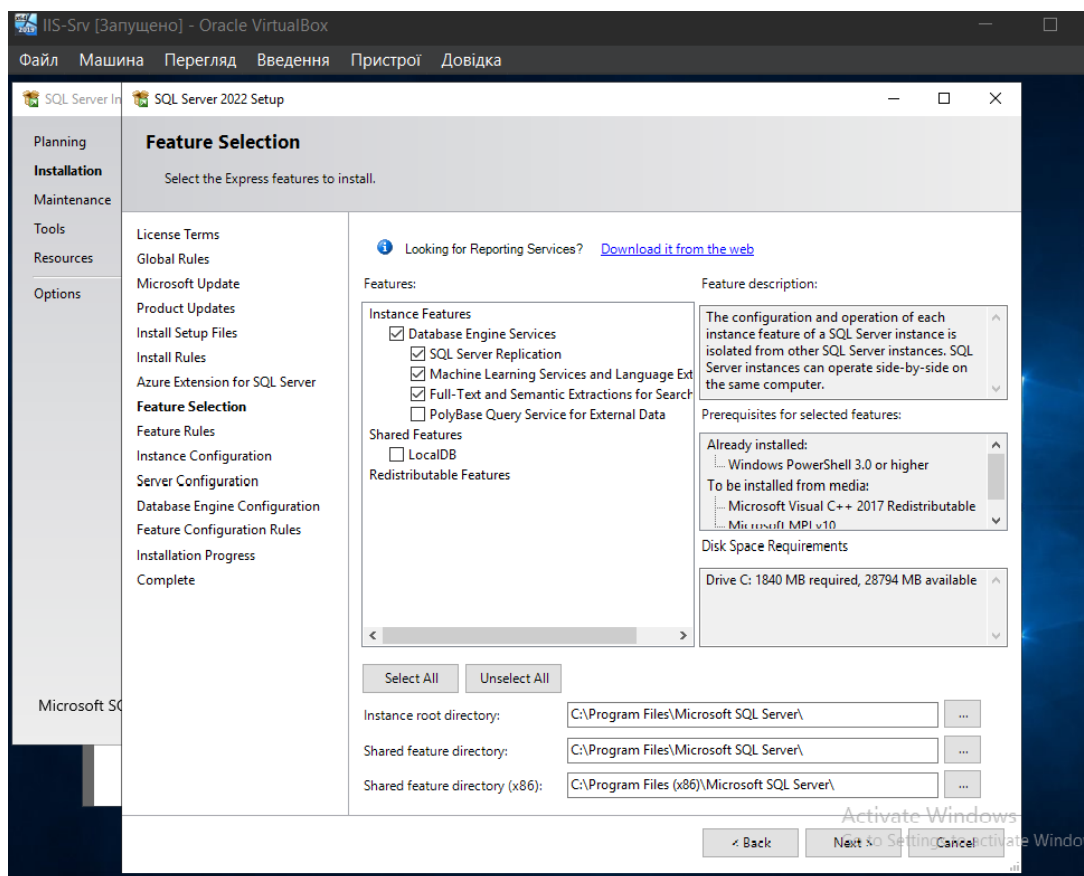


Рисунок 2.23 – Обираємо потрібні компоненти

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

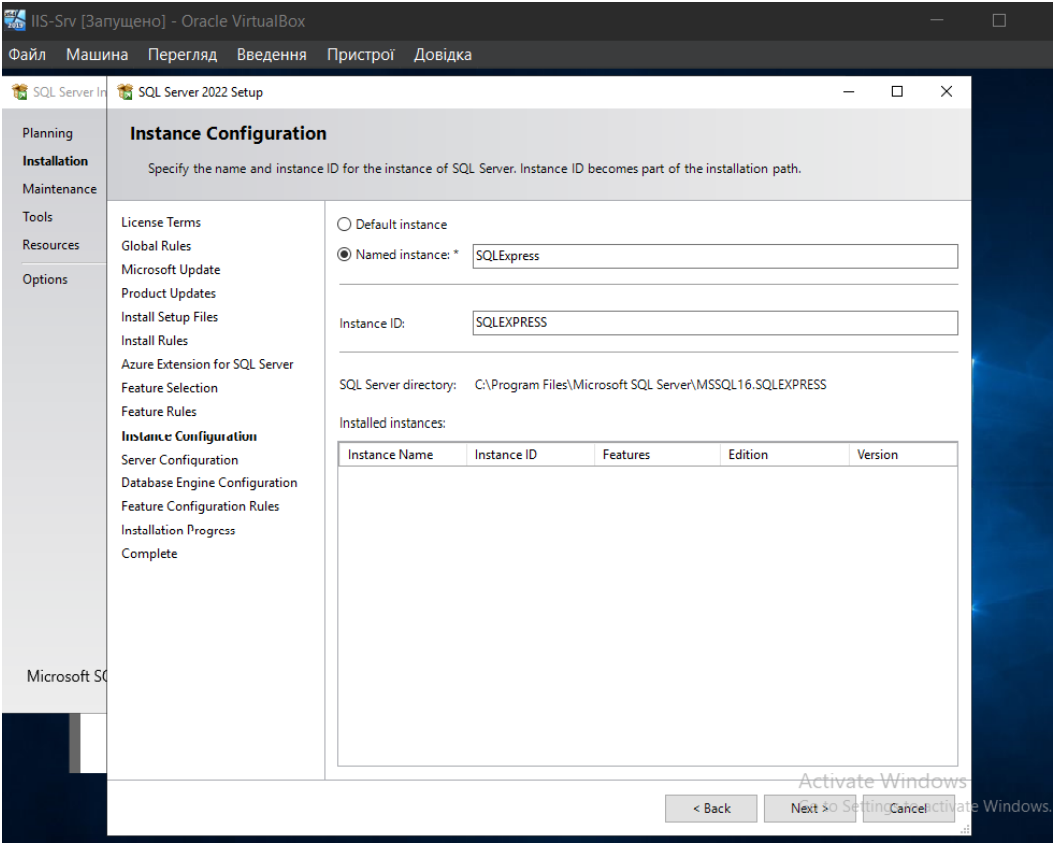


Рисунок 2.24 – Встановлюємо ім'я та ID сервера

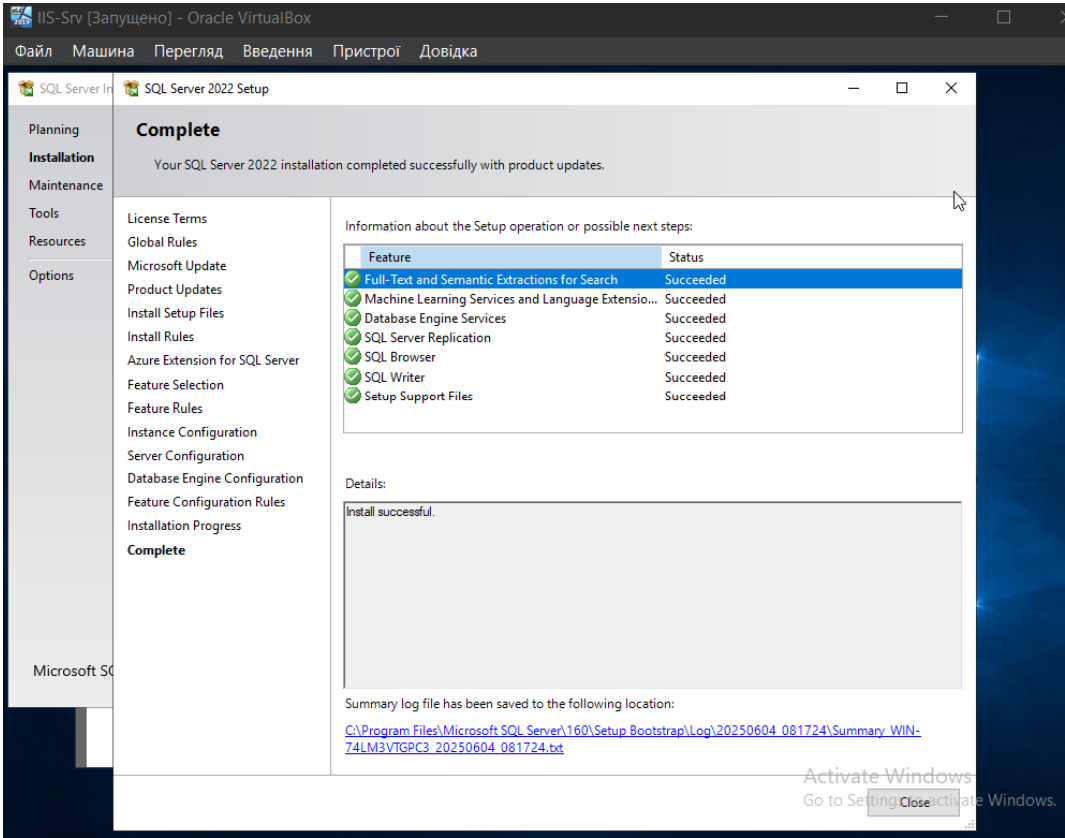


Рисунок 2.25 – SQL Server встановлено

Після встановлення SQL Server необхідно перевірити стан порту 1433 (рис.2.26 – 2.28), оскільки саме він використовується за замовчуванням для обміну даними через протокол TCP/IP між клієнтами та SQL Server. Цей порт є основним каналом зв'язку для підключень до серверу бази даних із зовнішніх програм, зокрема з інших серверів, клієнтських додатків або віддалених адміністраторів.

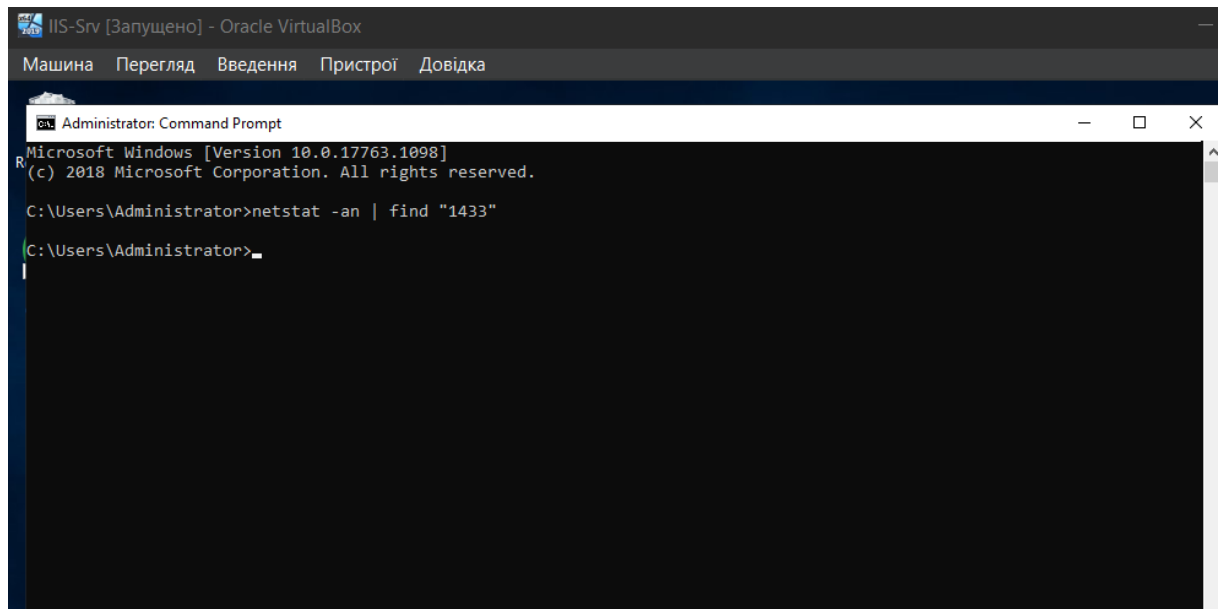


Рисунок 2.26 – Порт 1433 закрито

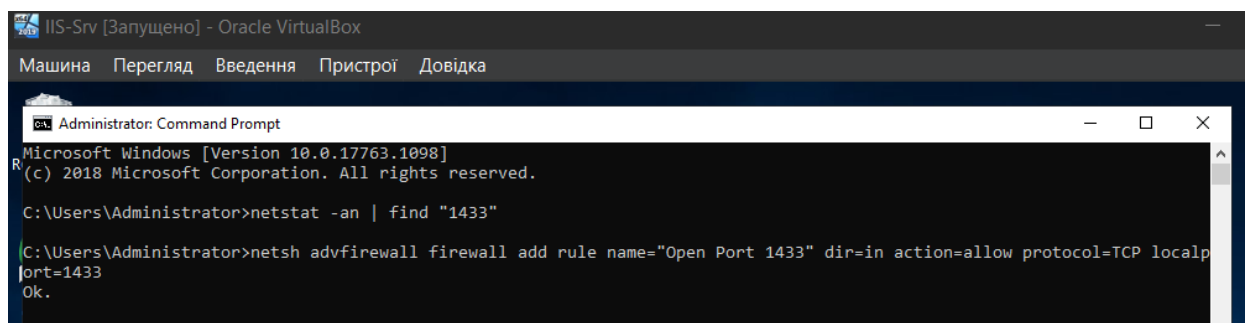


Рисунок 2.27 – Відкриваємо порт

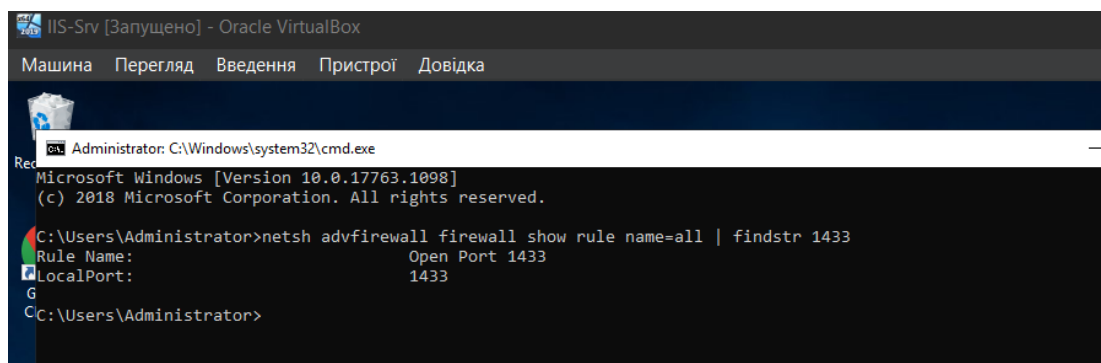


Рисунок 2.28 – Порт відкрито

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

Створення бази даних - Після встановлення SQL Server наступним етапом є створення бази даних, яка слугуватиме сховищем для збереження структурованої інформації. База даних створюється в середовищі SQL Server Management Studio (SSMS). Під час створення бази даних визначається її структура таблиць, типи даних, зв'язки між таблицями та обмеження. Правильне проєктування бази дозволяє уникнути надлишковості даних, покращує продуктивність і спрощує адміністрування.

У Microsoft SQL Server існує кілька основних категорій типів даних, які використовуються для зберігання різноманітної інформації в таблицях бази даних. В даній роботі будуть використовуватися такі типи:

- INT - ціле число
- TINYINT - дуже мале ціле число (0 до 255)
- DECIMAL(p,s) - точне десяткове число з фіксованою точністю
- FLOAT - число з плаваючою комою
- CHAR(n) - рядок фіксованої довжини n
- VARCHAR(n) - рядок змінної довжини до n символів
- VARCHAR(MAX) - довгий текст
- DATE - дата (рік, місяць, день)
- TIME - час (години, хвилини, секунди)
- DATETIME - дата + час
- BIT - логічне значення (0 або 1)
- UNIQUEIDENTIFIER - унікальний ідентифікатор (GUID)

У ході роботи була створена база даних для обліку робочого часу, керування персоналом, обробки запитів на відпустки, а також врахування календарних правил. Вона складається з п'яти основних таблиць. Діаграма цієї бази даних зображена на (рис.2.29).

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

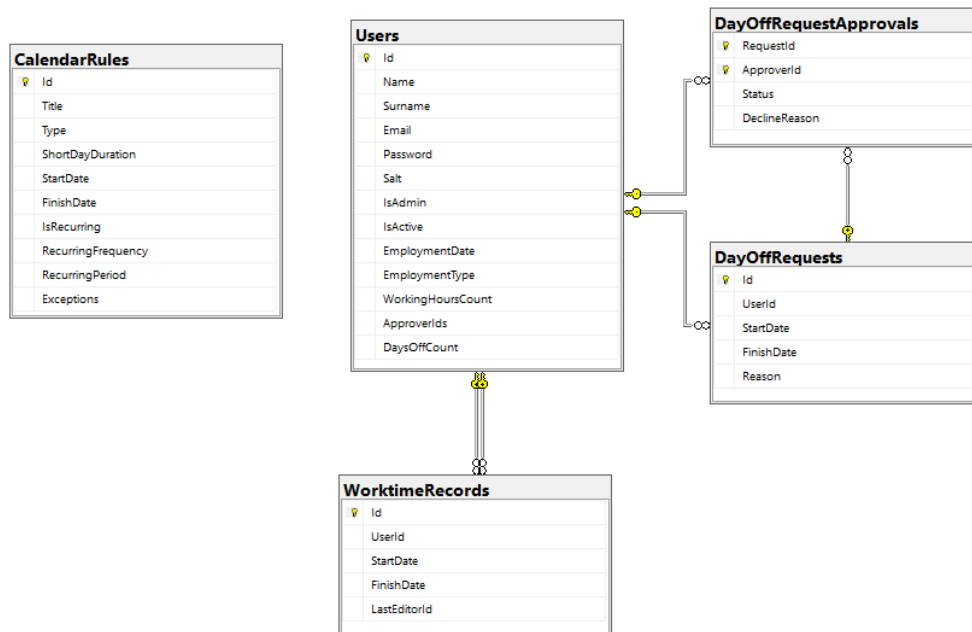


Рисунок 2.29 – Діаграма бази даних

У створеній базі даних таблиця Users (рис.2.30) зберігає основну інформацію про користувачів системи, зокрема ім'я, прізвище, email, пароль, статус адміністратора, дату працевлаштування та інші характеристики. Ця таблиця є центральною і пов'язана з кількома іншими через зовнішні ключі: її поле Id використовується як зовнішній ключ у таблицях, що містять інформацію про робочий час, запити на відпустку та їх погодження.

Окрім основних атрибутів, таблиця також містить поле Salt, яке використовується для безпечного зберігання паролів. Salt є унікальним випадковим рядком символів, що генерується для кожного користувача та додається до паролю перед його хешуванням. Такий підхід гарантує, що навіть за наявності однакових паролів у різних користувачів, збережені в базі хеші будуть відрізнятися. Це значно підвищує безпеку даних та ускладнює використання атак із застосуванням попередньо обчислених хеш-таблиць [25].

	Column Name	Data Type	Allow Nulls
Id		uniqueidentifier	☐
Name		nvarchar(128)	☐
Surname		nvarchar(128)	☐
Email		nvarchar(256)	☐
Password		nvarchar(128)	☒
Salt		nvarchar(128)	☒
IsAdmin		bit	☐
IsActive		bit	☒
EmploymentDate		datetime	☐
EmploymentType		tinyint	☐
WorkingHoursCount		decimal(4, 2)	☒
ApproverIds		nvarchar(MAX)	☐
DaysOffCount		int	☐
			☐

Рисунок 2.30 – Структура таблиці Users

Таблиця WorktimeRecords (рис.2.31) реєструє часові періоди, протягом яких працівники виконували свою роботу. Кожен запис прив'язаний до певного користувача через поле UserId, а також містить додаткове поле LastEditorId, яке використовується для фіксації того, хто вносив останні зміни до запису.

	Column Name	Data Type	Allow Nulls
Id		uniqueidentifier	☐
UserId		uniqueidentifier	☐
StartDate		datetime	☐
FinishDate		datetime	☒
LastEditorId		uniqueidentifier	☒
			☐

Рисунок 2.31 – Структура таблиці WorktimeRecords

DayOffRequests (рис.2.32) використовується для зберігання запитів на відпустку. Вона містить інформацію про дати початку та завершення запитуваної відпустки, а також причину її оформлення. Як і попередня таблиця, вона зв'язана з таблицею користувачів через поле UserId.

	Column Name	Data Type	Allow Nulls
🔑	Id	uniqueidentifier	☐
	UserId	uniqueidentifier	☐
	StartDate	datetime	☐
	FinishDate	datetime	☐
	Reason	tinyint	☐
			☐

Рисунок 2.32 – Структура таблиці DayOffRequests

Для відстеження погодження запитів на відпустку використовується таблиця DayOffRequestApprovals (рис.2.33). Вона містить зовнішні ключі до таблиці запитів (RequestId) та до користувача, який погоджує чи відхиляє запит (ApproverId). Окрім цього, в таблиці зберігається статус розгляду запиту (наприклад, "схвалено" або "відхилено") та, за потреби, причина відмови

	Column Name	Data Type	Allow Nulls
🔑	RequestId	uniqueidentifier	☐
🔑	ApproverId	uniqueidentifier	☐
	Status	tinyint	☒
	DeclineReason	nvarchar(256)	☒
			☐

Рисунок 2.33 – Структура таблиці DayOffRequestApprovals

Окремо виділена таблиця CalendarRules (рис.2.34), яка містить правила щодо календарних днів - наприклад, інформацію про святкові або скорочені дні, дати початку і завершення їх дії, частоту повторення, винятки. Ці дані використовуються для налаштування загального графіка роботи та врахування особливих календарних подій.

	Column Name	Data Type	Allow Nulls
🔑	Id	uniqueidentifier	☐
	Title	nvarchar(256)	☐
	Type	tinyint	☐
	ShortDayDuration	decimal(18, 0)	☒
	StartDate	datetime	☐
	FinishDate	datetime	☒
	IsRecurring	bit	☐
	RecurringFrequency	smallint	☒
	RecurringPeriod	tinyint	☒
	Exceptions	nvarchar(MAX)	☒
			☐

Рисунок 2.34 – Структура таблиці CalendarRules

2.4 Реалізація програмного забезпечення серверної частини

Серверна частина буде реалізована за допомогою мови програмування C# у середовищі ASP.NET Core - потужного фреймворку для створення масштабованих і продуктивних веб-додатків.

У рамках реалізації було створено контролери для обробки HTTP-запитів, моделі даних, що відповідають таблицям бази даних та сервіси для обробки логіки застосунку. Також налаштовано зв'язок із СУБД MS SQL Server за допомогою фреймворку Dapper.

Розробка здійснюється у середовищі Visual Studio - інструменті для розробки додатків під .NET. Visual Studio забезпечує зручне управління проєктами, автоматизацію збірки, інструменти для налагодження, а також підтримку роботи з базами даних та інтеграцію з системами контролю версій.

Для зберігання даних у застосунку використовується MS SQL Server, тому необхідно забезпечити коректне підключення бази даних до нашого проєкту (рис.2.35, 2.36). Для цього в конфігураційний файл (appsettings.json) вноситься рядок підключення (Connection String), який містить необхідні параметри: адресу сервера, назву бази даних, аутентифікаційні дані тощо.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 2.35 – Підключаємо Ms SQL до проєкту

```

{
  "ConnectionStrings": {
    "Database": "Server=WIN-AJT8AC8UL64;Database=TimeTracker;Integrated Security=True;"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Microsoft.Hosting.Lifetime": "Information",
      "Microsoft.AspNetCore.Authentication": "Error",
      "Microsoft.AspNetCore.Authorization": "Error"
    }
  },
  "MailConfiguration": {
    "From": "timetracker41@gmail.com",
    "SmtpServer": "smtp.gmail.com",
    "Port": 465,
    "Username": "timetracker41@gmail.com",
    "Password": "bwikoqhjtdkpddwj"
  }
}

```

Рисунок 2.36 – Створення ConnectionString

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

У подальшому, на основі моделі даних, буде створено необхідну бізнес-логіку для взаємодії з базою: отримання, додавання, редагування та видалення записів. Це забезпечується завдяки використанню фреймворку Dapper, який дозволяє ефективно працювати з SQL-запитами.

Dapper - це об'єктно-реляційний мапер (ORM) для .NET, який був розроблений компанією Stack Overflow (рис.2.37). Його головна мета - забезпечити максимальну продуктивність при роботі з базами даних, зберігаючи при цьому простоту використання.

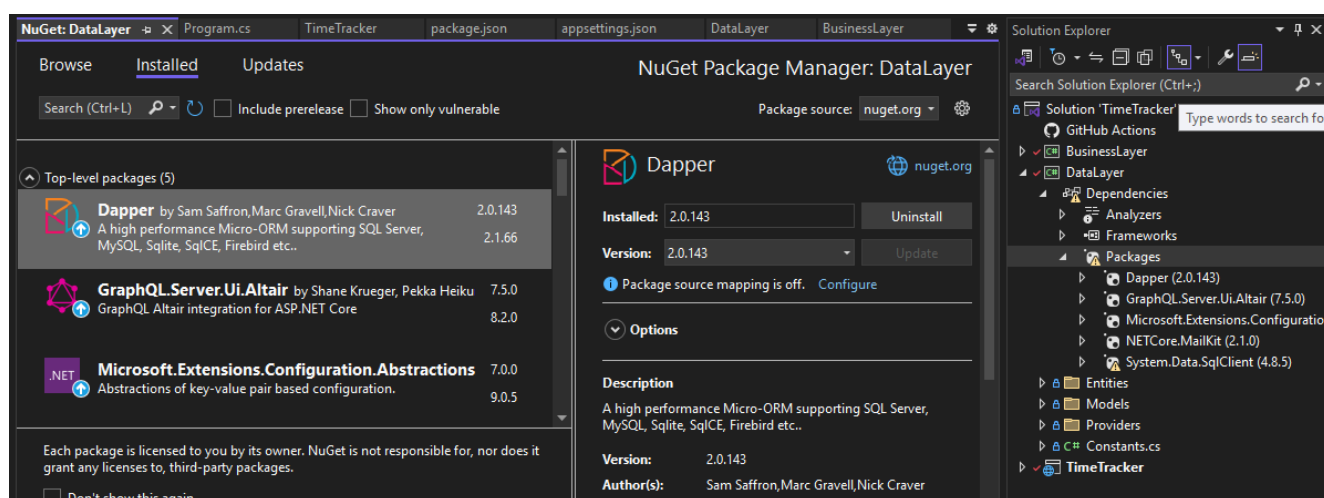


Рисунок 2.37 – Встановлення Dapper (NuGet Package)

На відміну від важчих ORM, таких як Entity Framework, Dapper не приховує SQL-запити від розробника - ви пишете запити вручну, що дає більше контролю над тим, що саме виконується на стороні бази даних. Dapper лише допомагає автоматично перетворювати результати SQL-запитів у об'єкти C#, що значно спрощує написання коду [26].

На цьому етапі розробки серверної частини потрібно створити моделі даних, які будуть відображати структуру таблиць у базі даних. Моделі - це класи на мові C#, які містять властивості, що відповідають стовпцям таблиць бази даних (рис.2.38). Лістинг основних моделей знаходиться в Додатку Д.

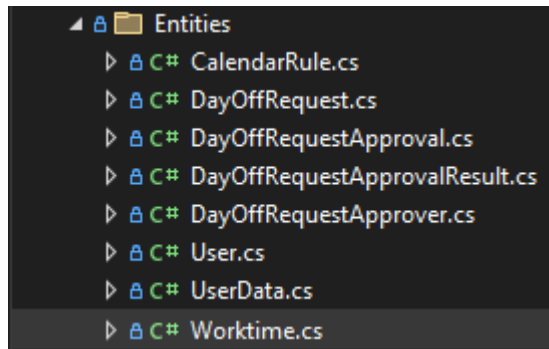


Рисунок 2.38 – Entities

Наступним кроком після створення моделей є створення інтерфейсів, які визначають контракти для доступу до даних. Один із таких інтерфейсів - це `IWorktimeProvider`. Інтерфейс описує набір методів, які будуть реалізовані у класі, що відповідає за роботу з таблицею `Worktime`.

Це дозволяє розділити логіку, відокремивши визначення функціональності від її реалізації, що сприяє кращій структуризації коду. Завдяки використанню інтерфейсів забезпечується гнучкість у зміні або заміні реалізації без необхідності втручання у код, який залежить від цієї функціональності. Крім того, інтерфейси значно спрощують процес тестування, оскільки дають змогу створювати підроблені реалізації для юніт-тестів.

Лістинг класу `IWorktimeProvider`:

```
public interface IWorktimeProvider
{
    int CreateWorktimeRecord(Worktime worktime);

    Worktime? GetWorktimeRecordById(Guid id);

    Worktime? GetUnfinishedWorktimeRecordByUserId(string userId);

    IEnumerable<Worktime> GetWorktimeRecords(Sorting? sorting, WorktimeFilter? filter,
    Paging? paging);

    Worktime UpdateFinishDate(DateTime finishDate, string userId);

    int UpdateWorktimeRecord(Worktime worktimeRecord);

    int GetRecordCount(WorktimeFilter? filter);

    public void CreateWorktimeRecords(IEnumerable<Worktime> worktimeRecords);
}
```

Інтерфейс `IWorktimeProvider` визначає набір методів, що забезпечують основні операції над записами робочого часу. До них належать методи для створення

одного або декількох записів, отримання записів за ідентифікатором або відповідно до певних умов, оновлення існуючих даних, а також метод для підрахунку загальної кількості записів у базі даних.

Після створення інтерфейсу `IWorktimeProvider`, наступним етапом є реалізація цього інтерфейсу у класі `WorktimeProvider`. Саме тут реалізується логіка взаємодії з базою даних - виконуються SQL-запити для створення, читання, оновлення та обробки записів таблиці `WorktimeRecords`.

Лістинг класу `WorktimeProvider`:

```
public class WorktimeProvider : Provider, IWorktimeProvider
{
    public WorktimeProvider(IConfiguration configuration) : base(configuration) { }

    public int CreateWorktimeRecord(Worktime worktime)
        => Execute(Queries.Worktime.CreateWorktimeRecord, worktime);

    public Worktime? GetWorktimeRecordById(Guid id)
        => Query<Worktime?>(Queries.Worktime.GetWorktimeRecordById, new { Id = id
    }).FirstOrDefault();

    public Worktime? GetUnfinishedWorktimeRecordByUserId(string userId)
        => Query<Worktime>(Queries.Worktime.GetUnfinishedWorktimeRecordByUserId, new
    {UserId = userId}).FirstOrDefault();

    public IEnumerable<Worktime> GetWorktimeRecords(Sorting? sorting, WorktimeFilter?
    filter, Paging? paging)
        => Query<Worktime>(Queries.Worktime.GetWorktimeRecords(sorting, filter, paging));

    public Worktime UpdateFinishDate(DateTime finishDate, string userId)
        => Query<Worktime>(Queries.Worktime.UpdateFinishDate, new { finishDate, UserId =
    userId }).First();

    public int UpdateWorktimeRecord(Worktime worktimeRecord)
        => Execute(Queries.Worktime.UpdateWorktimeRecord, worktimeRecord);

    public int GetRecordCount(WorktimeFilter? filter)
        => Query<int>(Queries.Worktime.GetRecordsCount(filter)).First();

    public void CreateWorktimeRecords(IEnumerable<Worktime> worktimeRecords)
        => Execute(Queries.Worktime.CreateWorktimeRecords(worktimeRecords));
}
```

`WorktimeProvider` наслідує базовий клас `Provider`, що, ймовірно, містить методи для виконання SQL-запитів, такі як `Execute()` та `Query<T>()`. Це дозволяє значно спростити взаємодію з базою даних. Кожен метод використовує SQL-запит, визначений у класі `Queries.cs`, і передає відповідні параметри.

У класі `WorktimeProvider` реалізовані методи, які відповідають за основні дії з даними про робочий час. Вони дозволяють створювати нові записи, отримувати

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

їх за різними критеріями, оновлювати вже існуючі дані, зокрема завершальні дати, а також підраховувати загальну кількість записів у базі даних.

Наступним етапом у реалізації серверної частини є винесення всієї SQL-логіки в окремий статичний клас `QueriesWorktime.cs`. Такий підхід полегшує супровід коду, дозволяє уникнути дублювання та чітко розмежовує бізнес-логіку від безпосередньої роботи з базою даних. У цьому класі запити описані у вигляді константних рядків або методів, які формують SQL динамічно - залежно від параметрів сортування, фільтрації чи пагінації. Це дозволяє реалізовувати складні запити з умовною логікою побудови. Лістинг класу `QueriesWorktime.cs` знаходиться в Додатку Д

Подібна структура забезпечує більшу гнучкість у розробці, спрощує масштабування проєкту та підтримання чистої архітектури. Цей етап реалізовано для усіх інших провайдерів за аналогією з `WorktimeProvider`, буде розширено SQL-запити додатковими умовами (наприклад, фільтрацією за періодами або працівниками)

Також було реалізовано механізм фільтрації користувачів, що дозволяє більш гнучко взаємодіяти з даними та відображати лише релевантну інформацію. Для цього створено клас `UserFilter`, який інкапсулює набір параметрів, за якими може здійснюватися фільтрація під час виконання SQL-запитів.

Лістинг класу `UserFilter.cs`:

```
public class UserFilter
{
    public string SearchText { get; set; }

    public DateTime? StartEmploymentDate { get; set; }

    public DateTime? EndEmploymentDate { get; set; }

    public IEnumerable<EmploymentType> EmploymentTypes { get; set; }

    public bool ShowOnlyActiveUsers { get; set; }

    public UserFilter(string searchText = "", DateTime? startEmploymentDate = null,
DateTime? endEmploymentDate = null, IEnumerable<EmploymentType>? employmentTypes = null,
bool showOnlyActiveUsers = true)
    {
        SearchText = searchText;
        StartEmploymentDate = startEmploymentDate;
        EndEmploymentDate = endEmploymentDate;
        EmploymentTypes = employmentTypes ?? new List<EmploymentType>();
        ShowOnlyActiveUsers = showOnlyActiveUsers;
    }
}
```

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Поле SearchText використовується для пошуку за загальним текстовим запитом - зазвичай це ім'я, прізвище або електронна адреса користувача. Поля StartEmploymentDate і EndEmploymentDate дозволяють звужити вибірку до певного періоду працевлаштування.

EmploymentTypes - це перелік типів зайнятості, що дозволяє, наприклад, знайти лише співробітників з повною або частковою зайнятістю. Якщо перелік порожній, такий фільтр не застосовується.

Крім того, завдяки параметру ShowOnlyActiveUsers можна легко відфільтрувати лише тих користувачів, які наразі є активними у системі, і не показувати неактуальні записи.

Конструктор класу передбачає значення за замовчуванням для кожного з параметрів, що спрощує використання фільтра без зайвого коду. Такий підхід забезпечує зручність, чисту архітектуру та можливість повторного використання логіки фільтрації у різних частинах застосунку.

GraphQL - це сучасний підхід до побудови API, який дозволяє клієнту самому визначати, які саме дані він хоче отримати від сервера. На відміну від REST, де кожен запит повертає заздалегідь визначену структуру, GraphQL дозволяє клієнту запитувати лише ті поля, які йому дійсно потрібні, що знижує обсяг передаваних даних і підвищує гнучкість [27].

У цьому проєкті GraphQL використовується для взаємодії з користувачами системи. Вся логіка запитів (Queries) та змін (Mutations) описується через окремі класи, де визначено, які поля доступні, які аргументи можна передати, та що буде результатом.

Лістинг класу UsersQuery.cs:

```
public UsersQuery(IUserProvider userProvider)
{
    Field<ListGraphType<UserType>>("list")
        .Description("Get list of users")
        .Argument<FilterInputType>("filter", "Filter type")
        .Argument<SortInputType>("sorting", "Sorting type")
        .Argument<PaginationInputType>("pagination", "Pagination type")
        .Resolve(context =>
        {
            UserFilter? filter = context.GetArgument<UserFilter?>("filter");
            Sorting? sorting = context.GetArgument<Sorting?>("sorting");
            Paging? pagination = context.GetArgument<Paging?>("pagination");
```

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

```

        return userProvider.GetAllUsers(filter, sorting, pagination).ToList();
    });

    Field<UserType>("user")
        .Description("Get user by Id")
        .Argument<StringGraphType>("id", "User ID")
        .Resolve(context =>
        {
            string? id = context.GetArgument<string?>("id");

            if (id == null || !Guid.TryParse(id, out _))
                return null;

            return userProvider.GetById(id);
        });

    Field<IntGraphType>("totalUsersCount")
        .Description("Get number of users")
        .Argument<FilterInputType>("filter", "Filter type")
        .Resolve(context =>
        {
            UserFilter? filter = context.GetArgument<UserFilter?>("filter");
            return userProvider.GetTotalUsersCount(filter);
        });
    }
}

```

У класі UsersQuery реалізовано логіку отримання даних про користувачів. Зокрема, передбачено можливість отримувати список користувачів із підтримкою фільтрації за заданими критеріями, сортуванням та розбиттям на сторінки. Крім того, реалізовано функціональність отримання окремого користувача за його ідентифікатором, а також підрахунку кількості користувачів, які відповідають певним умовам.

Таким чином, клієнт (frontend-застосунок) сам вирішує, які поля потрібні, і може гнучко керувати відображенням даних без зміни бекенду.

Клас UsersMutation відповідає за зміну даних: створення та оновлення користувачів. Тут відбувається передача складних об'єктів (CreateUserInputType, UpdateUserInputType) та використання сервісів, таких як IAuthenticationService для генерації паролів або IDaysOffProvider для оновлення затверджень днів відпустки. Лістинг класу UsersMutation.cs знаходиться у Додатку Д.

Цей підхід дозволяє об'єднати бізнес-логіку, валідацію і структуру запиту в одному місці, роблячи API централізованим, зручним для розширення і легко керованим. GraphQL добре підходить у випадках, коли дані складні, мають багато взаємозв'язків або коли потрібна висока гнучкість у роботі з ними.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Висновки до другого розділу

У цьому розділі було послідовно описано процес встановлення та налаштування ключових компонентів серверної інфраструктури, необхідних для розгортання вебзастосунку.

Було успішно встановлено операційну систему Windows Server 2019 у віртуальному середовищі, що забезпечило стабільну та ізольовану платформу для подальшої роботи. Наступним кроком було додано необхідні ролі, зокрема вебсервера IIS, що дозволило реалізувати функціональність, необхідну для обслуговування вебзапитів.

Для організації зберігання даних було встановлено MS SQL Server 2022 Express та створено структуру бази даних із таблицею користувачів, яка спроектована з урахуванням безпеки та ефективності збереження інформації.

Серверна частина застосунку реалізована за допомогою ASP.NET Core із використанням C# та фреймворку Dapper для роботи з базою даних. Впроваджені підходи та налаштування забезпечують надійну, масштабовану та безпечну основу для подальшої розробки і експлуатації вебінфраструктури.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3 РОЗГОРТАННЯ САЙТУ НА ВЕБСЕРВЕРІ, НАЛАШТУ- ВАННЯ МОНІТОРИНГУ ТА БАЛАНСУВАННЯ НАВАНТАЖЕННЯ

3.1 Встановлення та налаштування Application Request Routing (ARR)

Для реалізації функції балансування навантаження буде використано технологію Application Request Routing (ARR) - розширення для веб-сервера IIS, що дозволяє перенаправляти вхідні HTTP/HTTPS-запити на інші сервери на основі заданих правил. ARR виконує роль зворотного проксі та забезпечує розподіл запитів між кількома екземплярами веб-додатка, що сприяє підвищенню продуктивності та відмовостійкості системи.

Буде розглянуто процес встановлення ARR на сервер з IIS, створення server farm (групи серверів), налаштування маршрутів для обробки запитів, а також моніторинг стану вузлів. Такий підхід дозволяє зменшити навантаження на окремі сервери, забезпечити масштабованість та ефективне використання ресурсів при зростанні кількості користувачів.

Також ARR тісно інтегрується з IIS URL Rewrite Module, що дає змогу створювати складні сценарії перенаправлення та фільтрації запитів. Завдяки підтримці зворотного проксі ARR дозволяє приховати внутрішню архітектуру системи, що підвищує безпеку загального рішення. Усе це робить ARR ефективним і зручним засобом для побудови сучасного, надійного та масштабованого серверного середовища. Крім того, ARR можна безкоштовно завантажити з офіційного сайту Microsoft, що забезпечує зручний доступ до цього інструменту та гарантує його актуальність і підтримку (рис.3.1).

Після успішного встановлення в IIS з'являються додаткові можливості (рис.3.2), зокрема функції Application Request Routing (ARR) та Server Farms, які дозволяють налаштовувати зворотне проксі, балансування навантаження та керування трафіком між серверами.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

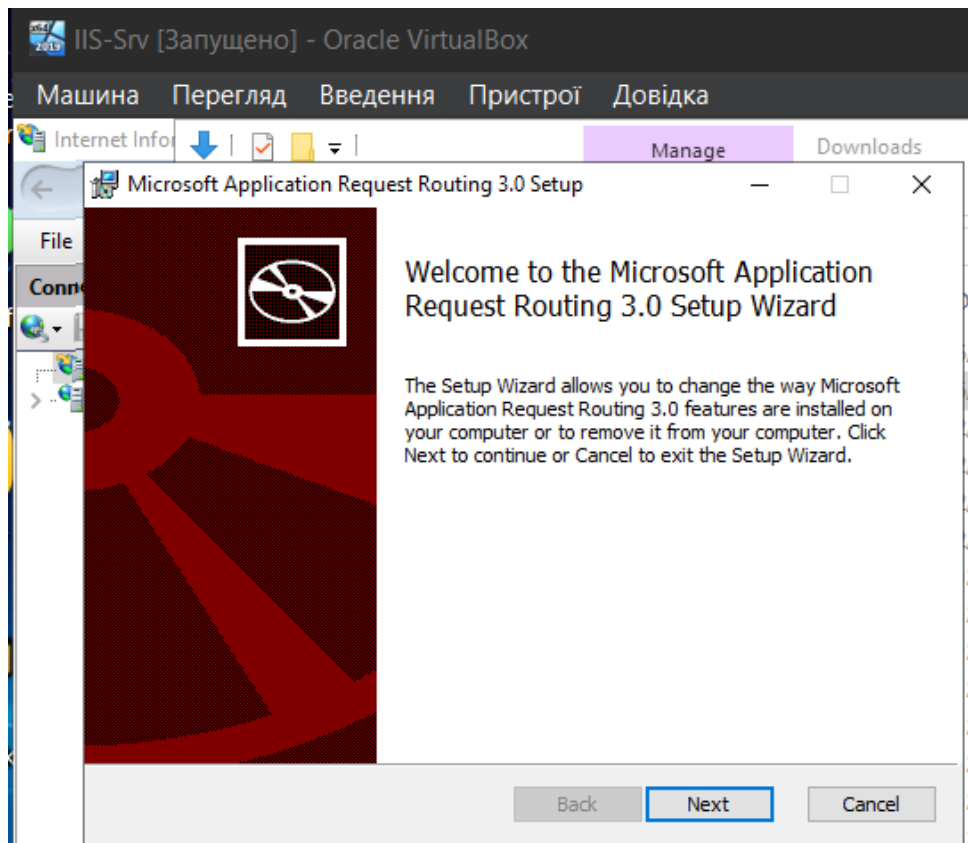


Рисунок – 3.1 – Встановлення ARR

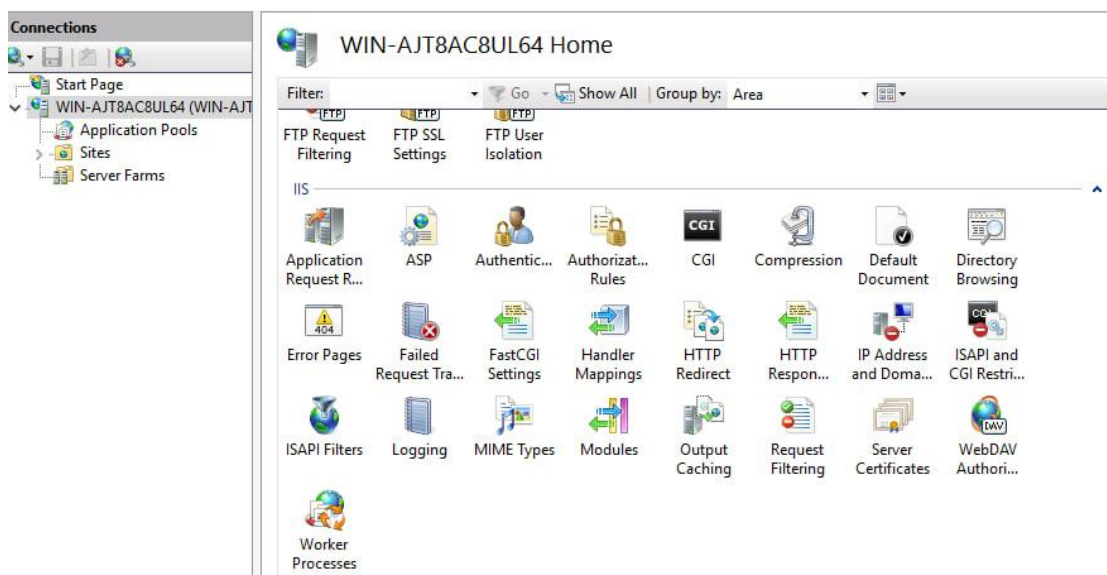


Рисунок 3.2 – Додаткові можливості після встановлення ARR

Після активації ARR в IIS наступним кроком є створення Server Farm - логічного об'єднання кількох серверів, яке використовується для балансування навантаження, підвищення доступності та масштабованості веб-застосунку.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

На цьому етапі ми створюємо нову Server Farm, даємо їй ім'я та додаємо до неї два сервери (рис.3.3 – 3.6). Це можуть бути локальні або віддалені сервери, які будуть обробляти HTTP-запити. Кожен із серверів отримує свою адресу і ARR автоматично розподіляє запити між ними відповідно до вибраної політики балансування (наприклад, за замовчуванням - рівномірно).

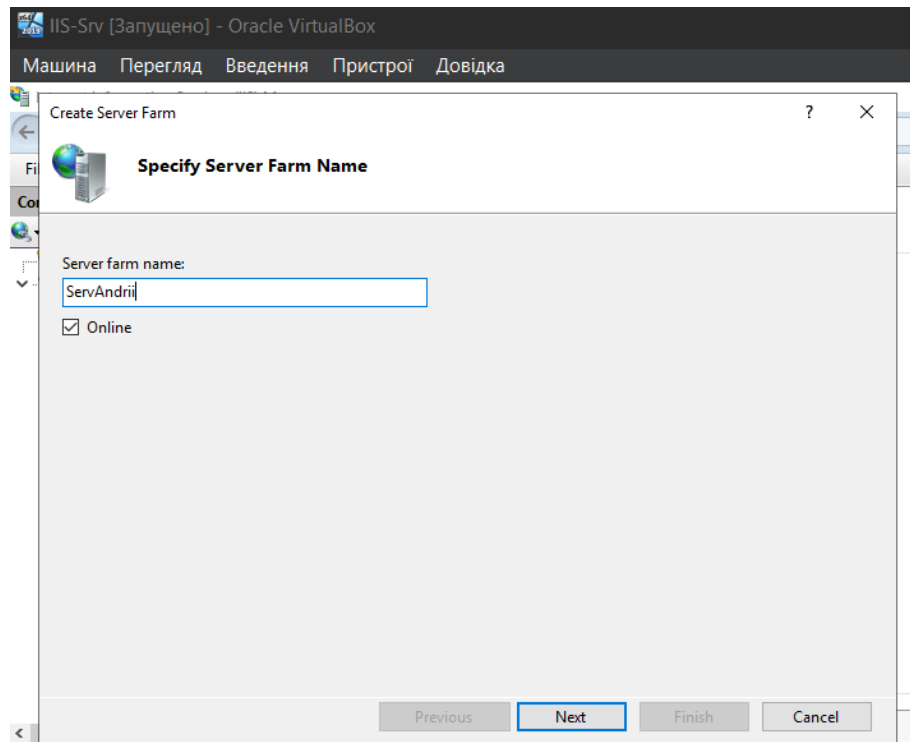


Рисунок 3.3 – Створення Server Farm

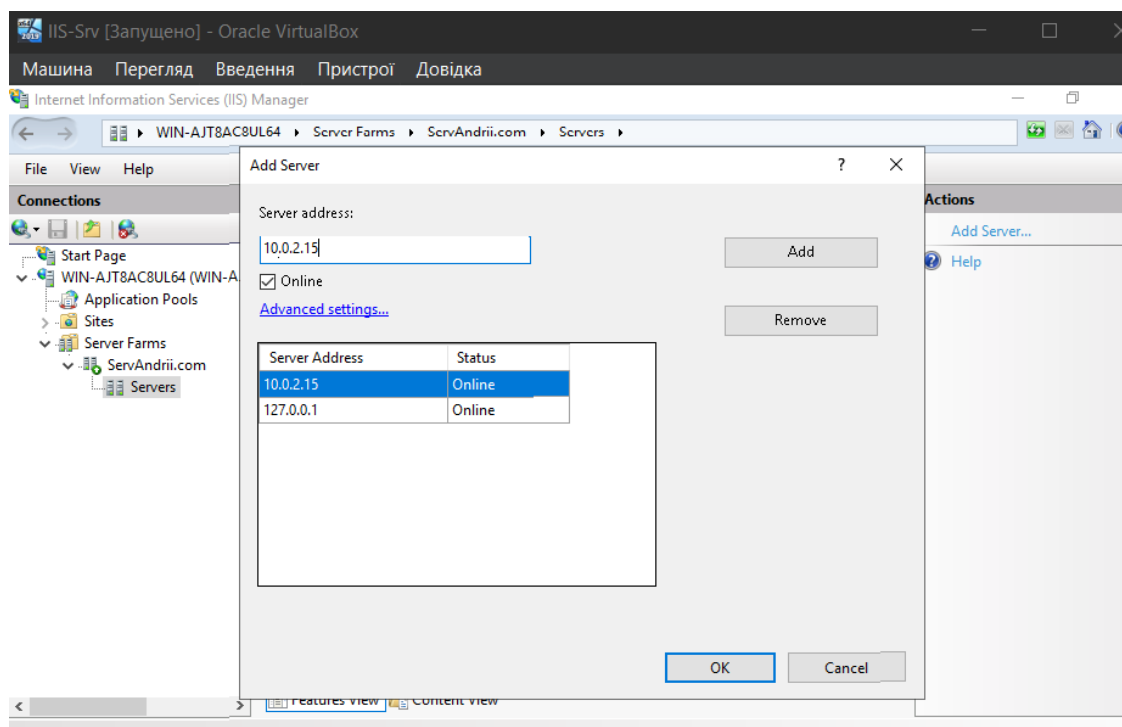


Рисунок 3.4 – Додаємо два сервери

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

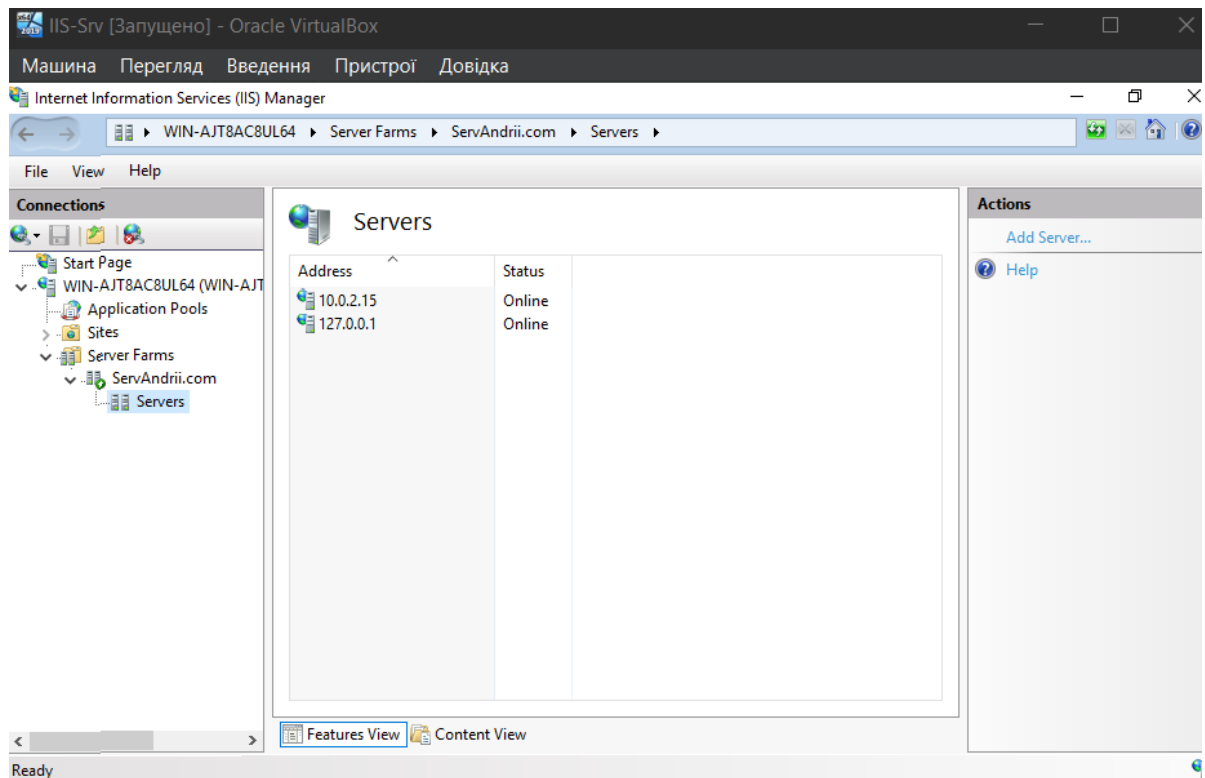


Рисунок 3.5 – Два сервера успішно додано, статус Online

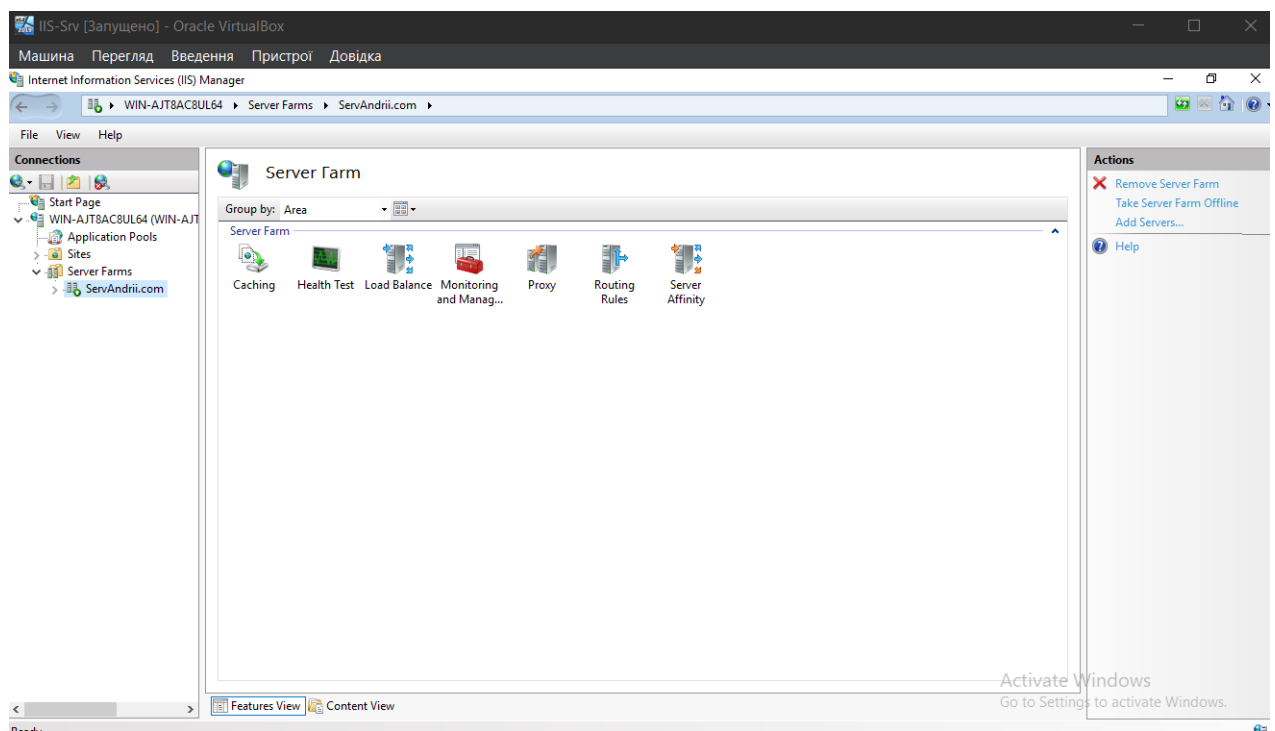


Рисунок 3.6 – Інструменти Server Farm

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

Тепер можна оглянути усі можливості Application Request Routing (ARR), доступні після створення Server Farm у IIS. На зображенні видно інтерфейс управління фермою серверів та набір функцій, які дозволяють гнучко налаштовувати маршрутизацію, кешування, проксіювання та балансування навантаження:

Caching - Дозволяє налаштувати кешування контенту, щоб зменшити навантаження на бекенд-сервери. Часто використовувані відповіді зберігаються на проксі-сервері та повторно використовуються без запиту до джерела.

Health Test - Надає можливість налаштувати регулярні перевірки стану серверів. Якщо якийсь сервер не відповідає, ARR автоматично виключає його з розподілу запитів, забезпечуючи надійність.

Load Balance - Містить параметри балансування навантаження між серверами. Можна вибрати метод розподілу (наприклад, рівномірно або за навантаженням) для забезпечення оптимальної продуктивності.

Monitoring and Management - Надає засоби моніторингу роботи Server Farm та керування станом серверів. Дає змогу переглядати статистику запитів, відповідей і навантаження.

Proxy - Активує функцію зворотного проксі - усі запити проходять через ARR, що дозволяє приховати реальні IP-адреси внутрішніх серверів і додатково захищає інфраструктуру.

Routing Rules - Дає змогу створювати гнучкі правила маршрутизації. Наприклад, можна спрямовувати запити на певний сервер залежно від URL, типу запиту або заголовків.

Server Affinity - зберігає сесію користувача з одним і тим самим сервером протягом усього сеансу. Це важливо для станозалежних застосунків, які зберігають інформацію в оперативній пам'яті.

URL Rewrite - це розширення для IIS (Internet Information Services), яке дозволяє керувати перенаправленням запитів на сервері. Воно дає змогу створювати гнучкі правила для зміни URL-адрес, як на вхідному, так і на вихідному трафіку (рис.3.7 – 3.9). Ми встановимо це розширення з офіційного сайту Microsoft [\[29\]](#).

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

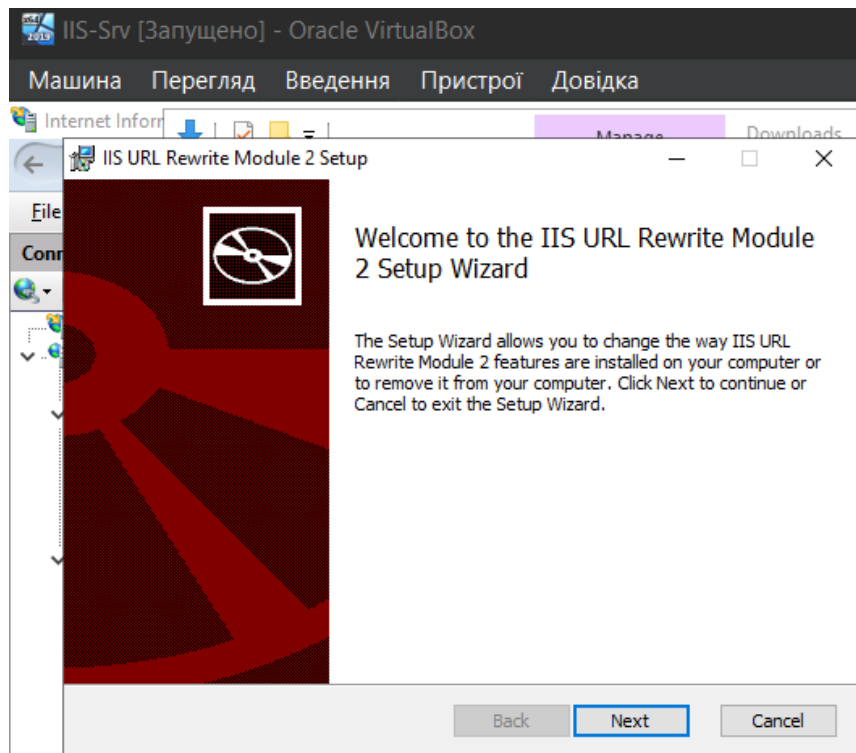


Рисунок 3.7 – Встановлення URL Rewrite

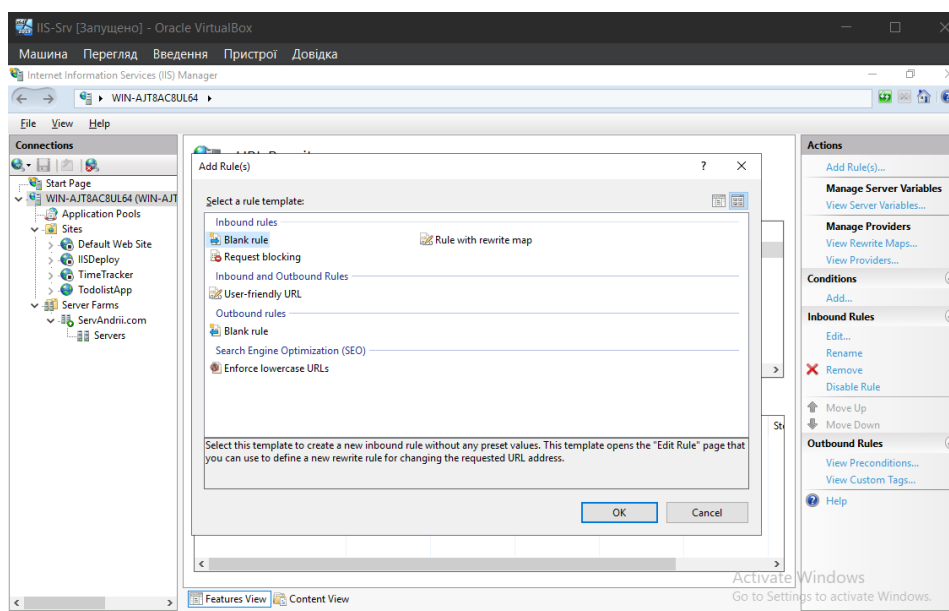


Рисунок 3.8 – Додавання нового правила

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

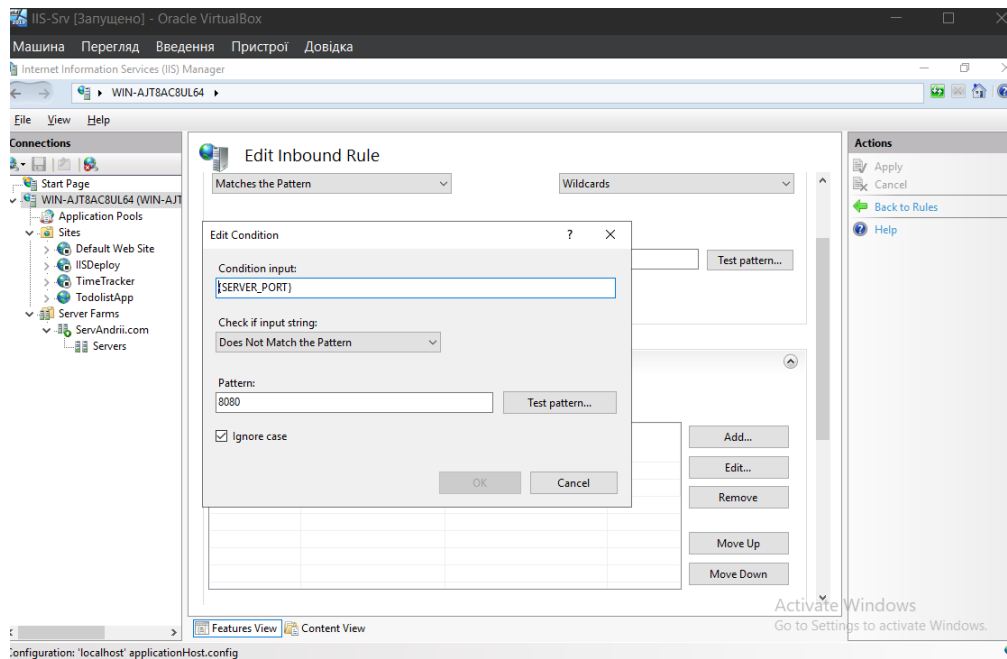


Рисунок 3.9 – Умова за портом

У середовищі веб-серверів Microsoft балансування навантаження є ключовим механізмом для забезпечення високої доступності, масштабованості та рівномірного розподілу трафіку між кількома екземплярами веб-додатків. IIS підтримує різні алгоритми балансування навантаження, які можна налаштувати відповідно до специфіки навантаження та вимог до продуктивності. В ARR представлено перелік алгоритмів балансування навантаження (рис.3.10), доступних для налаштування, зокрема:

Weighted Round Robin

Цей алгоритм розподіляє запити між серверами на основі заданих ваг. Кожному серверу надається певна вага, яка визначає, скільки запитів він має обслуговувати у співвідношенні до інших серверів. Наприклад, якщо один сервер має вагу 3, а інший - 1, то перший отримає три запити на кожен запит другого. Цей метод ефективний, коли сервери мають різні потужності або ресурси.

До переваг методу належить його гнучкість у розподілі навантаження, а також придатність для неоднорідного середовища серверів. Проте недоліком є те, що він не враховує поточний стан серверів, наприклад, рівень їх завантаженості чи наявність затримок.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Server Variable Hash

У цьому випадку запит направляється до сервера на основі хешу значення певної змінної середовища сервера (наприклад, IP-адреси клієнта, HTTP-заголовку тощо). Це забезпечує певну стійкість до сесійної прив'язки - клієнти, які надсилають однакові значення змінної, потрапляють на той самий сервер.

Це робить метод особливо корисним у сценаріях, де важливо зберігати сесію користувача на одному сервері, та забезпечує передбачуваність у маршрутизації запитів. Однак при дуже нерівномірному розподілі значень така схема може бути менш ефективною, оскільки деякі сервери можуть отримувати непропорційно велике навантаження.

Query String Hash

Цей метод базується на хешуванні параметрів query string (рядка запиту) URL-адреси. Запити направляються на сервери відповідно до результатів хешування значень у рядку запиту. Це дозволяє спрямовувати подібні запити до одного й того ж сервера, що може покращити кешування та узгодженість обробки. Метод добре підходить для динамічних веб-додатків, які активно використовують параметри в URL, однак він є чутливим до змін у структурі самого рядка запиту, що може спричинити порушення стабільності маршрутизації. Крім того, існує ризик нерівномірного розподілу навантаження між серверами.

Request Hash

Цей варіант використовує хешування цілих HTTP-запитів (або їх визначених частин) для розподілу між серверами. Алгоритм може враховувати метод, шлях, заголовки, тіло запиту тощо. Такий підхід забезпечує високу детермінованість у маршрутизації, тобто запити з однаковими характеристиками будуть оброблятися одним і тим же сервером [\[28\]](#).

Такий метод особливо ефективний для RESTful сервісів, де часто надходять подібні запити, і де важливо зберігати узгодженість у їх обробці. Крім того, він дозволяє враховувати вміст запиту при балансуванні навантаження. Водночас цей підхід може ускладнити процес налагодження системи, а також потребує уважного

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

вибору параметрів, які беруть участь у хешуванні, щоб уникнути непередбачуваної маршрутизації або перевантаження окремих серверів.

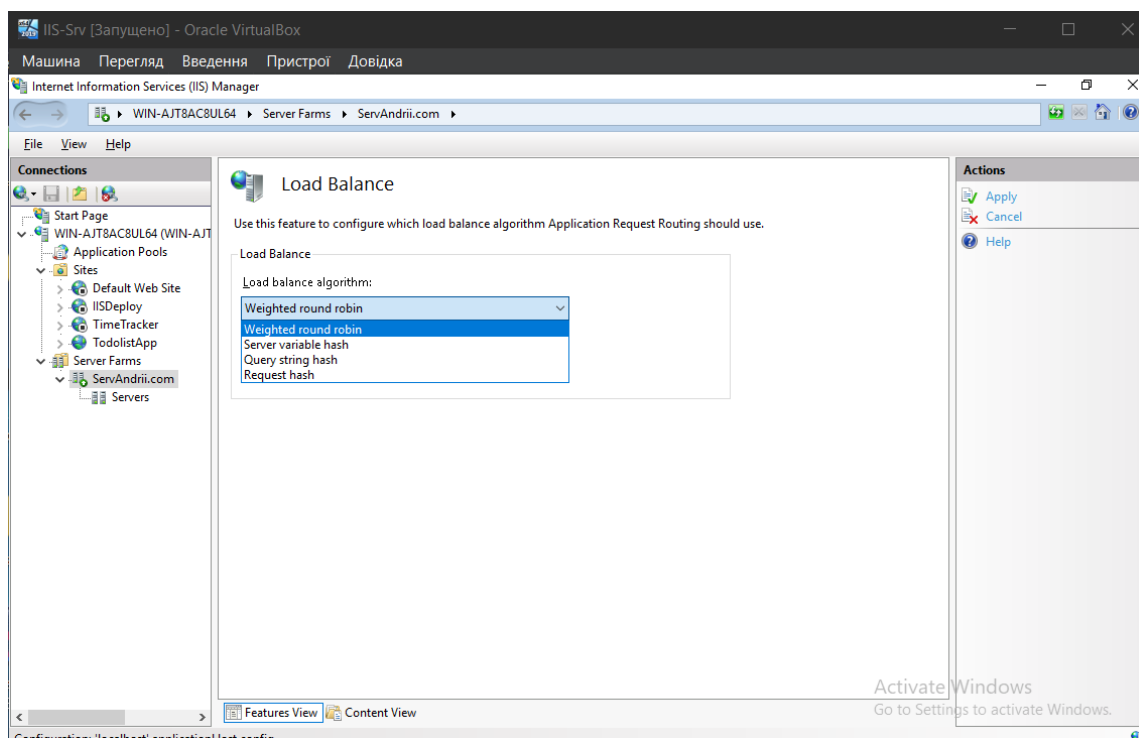


Рисунок 3.10 – Варіанти балансування навантаження

3.2 Розгортання сайту на вебсервері

Для розгортання веб-застосунків, створених на платформі ASP.NET Core, на сервері з використанням IIS необхідно встановити Hosting Bundle. Цей пакет містить усі компоненти, потрібні для повноцінного функціонування застосунку у виробничому середовищі. Зокрема, до його складу входить модуль ASP.NET Core Module (ANCM), який виконує ключову роль у зв'язуванні веб-сервера IIS з внутрішнім веб-сервером Kestrel, що безпосередньо обробляє HTTP-запити.

Hosting Bundle також включає середовище виконання .NET та ASP.NET Core Runtime, які забезпечують можливість запуску зібраних .NET-застосунків. Без наявності відповідної версії середовища виконання застосунок не зможе бути коректно завантажений або буде видавати помилки під час старту.

Також, увагу слід приділити тому, що версія .NET, встановлена на сервері, повинна точно відповідати тій версії, під яку було зібрано застосунок. Це гарантує

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

сумісність усіх компонентів і стабільну роботу програмного забезпечення. Усі ці компоненти можна встановити з офіційного сайту Microsoft (рис.3.11 – 3.12)

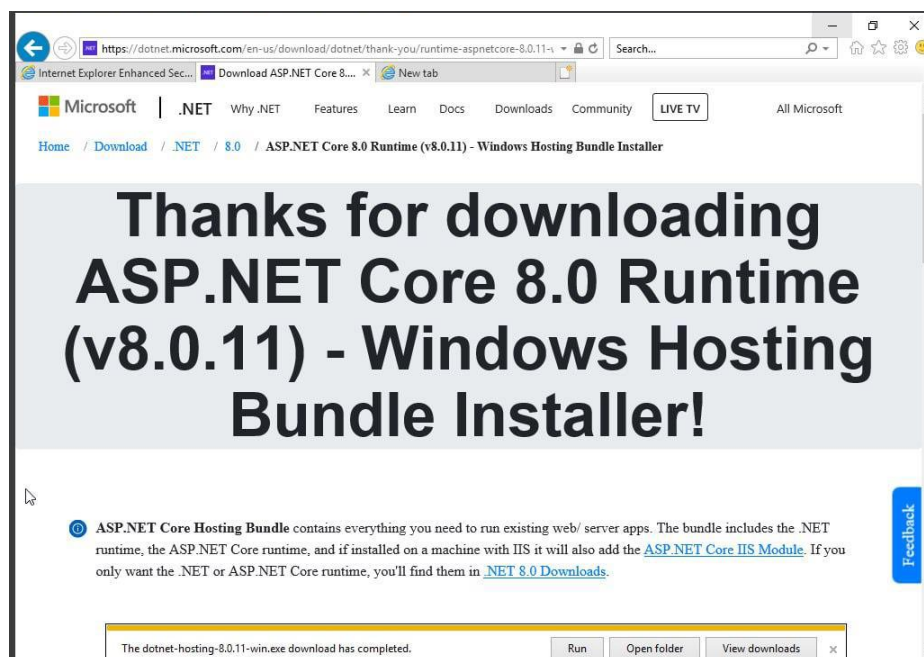


Рисунок 3.11 – Встановлення hosting bundle.

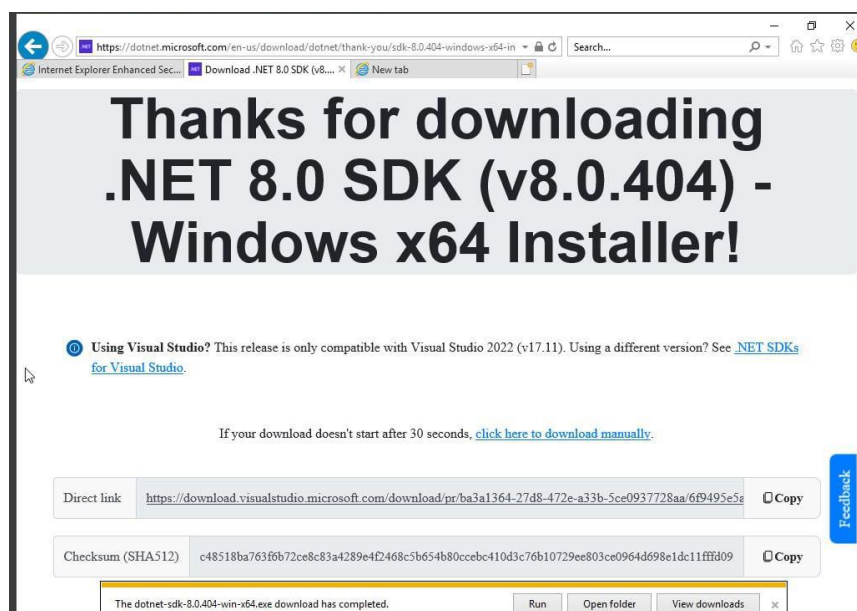


Рисунок 3.12 – Встановлення .NET.

У середовищі розробки Visual Studio процес підготовки застосунку до розгортання реалізується за допомогою вбудованого інструменту публікації (Publish)

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

(рис.3.13). Цей інструмент дозволяє сформувати повністю скомпільований і готовий до розміщення пакет, який містить усі необхідні файли для запуску застосунку на сервері.

У рамках цього процесу обирається відповідний метод публікації, наприклад, створення локальної папки. У результаті цього Visual Studio виконує побудову проєкту, копіює необхідні залежності, конфігураційні файли та ресурси до вказаної директорії, формуючи повноцінний набір файлів для подальшого розгортання.

Цей підхід дозволяє розділити процес розробки й експлуатації, оскільки створений публікаційний пакет можна передати для розміщення на веб-сервері без додаткових дій з боку розробника. Таким чином, використання функціоналу публікації в Visual Studio сприяє стандартизації процесу підготовки застосунку до розгортання та забезпечує його відповідність вимогам цільового середовища.

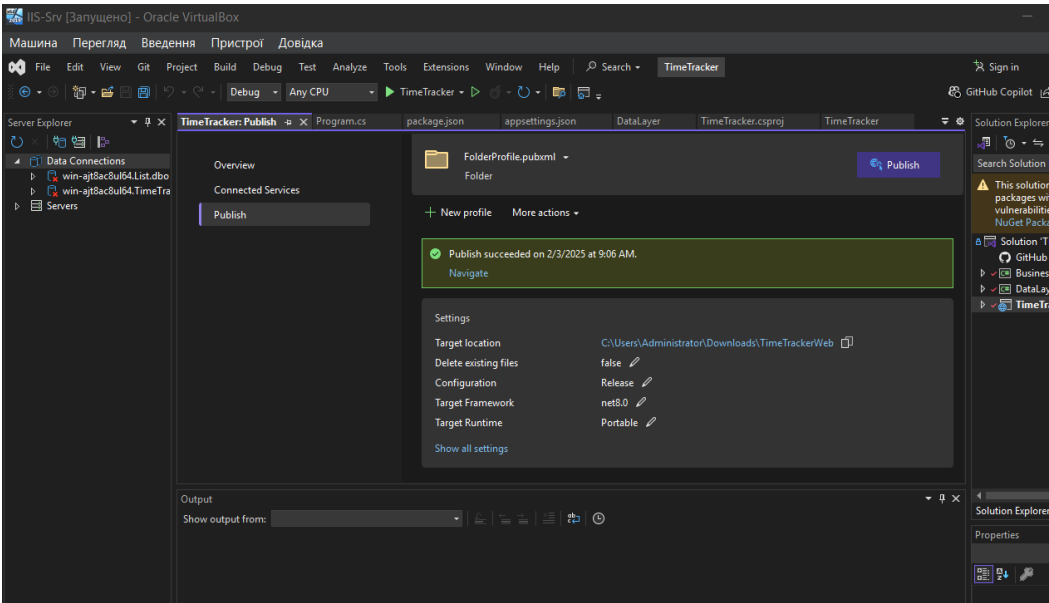


Рисунок 3.13 – Publish.

Після цього, папку з нашим проєктом потрібно перенести у каталог wwwroot (рис.3.14). Папка wwwroot - це стандартна коренева папка для веб-контенту у веб-сервері IIS

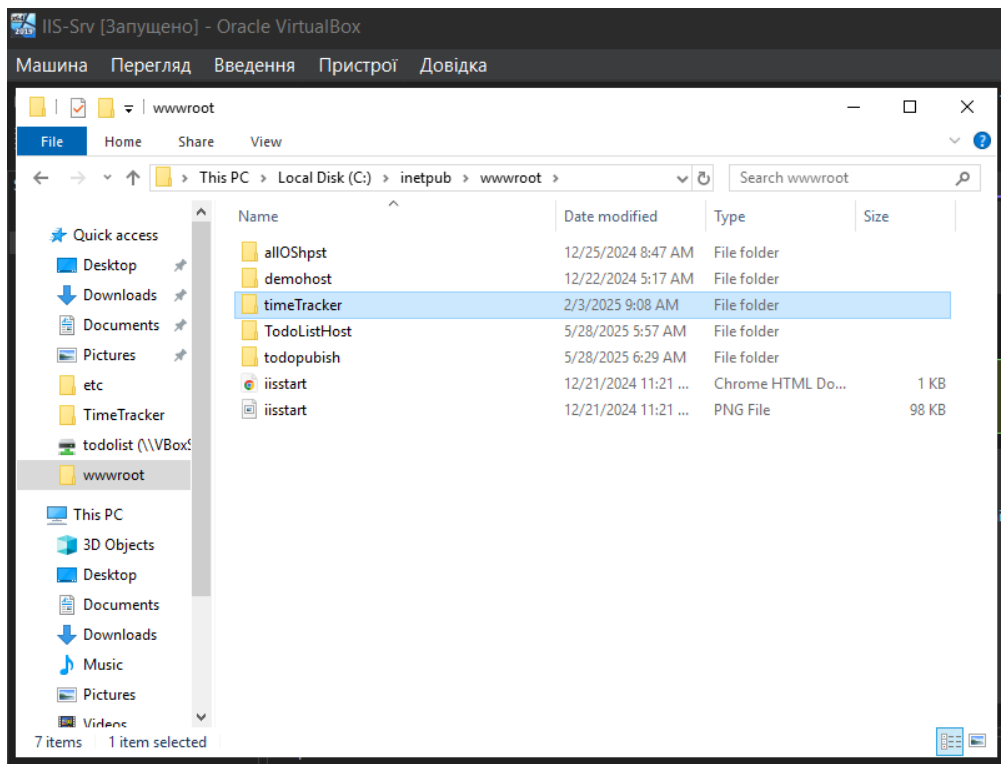


Рисунок 3.14 – Додаємо проєкт у wwwroot.

Під час додавання нового сайту в IIS необхідно вказати його назву (рис.3.15 – 3.17). Також задається шлях до кореневої директорії, яка містить скомпільовані файли застосунку, отримані внаслідок попереднього етапу публікації у середовищі Visual Studio. Також, можна обрати ip address та порт на якому буде працювати наш сайт.

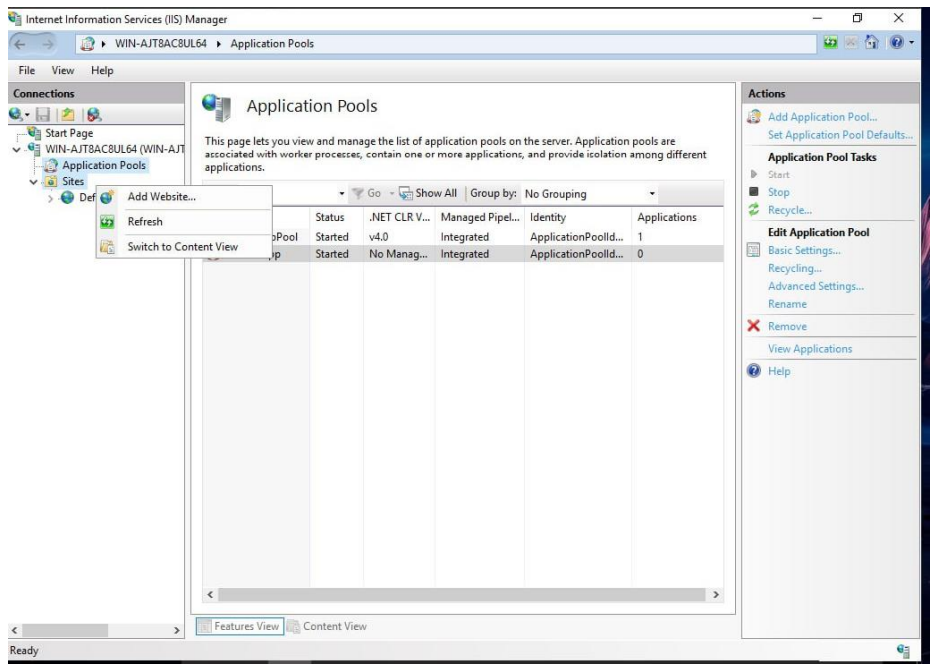


Рисунок 3.15 – Додаємо вебсайт у IIS

Add Website

Site name: TimeTracker Application pool: TimeTracker Select...

Content Directory

Physical path: C:\inetpub\wwwroot\timeTracker ...

Pass-through authentication

Connect as... Test Settings...

Binding

Type: http IP address: All Unassigned Port: 80

Host name: myapp.andrii.com

Example: www.contoso.com or marketing.contoso.com

☒ Start Website immediately

OK Cancel

Рисунок 3.16 – Задаємо параметри конфігурації

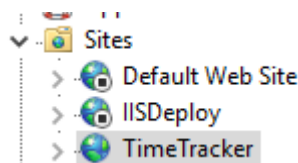


Рисунок 3.17 – Сайт було успішно додано до IIS

У файлі hosts виконуємо прив'язку імені хоста до відповідної IP-адреси (рис.3.18). Це дає змогу локально переадресовувати доменні імена на конкретні мережеві адреси без участі зовнішніх DNS-серверів. У контексті налаштування веб-сайту це дозволяє забезпечити коректну маршрутизацію запитів до сервера за заданим доменом, імітуючи роботу сайту в умовах реального доменного середовища.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

hosts - Notepad
File Edit Format View Help
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com           # source server
#       38.25.63.10       x.acme.com               # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1         localhost
#       ::1               localhost

10.0.2.15               myapp.andrii.com

```

Рисунок 3.18 – Конфігурація файлу hosts

3.3 Тестування роботи вебсервера

Після завершення налаштування вебсервера та конфігурації середовища розміщення у системі IIS, було здійснено запуск вебсайту з метою перевірки його працездатності (рис.3.19). Одним з ключових етапів є тестування доступності вебзастосунку за вказаною IP-адресою, та host name. Успішне відкриття вебінтерфейсу через браузер свідчить про правильність налаштування вебсервера, наявність зв'язку між клієнтом і сервером, а також про готовність системи до подальшої експлуатації.

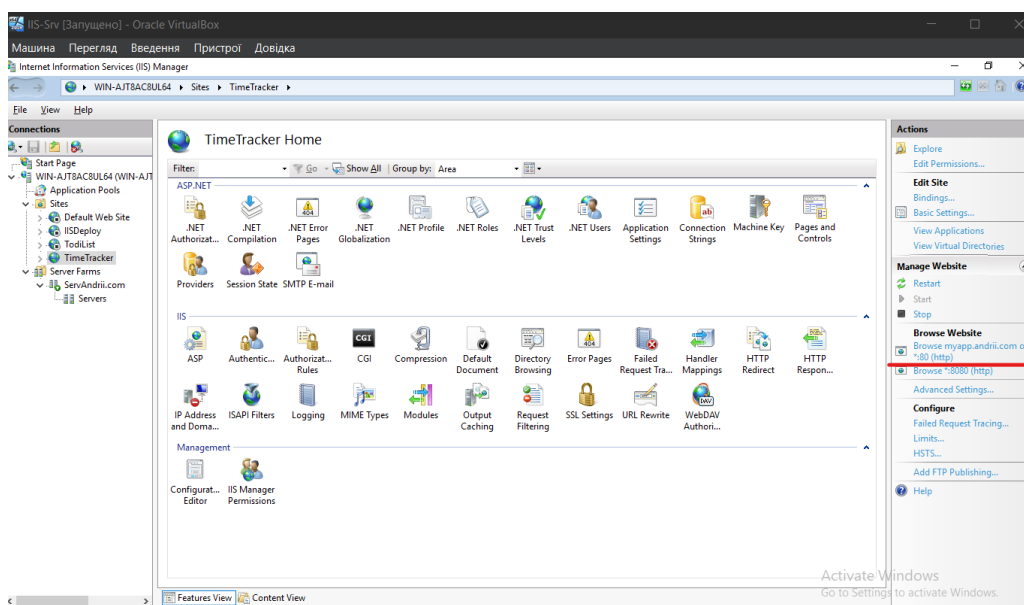


Рисунок 3.19 – Запуск вебсайту

Відображається налаштоване доменне ім'я (host name), що свідчить про коректну роботу вебсервера (рис.3.20).

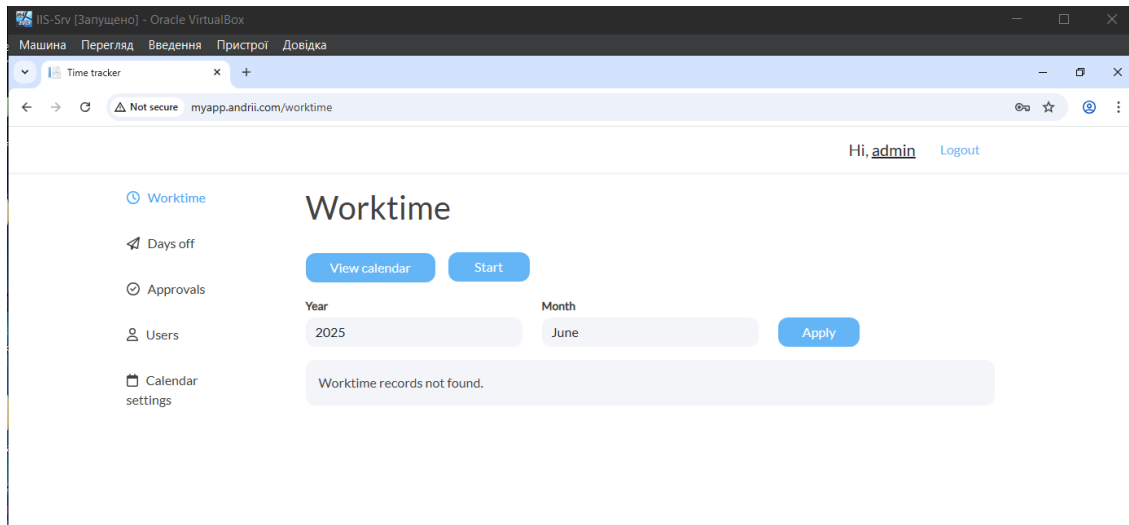


Рисунок 3.20 – Демонстрація успішності запуску

Для перевірки успішного підключення бази даних до вебсервера було здійснено візуальне порівняння вмісту таблиці користувачів, що зберігається у Microsoft SQL Server, із даними, які відображаються на вебінтерфейсі. Зіставлення усіх параметрів дозволило впевнено зробити висновок про коректність налаштування з'єднання між серверною частиною вебзастосунку та системою керування базами даних (рис.3.21 – 3.23). Це свідчить про стабільну роботу запитів до бази, правильне отримання й відображення інформації на стороні клієнта.

Як уже було зазначено раніше, у системі зберігання облікових даних користувачів реалізовано механізм захисту паролів із використанням salt. Це додаткове значення, яке генерується випадковим чином і додається до паролю перед його хешуванням. У таблиці бази даних ми можемо переконатися в наявності відповідного поля Salt, яке містить індивідуальні рядки для кожного користувача, що підтверджує застосування цього методу.

Такий підхід гарантує, що навіть при введенні однакових паролів різними користувачами, результати хешування будуть різними завдяки унікальності значень salt. Це забезпечує значно вищий рівень безпеки, оскільки унеможливорює використання типових атак, як-от підбір хешів із заздалегідь підготовлених таблиць

(rainbow tables). Таким чином, навіть у разі витоку бази даних, зломисник не зможе отримати справжні паролі користувачів.

	Id	Name	Surname	Email	Password	Salt	IsAdmin	IsActive
1	A21723A2-A5F3-4066-864D-3AEF0533478D	admin	admin	admin@gmail.com	f3g5USGFQ2UZ4r8Vsu2bfA==	0LNxldjU1qC2ye2Zx+yeY7Q+iyEgZkG3	1	1
2	B7B3CE16-376C-4F0A-8489-4E681E6CFA5D	fulltimer	fulltimer	fulltimer@gmail.com	jb0mPlLpqn2k6JQ6RP8S/A==	bO8kn3G+Vjd88MHpzldQOdeRzXDO/ErX	1	1
3	06184100-EAB5-47BC-947F-61B5158D202D	FULLLLTIME	FULLLLTIME	FULLLLTIME@gmail.com	as3laYdXZwBCFxnR1FYUdw==	6vIk/PHsa4PWC5dOC4JeDiTdZD37uTbK	0	1
4	6FDDB5C1-7B62-4411-8418-7D2AFD900A0E	noadmin	noadmin	noadmin@gmail.com	FN3xL7Oyv55UVN7xeMfnw==	ikuNQyJv3wtyoYELY0Fva4zHBpTJVa2X	0	1
5	90B58E87-04A4-4F7C-AF76-AC5394DBE935	Kostas	giannis	kostas@gmail.com	3qSWbSpFT/Ge4TsN4J6w==	IZ6WT4Wm3u662H5pa04ebqZSGoOKHm+a	0	1
6	CE9B95D9-75BF-49A5-8154-C33891260952	approver	andrii	approver@gmail.com	K2ENpcXifxsC03UfJHHw==	MiS241sK8rzD+I8cR+sCzOak9eNBwb6	1	1

Рисунок 3.21 – Вибірка таблиці Users

Name	Email	Employment type	Employment date
admin admin	admin@gmail.com	part-time	7/24/2024
approver andrii	approver@gmail.com	full-time	7/25/2023
FULLLLTIME FULLLLTIME	FULLLLTIME@gmail.com	full-time	2/24/2024
fulltimer fulltimer	fulltimer@gmail.com	part-time	7/25/2024
Kostas giannis	kostas@gmail.com	part-time	2/28/2025
noadmin noadmin	noadmin@gmail.com	full-time	5/13/2025

Total records: 6

Рисунок 3.22 – Перевірка даних користувачів, отриманих з бази даних

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

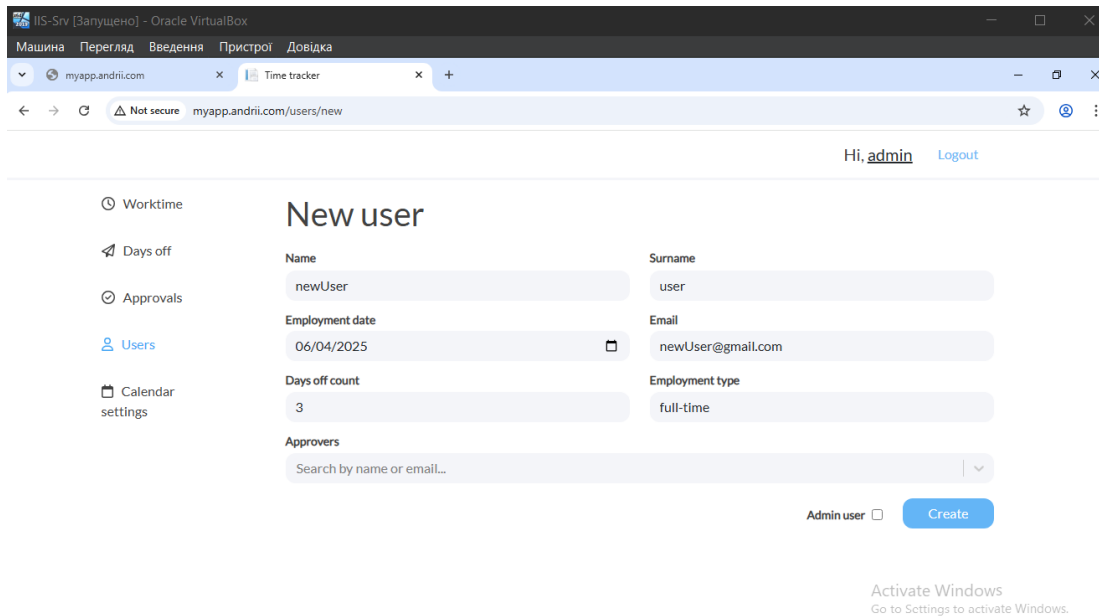


Рисунок 3.23 – Створення нового користувача

Згідно з результатами моніторингу системи, спостерігається стабільна робота двох серверів, що виконують функцію балансування навантаження. Обидва вузли залучені до обробки вхідних запитів у рівнозначній пропорції, що становить 50% на кожен сервер (рис.3.24 – 3.25).

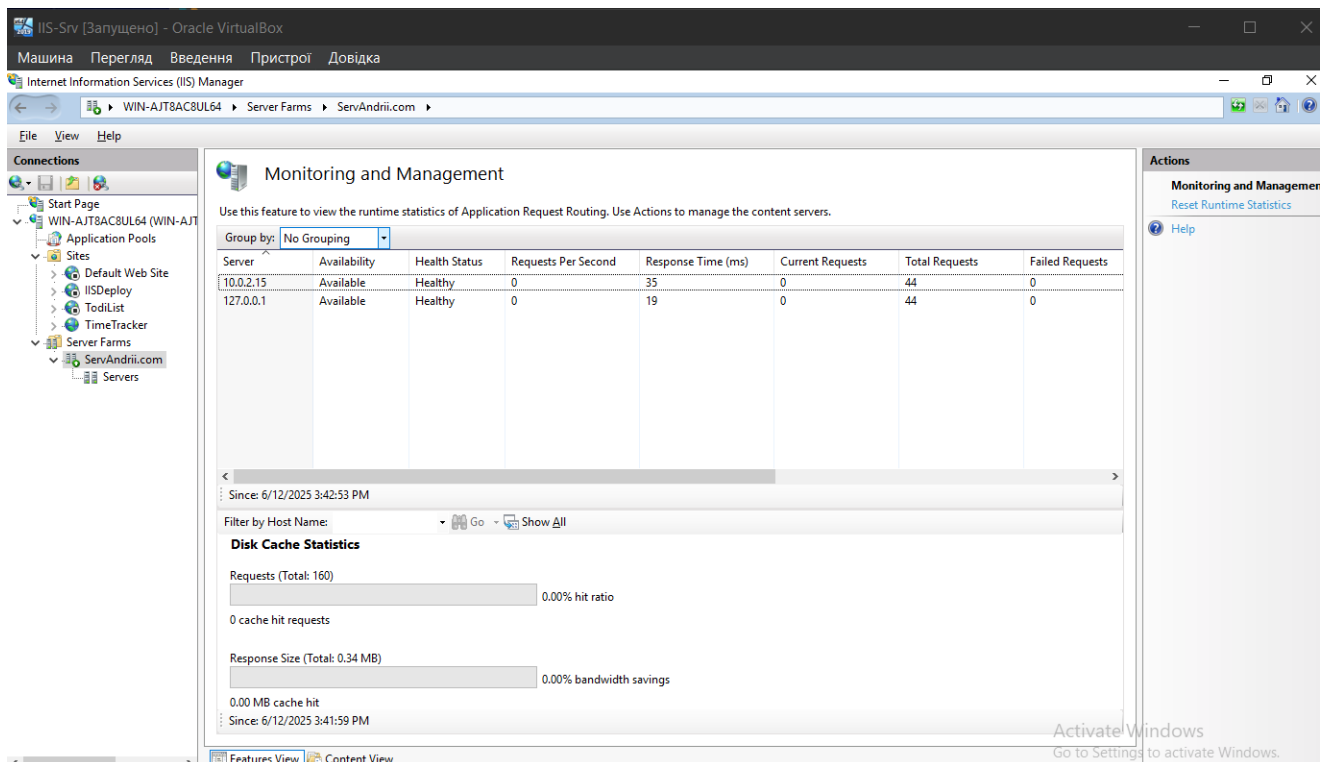


Рисунок 3.24 – Огляд роботи серверів

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

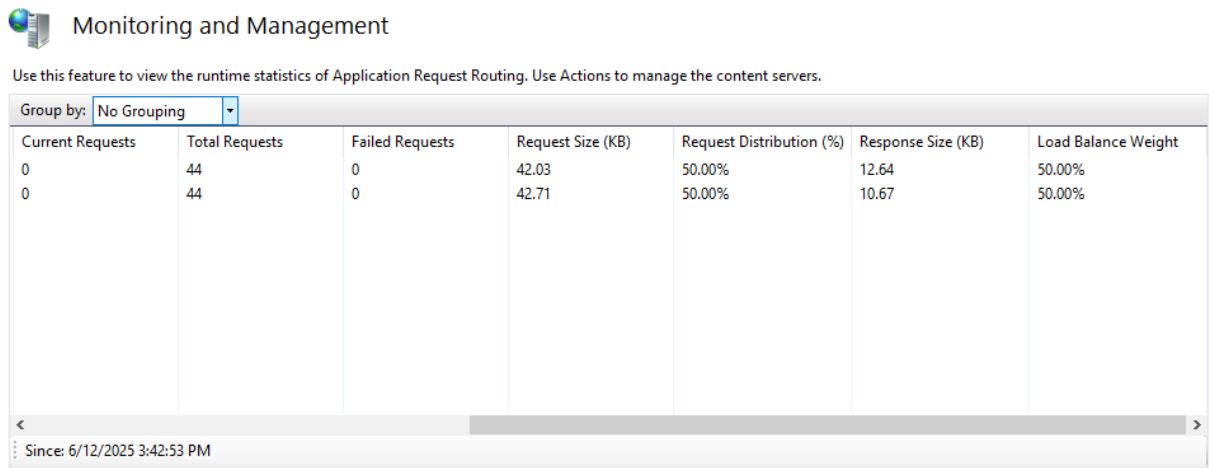


Рисунок 3.25 – Огляд роботи серверів

З метою тестування ефективності механізмів балансування навантаження змінимо конфігурацію розподілу трафіку між серверами. Зокрема, передбачено можливість ручного коригування ваги кожного вузла в балансувальнику, що дозволяє моделювати асиметричні умови навантаження. Наприклад, один із серверів може бути налаштований на обробку 75% усіх вхідних запитів, тоді як інший - лише 25% (рис.3.26). Такий підхід дає змогу дослідити поведінку системи в умовах нерівномірного розподілу ресурсів, перевірити стійкість окремих компонентів до навантаження та оцінити гнучкість інфраструктури в адаптації до змін у трафіку.

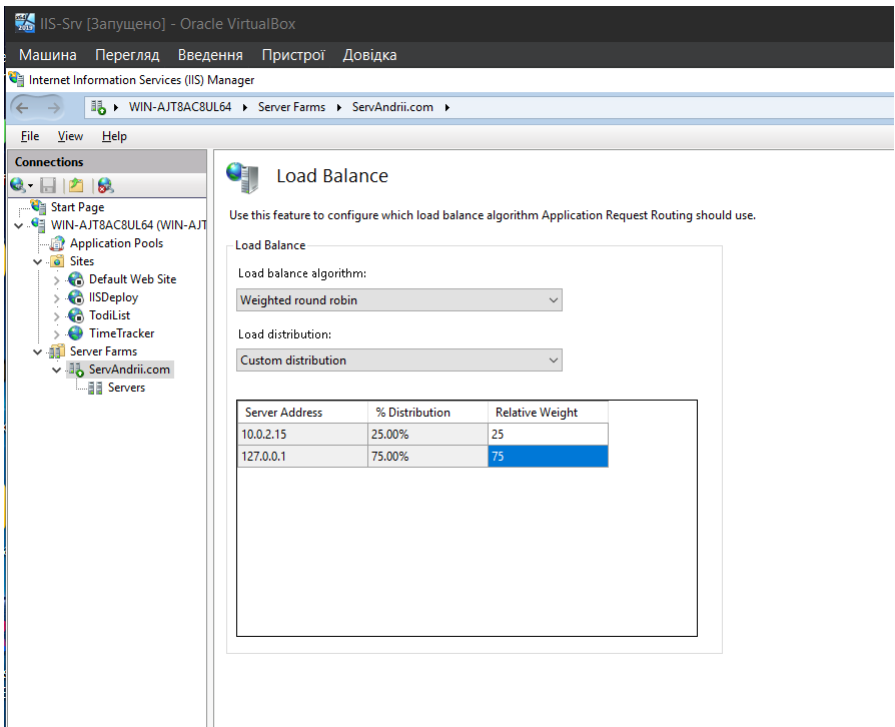


Рисунок 3.26 – Змінюємо налаштування балансування навантаження

Аналітичні показники підтверджують ефективність застосованого алгоритму зваженого балансування (рис.3.27). Параметр Request Distribution (%) відображає фактичний розподіл вхідних HTTP-запитів між серверами - 74.29% запитів було спрямовано на вузол із більшою вагою, що майже повністю відповідає очікуваному теоретичному значенню. Інший сервер, відповідно, обробив 25.71% запитів. Також можна простежити відмінності у таких характеристиках, як Response Time (мс), Total Requests, Request Size (KB) та Response Size, що вказує на реальні зміни в динаміці роботи кожного з вузлів унаслідок нерівномірного навантаження.

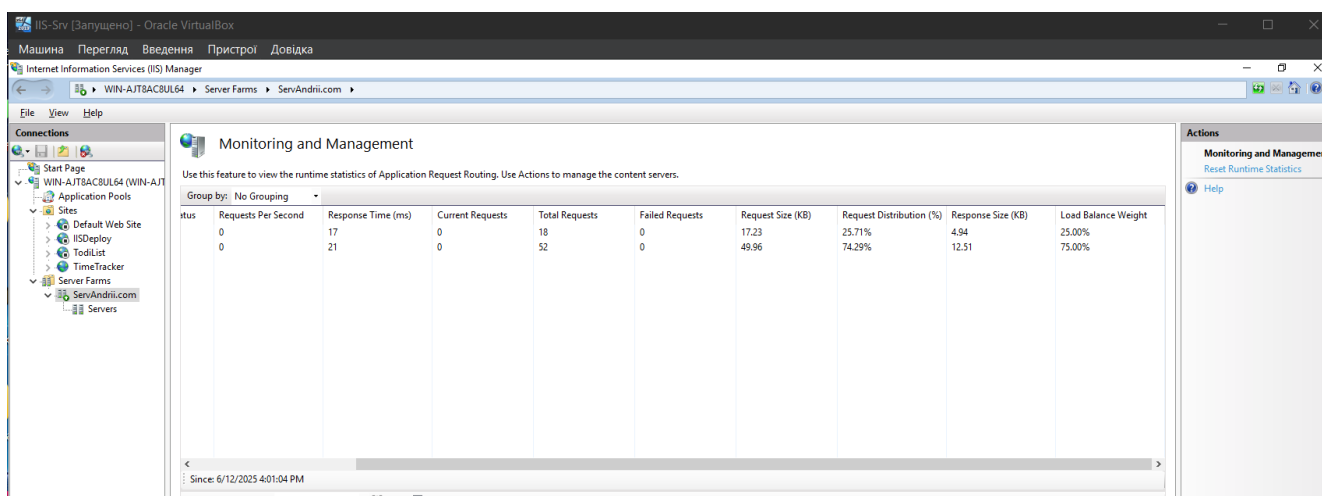


Рисунок 3.27 – Огляд роботи серверів після зміни налаштувань

Механізм Health Test у середовищі балансування навантаження виконує критично важливу функцію забезпечення високої доступності та надійності вебзастосунку. Його основне призначення полягає у періодичному моніторингу працездатності серверів, які входять до складу серверної ферми (рис.3.28).

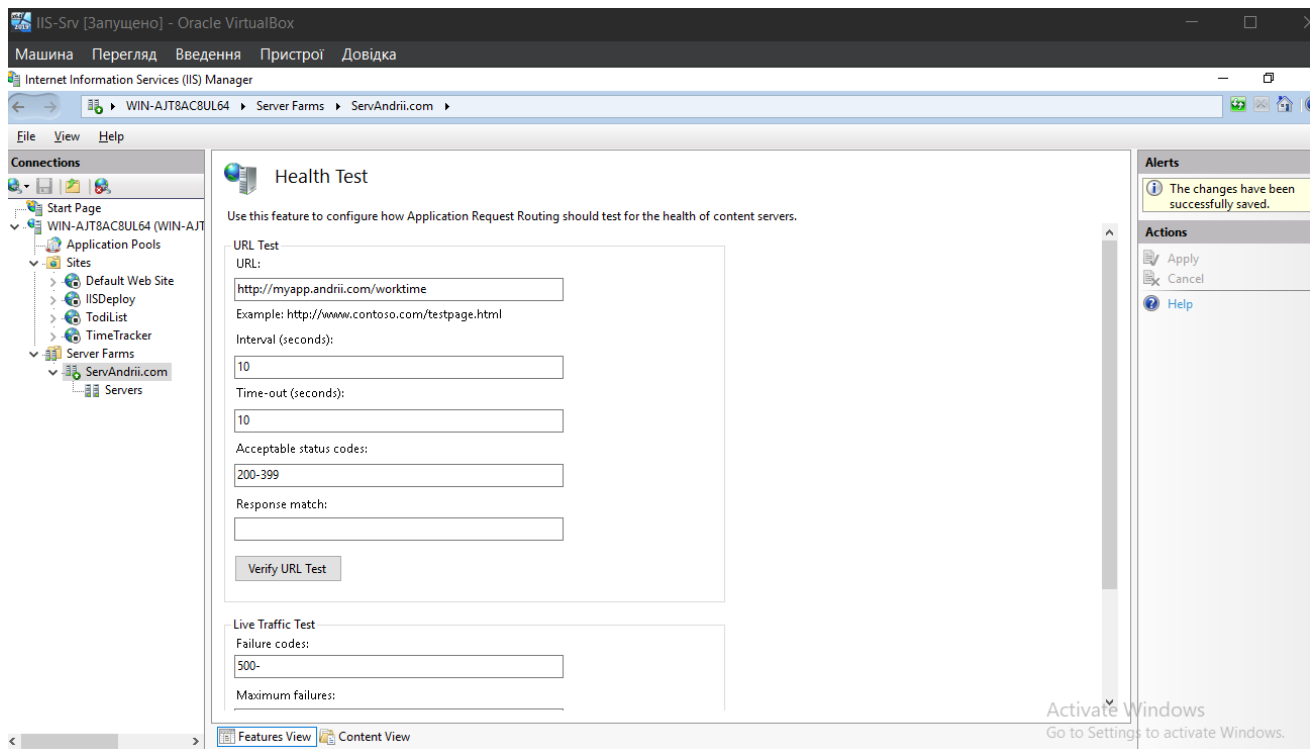


Рисунок 3.28 – Налаштування Health Test

Під час роботи системи Health Test надсилає запити до попередньо визначених ресурсів на кожному з вузлів. У разі, якщо сервер не відповідає у встановлений термін або повертає некоректну відповідь, він автоматично виключається з пулу доступних для балансувальника серверів. Це дозволяє уникнути ситуацій, за яких запити користувачів перенаправляються на несправні або перевантажені вузли, що, у свою чергу, знижує ризик втрати даних і забезпечує безперервність надання послуг [30][31].

У процесі тестування механізму моніторингу було примусово відключено сайт на підключених серверах. У результаті цього, після чергового виконання Health Test, система виявила відсутність коректної відповіді на тестовий запит. В інтерфейсі моніторингу відповідний сервер набув статусу Unhealthy, що свідчить про його тимчасове виключення з процесу розподілу запитів (рис.3.29). Така поведінка системи підтверджує правильну роботу налаштованого механізму перевірки стану та забезпечення відмовостійкості вебзастосунку.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

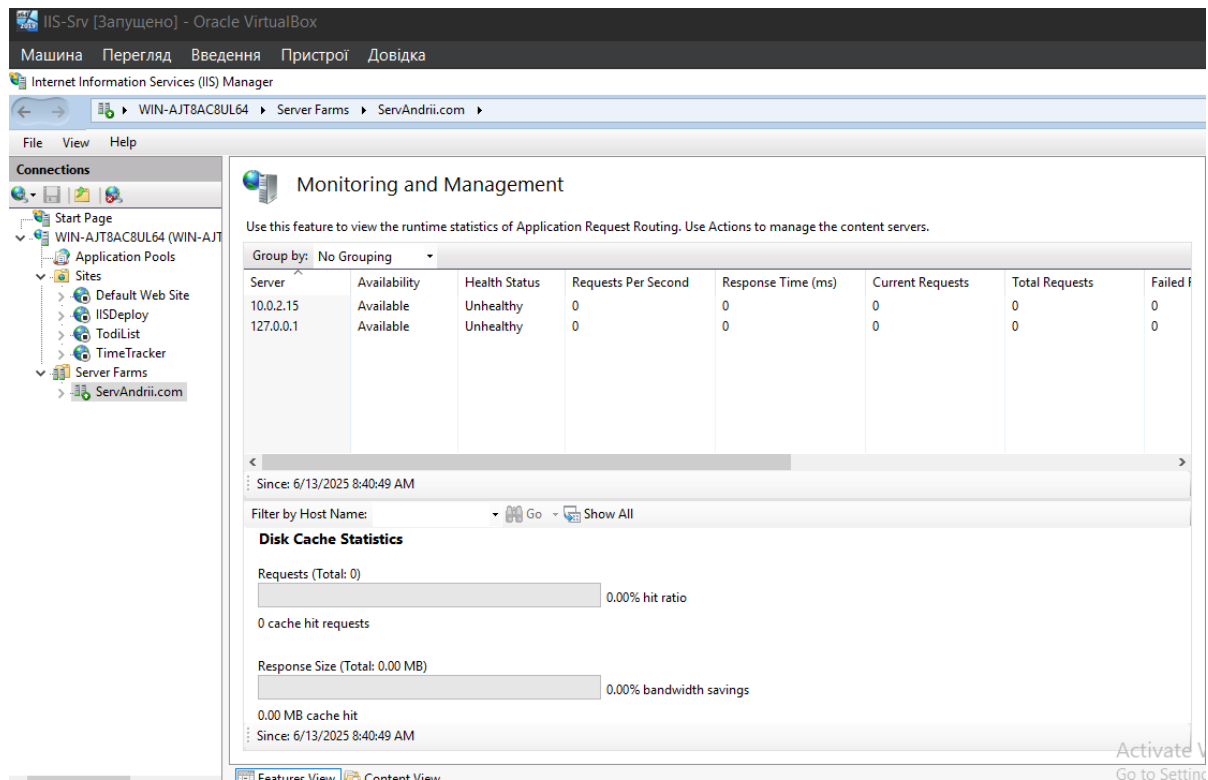


Рисунок 3.29 – Перевірка Health Test

Висновки до третього розділу

У цьому розділі було детально розглянуто процес розгортання вебзастосунку на сервері з використанням IIS, а також впровадження механізмів моніторингу, маршрутизації та балансування навантаження з метою підвищення стабільності, продуктивності та безпеки системи. Основним інструментом виступає технологія Application Request Routing (ARR), яка забезпечує зворотне проксіювання та розподіл запитів між кількома екземплярами сервера. Було налаштовано Server Farm із двома серверами, реалізовано гнучкі правила маршрутизації за допомогою URL Rewrite, активовано функції моніторингу, кешування та перевірки доступності вузлів. У результаті побудови серверного середовища було створено умови для стабільної роботи вебзастосунку в умовах реального навантаження.

Подальше тестування підтвердило коректність налаштувань: вебзастосунок успішно підключається до бази даних, відображає актуальну інформацію на клієнтській стороні, а реалізовані механізми захисту працюють надійно. Система моніторингу та балансування навантаження забезпечила рівномірний та зважений розподіл трафіку між серверами, а також продемонструвала здатність автоматично

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

реагувати на відмови окремих вузлів, підтримуючи безперервну роботу вебза-
стосунку.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У межах кваліфікаційної роботи було успішно реалізовано комплекс заходів, спрямованих на створення та налаштування серверної інфраструктури для розміщення вебзастосунку з використанням технологій Microsoft IIS та ASP.NET Core.

Для початку було визначено мету дослідження, обґрунтовано вибір засобів для побудови вебстеку, проведено аналіз архітектури вебсерверів і розглянуто принципи роботи обраної платформи. Це дозволило сформулювати технічні вимоги до реалізації проєкту.

Було здійснено поетапне налаштування серверного середовища, включаючи встановлення серверної операційної системи, конфігурацію вебсерверу IIS, підтримку платформи ASP.NET Core та підключення до бази даних. Усі етапи були реалізовані з урахуванням вимог до безпеки та надійності.

Також було детально розглянуто процес розгортання вебзастосунку на IIS із використанням механізмів моніторингу та балансування навантаження на базі Application Request Routing (ARR). Було налаштовано серверну ферму, правила маршрутизації, кешування та перевірки доступності вузлів, що забезпечило стабільність та надійність системи. Проведено тестування працездатності вебсайту, перевірку взаємодії з базою даних, оцінено ефективність роботи системи в умовах балансування навантаження, а також проаналізовано роботу моніторингових механізмів у разі відмови окремих вузлів.

Отже, усі поставлені завдання в межах кваліфікаційної роботи були виконані в повному обсязі. Побудована система продемонструвала стабільність, масштабованість і здатність до роботи в умовах реального навантаження, що підтверджує доцільність обраних технічних рішень.

Згодом розроблену систему можна вдосконалити та розширити відповідно до зростаючих потреб користувачів і навантаження. Подальший розвиток може включати автоматизацію процесів обслуговування, поліпшення засобів моніторингу, підвищення рівня безпеки та інтеграцію з іншими сервісами. Крім того, можливе масштабування архітектури з урахуванням нових вимог або переходу до більш складних інфраструктурних рішень.

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Historical trends in the usage statistics of web servers [Електроний ресурс] – Режим доступу до ресурсу: https://w3techs.com/technologies/history_overview/web_server
2. Introduction to IIS Architectures [Електроний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/iis/get-started/introduction-to-iis/introduction-to-iis-architecture>
3. Using the Application Request Routing Module [Електроний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/iis/extensions/planning-for-arr/using-the-application-request-routing-module>
4. Access Protection [Електроний ресурс] – Режим доступу до ресурсу: <https://www.iis.net/overview/security/accessprotection>
5. Introduction to nginx [Електроний ресурс] – Режим доступу до ресурсу: <https://dianfajar.com/nginx-the-lightning-fast-web-server-you-should-be-using/>
6. An In-Depth Look at Apache Web Server [Електроний ресурс] – Режим доступу до ресурсу: <https://www.hostmycode.in/blog/an-in-depth-look-at-apache-web-server>
7. What is an origin server? [Електроний ресурс] – Режим доступу до ресурсу: <https://developers.cloudflare.com/reference-architecture/architectures/cdn/>
8. What Is Windows Server? [Електроний ресурс] – Режим доступу до ресурсу: <https://www.solarwinds.com/resources/it-glossary/windows-server>
9. What is a Linux Server and Why use it [Електроний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/linux-unix/what-is-a-linux-server-and-why-use-it/>
10. UNIX [Електроний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/ru-ru/cpp/c-runtime-library/unix?view=msvc-170> .
11. Інформація про macOS Server [Електроний ресурс] – Режим доступу до ресурсу: <https://mac.org.ua/macOS/about-macos-server/> .

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

12. ASP.NET tutorials [Електроний ресурс] – Режим доступу до ресурсу:
<https://learn.microsoft.com/en-us/aspnet/tutorials>
13. Using Node.js for Real-Time Applications [Електроний ресурс] – Режим доступу до ресурсу: <https://medium.com/%40shishird531/the-benefits-of-using-node-js-for-real-time-applications-c33b97a8deac> .
14. Python [Електроний ресурс] – Режим доступу до ресурсу:
<https://leapcell.medium.com/top-10-python-web-frameworks-compared-cfed1ec3f9df> .
15. PHP Introduction [Електроний ресурс] – Режим доступу до ресурсу:
https://www.w3schools.com/php/php_intro.asp .
16. Entity Framework [Електроний ресурс] – Режим доступу до ресурсу:
https://en.wikipedia.org/wiki/Entity_Framework
17. Dapper ORM [Електроний ресурс] – Режим доступу до ресурсу:
<https://en.wikipedia.org/wiki/Dapper ORM>
18. What is PostgreSQL? [Електроний ресурс] – Режим доступу до ресурсу:
<https://www.postgresql.org/about/>
19. MySQL, PostgreSQL [Електроний ресурс] – Режим доступу до ресурсу:
<https://aws.amazon.com/ru/compare/the-difference-between-mysql-vs-postgresql/>
20. MongoDB [Електроний ресурс] – Режим доступу до ресурсу: <https://javascript.org.ua/mongodb-proti-mysql-vibir-pravilno%D1%97-bazi-danih-dlya-vashogo-zastosunku/>
21. Microsoft SQL Server [Електроний ресурс] – Режим доступу до ресурсу:
<https://www.simplyblock.io/glossary/what-is-microsoft-sql-server/>
22. SQL Server Versions and Features [Електроний ресурс] – Режим доступу до ресурсу: <https://www.mssqltips.com/tutorial/sql-server-versions-and-features/>
23. Server Core vs Server with Desktop Experience install options [Електроний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows-server/get-started/install-options-server-core-desktop-experience>

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

24. Difference between installing RDS the 'normal' way and the special RDS deployment option [Електроний ресурс] – Режим доступу до ресурсу: <https://serverfault.com/questions/721967/whats-the-difference-between-installing-rds-the-normal-way-and-the-special-rd>
25. Adding Salt to Hashing: A Better Way to Store Passwords [Електроний ресурс] – Режим доступу до ресурсу: <https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> .
26. Dapper: A High Performance Micro ORM [Електроний ресурс] – Режим доступу до ресурсу: <https://www.projectscouts.com/dapper-a-high-performance-micro-orm/>
27. Впровадження GraphQL у проєктах [Електроний ресурс] – Режим доступу до ресурсу: <https://markup-ua.com/vprovadzhennya-graphql-u-proyektax-praktichni-poradi/>
28. ARR as a load balancer [Електроний ресурс] – Режим доступу до ресурсу: <https://techcommunity.microsoft.com/blog/iis-support-blog/application-request-routing-and-url-rewrite-%E2%80%93-part-1server-farms/347924>
29. URL Rewrite [Електроний ресурс] – Режим доступу до ресурсу: <https://www.iis.net/downloads/microsoft/url-rewrite> .
30. Define and Configure an Application Request Routing Server Farm [Електроний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/iis/extensions/configuring-application-request-routing-arr/define-and-configure-an-application-request-routing-server-farm> .
31. HTTP Load Balancing using Application Request Routing [Електроний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/iis/extensions/configuring-application-request-routing-arr/http-load-balancing-using-application-request-routing> .

ДОДАТКИ

		Латаи А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняння вебсерверів

Характеристика	IIS (Microsoft)	Nginx	Apache
Платформа	Windows	Кросплатформений	Кросплатформений
Тип серверу	Вебсервер, інтегрований із Windows-інфраструктурою	Вебсервер, реверс-проксі, балансувальник	Вебсервер, з можливістю реверс-проксі
Продуктивність	Висока	Дуже висока	Висока
Підтримка HTTP/2 / HTTP/3	HTTP/2, HTTP/3	HTTP/2, HTTP/3	HTTP/2
Підтримка мов програмування	ASP.NET, PHP, ін. через розширення	Будь-які	Будь-які
Балансування навантаження	Через ARR	Вбудоване	Через сторонні модулі або проксі
Кешування	Через ARR	Вбудоване	Через модулі
SSL/TLS (HTTPS)	Так	Так	Так

Порівняння серверних операційних систем

Характеристика	Windows Server	Linux	Unix	macOS Server
Тип ліцензії	Комерційна	Відкрита (Open Source)	Комерційна	Комерційна
Цільове середовище	Корпорації, , навчальні заклади	Хостинг, веб-сервери, хмари, DevOps	Банки, критичні системи	Малі офіси, навчальні заклади
Популярні компоненти	Active Directory, Group Policy, IIS, Hyper-V	Apache, Nginx, MySQL, Docker, Kubernetes, Ansible, Zabbix	AIX (IBM), Solaris (Oracle)	File Sharing, Profile Manager
Стабільність у роботі	Висока	Дуже висока	Висока	Середня
Безпека	Висока	Дуже висока	Висока	Середня
Гнучкість/налаштування	Обмежена	Максимальна	Обмежена	Мінімальна
Підтримка віртуалізації	Hyper-V	Підтримка різних гіпервізорів	Solaris	Мінімальна

Порівняння СУБД

Характери-стика	PostgreSQL	MySQL	MS SQL Server	MongoDB
Тип СУБД	Реляційна (SQL)	Реляційна (SQL)	Реляційна (SQL)	Документна (NoSQL)
Ліцензія	Відкрита	Відкрита	Комерційна	Відкрита
Структура даних	Таблиці	Таблиці	Таблиці	JSON-подібні документи
Масштабо-ваність	Висока	Висока	Дуже висока	Висока
Продук-тивність	Висока	Висока	Висока	Висока при записі
Сфера за-стосування	Складні си-стеми	Вебсайти, CMS	Корпоративні системи, вебсайти	Гнучкі, мас-штабовані дані
Транзакції	Так	Так	Так	Обмежено

Порівняння редакцій Microsoft SQL Server

Параметр	Enterprise	Standard	Web	Express	Developer
Максимальна кількість ядер CPU	Необмежена	До 24	До 4	1	Необмежена
Максимальний обсяг оперативної пам'яті	Максимальний доступний	До 128 ГБ	Обмежено	1 ГБ	Максимальний доступний
Максимальний розмір бази даних	До 524 ПБ	До 524 ПБ	Зазвичай менший	10 ГБ	До 524 ПБ
Transparent Data Encryption	Так	Ні	Ні	Ні	Так
Реплікація	Повна підтримка	Обмежена	Обмежена	Ні	Повна підтримка
Вартість	Найвища	Середня	Безкоштовна	Безкоштовна	Безкоштовна

ЛІСТИНГ класу User.cs

```
public class User
{
    public Guid Id { get; set; }

    public string Name { get; set; } = null!;

    public string Surname { get; set; } = null!;

    public string Email { get; set; } = null!;

    public string Password { get; set; } = null!;

    public string Salt { get; set; } = null!;

    public bool IsAdmin { get; set; }

    public bool IsActive { get; set; }

    public DateTime EmploymentDate { get; set; }

    public EmploymentType EmploymentType { get; set; }

    public decimal WorkingHoursCount { get; set; }

    public List<Guid> ApproverIds { get; set; } = new List<Guid>();

    public int DaysOffCount { get; set; }

    public User() { }

    public User(UserData userData)
    {
        this.Id = userData.Id;
        this.Name = userData.Name;
        this.Surname = userData.Surname;
        this.Email = userData.Email;
        this.Password = userData.Password;
        this.Salt = userData.Salt;
        this.IsAdmin = userData.IsAdmin;
        this.IsActive = userData.IsActive;
        this.EmploymentDate = userData.EmploymentDate;
        this.EmploymentType = userData.EmploymentType;
        this.WorkingHoursCount = userData.WorkingHoursCount;
        this.ApproverIds = string.IsNullOrEmpty(userData.ApproverIds) ? new
List<Guid>() : userData.ApproverIds.Split(';').Select(id => Guid.Parse(id)).ToList();
        this.DaysOffCount = userData.DaysOffCount;
    }

    public UserData ToData()
    => new()
    {
        Id = this.Id,
        Name = this.Name,
        Surname = this.Surname,
        Email = this.Email,
        Password = this.Password,
        Salt = this.Salt,
        IsAdmin = this.IsAdmin,
        IsActive = this.IsActive,
        EmploymentDate = this.EmploymentDate,
        EmploymentType = this.EmploymentType,
```

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        WorkingHoursCount = this.WorkingHoursCount,
        ApproverIds = string.Join(';', this.ApproverIds),
        DaysOffCount = this.DaysOffCount,
    };
}

```

Лістинг класу UserMutation.cs

```

public class UsersMutation : ObjectGraphType
{
    private readonly IUserProvider userProvider;
    private readonly IAuthenticationService authenticationService;
    private readonly IDaysOffProvider daysOffProvider;

    public UsersMutation(IUserProvider userProvider, IDaysOffProvider daysOffProvider,
        IAuthenticationService authenticationService)
    {
        this.userProvider = userProvider;
        this.authenticationService = authenticationService;
        this.daysOffProvider = daysOffProvider;

        Field<BooleanGraphType>("UserCreation")
            .Description("Create a new user")
            .Argument<NonNullGraphType<CreateUserInputType>>("input")
            .Resolve(context => ResolveUserCreation(context));

        Field<UserType>("UserUpdate")
            .Description("Update user")
            .Argument<NonNullGraphType<UpdateUserInputType>>("user")
            .Resolve(context =>
            {
                var input = context.GetArgument<UpdateUserInput>("user");

                var user = new User()
                {
                    Id = Guid.Parse(input.Id),
                    IsAdmin = input.IsAdmin,
                    IsActive = input.IsActive,
                    EmploymentDate = DateTime.Parse(input.EmploymentDate),
                    EmploymentType = input.EmploymentType,
                    Name = input.Name,
                    Surname = input.Surname,
                    Email = input.Email,
                    WorkingHoursCount = input.EmploymentType == Constants.Employ-
mentType.FullTime ? Constants.MaxWorkingHours : input.WorkingHoursCount,
                    ApproverIds = input.ApproversIdList,
                    DaysOffCount = input.DaysOffCount,
                };

                var updatedUser = userProvider.Update(user);
                if (updatedUser != null)
                    UpdateDayOffApprovals(user.Id, user.ApproverIds);

                return updatedUser;
            });

        private bool ResolveUserCreation(IResolveFieldContext context)
        {
            var input = context.GetArgument<CreateUserInput>("input");
            var salt = authenticationService.GenerateSalt();
            var user = new User()
            {
                Id = Guid.NewGuid(),
                IsAdmin = input.IsAdmin,

```

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        EmploymentDate = DateTime.Parse(input.EmploymentDate),
        Salt = salt,
        EmploymentType = input.EmploymentType,
        Name = input.Name,
        Surname = input.Surname,
        Email = input.Email,
        Password = authenticationService.GenerateHash(input.Password, salt),
        WorkingHoursCount = input.EmploymentType == Constants.Employ-
mentType.FullTime ? Constants.MaxWorkingHours : input.WorkingHoursCount,
        ApproverIds = input.ApproversIdList,
        DaysOffCount = input.DaysOffCount,
    };

    userProvider.Create(user);

    return true;
}

private void UpdateDayOffApprovals(Guid userId, List<Guid> newApproverIds)
{
    var filter = new DayOffRequestFilter() { UserId = userId, StartDate =
DateTime.Now };
    var activeRequestIds = daysOffProvider.GetActiveRequests(filter).Select(r =>
r.Id).ToList();
    var activeApprovals = daysOffProvider.GetApprovals(activeRequestIds).GroupBy(a
=> a.RequestId);

    foreach (var activeRequest in activeApprovals)
    {
        var activeApprovalIds = activeRequest.Select(a => a.ApproverId);
        var removedApproverIds = activeRequest.Where(a => !newApproverIds.Con-
tains(a.ApproverId)).Select(a => a.ApproverId);
        var addedApproverIds = newApproverIds.Where(i => !activeApprovalIds.Con-
tains(i));

        if (addedApproverIds.Any())
            daysOffProvider.CreateApprovals(addedApproverIds, activeRequest.Key);

        if (removedApproverIds.Any())
            daysOffProvider.DeleteApprovals(removedApproverIds, activeRequest.Key);
    }
}
}

```

Лістинг класу Worktime.cs

```

public class Worktime
{
    public Guid Id { get; set; }

    public Guid UserId { get; set; }

    public DateTime StartDate { get; set; }

    public DateTime? FinishDate { get; set; }

    public Guid? LastEditorId { get; set; }
}

```

Лістинг класу CalendarRule.cs

```

public class CalendarRule
{

```

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public Guid Id { get; set; }

public string Title { get; set; } = null!;

public CalendarRuleSetupType Type { get; set; }

public decimal? ShortDayDuration { get; set; }

public DateTime StartDate { get; set; }

public DateTime? FinishDate { get; set; }

public bool IsRecurring { get; set; }

public int RecurringFrequency { get; set; }

public CalendarRuleRecurringPeriod RecurringPeriod { get; set; }

public string? Exceptions { get; set; }
}

```

ЛІСТИНГ класу DayOffRequest.cs

```

public class DayOffRequest
{
    public Guid Id { get; set; }

    public Guid UserId { get; set; }

    public DateTime StartDate { get; set; }

    public DateTime FinishDate { get; set; }

    public DayOffReason Reason { get; set; }

    public IList<DayOffRequestApprovalResult> Approvals { get; set; } = new
List<DayOffRequestApprovalResult>();

    public bool IsEditable => StartDate.Date > DateTime.Now.Date;
}
}

```

ЛІСТИНГ класу QueriesWorktime.cs

```

public static class Worktime
{
    public const string CreateWorktimeRecord = @"
        INSERT INTO WorktimeRecords
        VALUES (@Id, @UserId, @StartDate, @FinishDate, @LastEditorId)
    ";

    public static string CreateWorktimeRecords(IEnumerable<Entities.Worktime> workti-
meRecords)
    {
        return $"INSERT INTO WorktimeRecords
        VALUES {string.Join(',',
        worktimeRecords.Select(item =>
        $"('{item.Id}', '{item.UserId}', '{item.StartDate}', '{item.FinishDate}', NULL)"))}";
    }

    public const string GetWorktimeRecordById = @"
        SELECT * FROM WorktimeRecords WHERE Id = @Id
    ";
}

```

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public const string GetUnfinishedWorktimeRecordByUserId = @"
    SELECT * FROM WorktimeRecords
    WHERE userId = @UserId AND FinishDate IS NULL
";

public static string GetWorktimeRecords(Sorting? sorting, WorktimeFilter? filter,
Paging? paging)
{
    string query = @"
        SELECT *
        FROM WorktimeRecords
        {AddFiltering(filter)}
    ";

    query += sorting != null ? $" {AddSorting(sorting)}" : "";
    query += paging != null ? $" {AddPaging(paging)}" : "";

    return query;
}

public const string UpdateFinishDate = @"
    DECLARE @UpdatedRecordId UNIQUEIDENTIFIER
    SET @UpdatedRecordId = (SELECT Id FROM WorktimeRecords WHERE userId = @UserId
AND finishDate IS NULL)

    UPDATE WorktimeRecords
    SET FinishDate = @FinishDate
    WHERE userId = @UserId AND finishDate IS NULL

    SELECT * FROM WorktimeRecords WHERE Id = @UpdatedRecordId
";

public const string UpdateWorktimeRecord = @"
    UPDATE WorktimeRecords SET
    StartDate = @StartDate,
    FinishDate = @FinishDate,
    LastEditorId = @LastEditorId
    WHERE Id = @Id
    ";

```

		Латаш А.А.			ККІКБ.КР.Б – 123 – 9 – ПЗ	Арк.
		Шелуха О.О.				93
Змн.	Арк.	№ докум.	Підпис	Дата		