Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра моделювання і програмного забезпечення

Пояснювальна записка
до кваліфікаційної роботи магістра
за спеціальністю 121 «Інженерія програмного забезпечення»

на тему: ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ХМАРНИХ ТЕХНОЛОГІЙ В ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ПРИКЛАДІ РОЗГОРТАННЯ ВЕБ-ОРІЄНТОВАНОЇ СОЦІАЛЬНОЇ МЕРЕЖІ

Проектував _____ А. В. Мусатов
Керівник роботи _____ О. В. Шамрай
Нормоконтроль _____ О. В. Шамрай
Завідувач кафедри _____ А. М. Стрюк

Кривий Ріг
2024

РЕФЕРАТ

Пояснювальна записка: 71 сторінка, 44 малюнки, 12 використаних джерела.

Об’єктом проектування є розгортання серверної частини соціальної мережі на базі платформи .NET.

Проект складається з трьох розділів.

Перший розділ присвячений опису особливостей проекту що треба разгорнути в хмарі та загальному опису хмарних технологій. Здійснено детальний огляд платформи, опис використовуваних архітектур веб-додатків та їх аналіз.

У другому розділі розкриваються питання розгортання та вибору хмарних технологій, розгортання бази даних.

Третій розділ присвячений опису фінансової частини обраної хмарної технології. Також було протестовано розгорнутий веб-додаток.

.NET, ASP.NET, Azure, Azure App Services, Azure SQL, SQL, БАЗА ДАНИХ, АВТОРИЗАЦІЯ\АВТЕНТИФІКАЦІЯ, JWT, ENTITY FRAMEWORK, SOLID.

Explanatory note: 71 pages, 44 figures, 12 sources used.

The design object is the deployment of the server part of a social network based on the .NET platform.

The project consists of three sections.

The first section is devoted to a description of the features of the project that needs to be deployed in the cloud and a general description of cloud technologies. A detailed overview of the platform, a description of the used web application architectures and their analysis are provided.

The second section reveals the issues of deployment and selection of cloud technologies, database deployment.

The third section is devoted to a description of the financial part of the selected cloud technology. The deployed web application was also tested.

.NET, ASP.NET, Azure, Azure App Services, Azure SQL, SQL, DATABASE, AUTHORIZATION\AUTHENTICATION, JWT, ENTITY FRAMEWORK, SOLID.

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;
СУБД – система управління базами даних;
СУРБД – систем керування реляційними базами даних;
ПЗ – програмне забезпечення;
API – Application Programming Interface (Інтерфейс прикладного програмування);
EF – Entity Framework;
DIP – Dependency Inversion Principle (принцип інве́рсії зале́жностей);
JWT – JSON Web Token, Веб-токен JSON;
SQL – Structured query language (мова структурованих запитів).
SaaS - Software-as-a-Service
PaaS - Platform-as-a-Service
IaaS - Infrastructure-as-a-Service

ВСТУП

Сучасний світ важко уявити без соціальних мереж – вони оточують нас усюди та виконують різноманітні функції. Ще століття тому важко було навіть уявити, що буде можливо обмінюватися інформацією без затримок та затрат. Соціальні мережі торкнулися абсолютно кожного аспекту нашого життя, саме тому важливим є робота над їх вдосконаленням та розумним користуванням.

Користувачі часто використовують соціальні мережі для висловлювання своїх думок. Саме тому розробка соціальної мережі є особливо актуальною в наш час. Під час війни в Україні дуже важливо тримати зв'язок зі своїми родичами.

У процесі планування і проектування мережі потрібно проробити БД і інфраструктуру веб-додатку. Це дуже важливо, бо навіть при скурпульозному аналізу потреб та вимог, БД не раз буде оновлюватися й перероблюватися.

Потенційними користувачами програмного забезпечення є особи всіх віків, які бажають використовувати веб-додаток для поширення своїх думок.

Також дуже важлива частина роботи – аналіз платформи .NET. Розбір архітектури та функціоналу що .NET надає. Проаналізувати методи, що надає платформа, для взаємодії з БД.

Реалізувати авторизацію і аутентифікацію за сучасними вимогами бізнесу. Проаналізувати сучасні методи і обрати найоптимальнішу.

Недостатня чіткість у розумінні та застосуванні хмарних технологій для розгортання та масштабування веб-орієнтованих соціальних мереж. Протиріччя Існує протиріччя між потребою в динамічному, масштабованому та економічному рішенні для розгортання веб-орієнтованих соціальних мереж та обмеженнями традиційних методів. Гіпотеза дослідження Впровадження хмарних технологій може суттєво покращити процес розгортання, масштабування та обслуговування веб-орієнтованих соціальних мереж, роблячи його більш динамічним, економічним та ефективним.

Ця проблема охоплює кілька ключових аспектів:

1. Технічні аспекти:
   - Інфраструктура: Вибір і налаштування хмарних платформ для забезпечення необхідних ресурсів (обчислювальних потужностей, зберігання даних, мережевих можливостей).
   - Архітектура додатку: Розробка архітектури програмного забезпечення, яка оптимально використовує можливості хмарних технологій, таких як мікросервіси, контейнеризація, серверлесс обчислення

- Масштабованість та продуктивність: Забезпечення можливості масштабування системи для обробки великої кількості одночасних користувачів та великих обсягів даних без втрати продуктивності.

2. Безпека та конфіденційність:
   - Захист даних: Впровадження методів і засобів для захисту особистих даних користувачів, запобігання витоку інформації та забезпечення безпеки зберігання даних у хмарі.
   - Дотримання стандартів: Відповідність хмарної інфраструктури і програмного забезпечення стандартам безпеки та регуляторним вимогам.
3. Економічні аспекти:
   - Вартість володіння: Аналіз витрат на впровадження, експлуатацію та підтримку хмарних рішень порівняно з традиційними підходами.
   - Ефективність ресурсів: Оптимізація використання хмарних ресурсів для зменшення витрат при збереженні високої продуктивності та надійності системи.
4. Організаційні аспекти:
   - Процеси розробки: Інтеграція хмарних технологій у процеси розробки програмного забезпечення, включаючи безперервну інтеграцію та безперервне розгортання (CI/CD).
   - Управління проектом: Координація роботи команд розробників, тестувальників та операторів для забезпечення ефективної співпраці у хмарному середовищі.

Таким чином, основна проблема полягає у комплексному дослідженні та розробці підходів для ефективного використання хмарних технологій у всіх аспектах створення та експлуатації веб-орієнтованих соціальних мереж. Це включає розв'язання технічних, безпечних, економічних та організаційних викликів для забезпечення успішного впровадження технологій.

1. Дослідження предметної частини і визначення цілей дослідження

1.1 Науковий апарат

Дослідження використання хмарних технологій в інженерії програмного забезпечення на прикладі розгортання веб-орієнтованої соціальної мережі. Очікується, що дане дослідження матиме значний вплив на розвиток інженерії програмного забезпечення соціальних мереж та сприятиме більш широкому впровадженню хмарних технологій в цій галузі.

Об'єкт дослідження: Процес розгортання, масштабування та обслуговування веб-орієнтованих соціальних мереж. Це включає в себе планування та проектування інфраструктури соціальної мережі. Розгортання програмного забезпечення та даних соціальної мережі. Масштабування інфраструктури для задоволення зростаючих потреб. Обслуговування та експлуатацію соціальної мережі. Моніторинг та оптимізацію продуктивності соціальної мережі. Забезпечення безпеки та надійності соціальної мережі. Предметом дослідження є використання хмарних технологій для покращення процесу розгортання, масштабування та обслуговування веб-орієнтованих соціальних мереж. Це є частиною вивчення різних хмарних платформ та технологій, таких як Infrastructure as a Service (IaaS), Platform as a Service (PaaS) та Software as a Service (SaaS). Аналіз переваг та недоліків використання хмарних технологій для соціальних мереж. Розробку рекомендацій щодо вибору оптимальних хмарних платформ та технологій для соціальних мереж. Створення прототипів та оцінку ефективності різних хмарних рішень для розгортання соціальних мереж. Розробку практичних рекомендацій щодо впровадження хмарних технологій в інженерії програмного забезпечення соціальних мереж.

Задачі дослідження використання хмарних технологій в інженерії програмного забезпечення на прикладі розгортання веб-орієнтованої соціальної мережі.

Дослідити теоретичні основи хмарних технологій, включаючи моделі обслуговування (IaaS, PaaS, SaaS), типи хмарних розгортань (приватні, публічні, гібридні) та ключові характеристики хмарних сервісів (масштабованість, еластичність, гнучкість, тощо). Проаналізувати архітектурні моделі та паттерни проектування, які використовуються для розгортання веб-орієнтованих соціальних мереж в хмарі. Дослідити проблеми та виклики, пов'язані з використанням хмарних технологій для соціальних мереж, такі як безпека, конфіденційність, надійність та відповідність нормативним вимогам.

Аналітичні дослідження: Провести порівняльний аналіз різних хмарних платформ та технологій, таких як Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP) та ін., з точки зору їх можливостей, характеристик та вартості для розгортання соціальних мереж. Вивчити кращі практики та досвід провідних компаній в розгортанні соціальних мереж в хмарі. Проаналізувати вплив хмарних технологій на ключові показники ефективності (KPI) соціальних мереж, такі як продуктивність, масштабованість, гнучкість та вартість володіння.

Експериментальні дослідження: Розробити прототипи хмарних рішень для розгортання веб-орієнтованих соціальних мереж. Оцінити ефективність різних хмарних рішень з точки зору продуктивності, масштабованості, гнучкості та вартості. Провести навантажувальні тестування та моделювання для оцінки стійкості та надійності хмарних рішень.

Розробчі дослідження: Розробити рекомендації щодо вибору оптимальних хмарних платформ та технологій для розгортання соціальних мереж з урахуванням їх специфічних потреб та вимог. Створити практичні посібники та інструменти для розробників соціальних мереж, які допоможуть їм впроваджувати хмарні технології у своїх проектах. Розробити рекомендації щодо забезпечення безпеки, конфіденційності та відповідності нормативним вимогам при розгортанні соціальних мереж в хмарі.

Узагальнення та впровадження результатів: Систематизувати та узагальнити отримані результати дослідження. Опублікувати наукові статті та доповіді за результатами дослідження. Презентувати результати дослідження на наукових конференціях та семінарах. Розробити план впровадження результатів дослідження в практику розробки та експлуатації соціальних мереж.

1.2 Дослідження особливості предметної галузі

Хмарні технології стали невід'ємною частиною сучасної інженерії програмного забезпечення, пропонуючи потужні інструменти та платформи для розробки, розгортання і масштабування веб-орієнтованих додатків. Соціальні мережі, які є одним із найбільш вимогливих до ресурсів видів веб-додатків, особливо виграють від впровадження хмарних рішень. Це дослідження зосереджується на вивченні специфічних аспектів використання хмарних технологій для розробки та експлуатації веб-орієнтованої соціальної мережі, розглядаючи технічні, безпекові, економічні та організаційні виклики. Технічні особливості:

- Хмарні платформи. Основні постачальники хмарних послуг (AWS, Google Cloud Platform, Microsoft Azure) пропонують різноманітні сервіси для обчислювальних ресурсів, зберігання даних і мережевої інфраструктури. Вибір платформи залежить від специфічних вимог проекту, таких як продуктивність, надійність і вартість.

- Контейнеризація та оркестрація. Використання Docker для контейнеризації додатків та Kubernetes для їх оркестрації дозволяє ефективно управляти розподіленими системами і забезпечувати їх масштабованість.
- Мікросервіси. Архітектура мікросервісів сприяє розділенню додатку на незалежні сервіси, які можуть бути розгорнуті та масштабовані окремо. Це особливо важливо для соціальних мереж, де різні компоненти (наприклад, обробка користувацьких даних, новинна стрічка, повідомлення) можуть мати різні вимоги до масштабування.
- Серверлесс обчислення. Використання функцій як сервісу (FaaS) дозволяє запускати окремі функції у відповідь на події, що забезпечує високу гнучкість і економічність у використанні ресурсів.
- Автоматичне масштабування. Хмарні платформи пропонують можливість автоматичного масштабування ресурсів залежно від навантаження. Це особливо важливо для соціальних мереж, де кількість одночасних користувачів може різко змінюватися.
- Оптимізація продуктивності. Використання інструментів для моніторингу і оптимізації продуктивності (наприклад, APM-систем) дозволяє виявляти та усувати вузькі місця в роботі додатку.

Безпека та конфіденційність

- Шифрування. Дані повинні бути зашифровані як при передачі, так і при зберіганні. Це забезпечує захист конфіденційної інформації користувачів від несанкціонованого доступу.
- Контроль доступу. Впровадження чітких політик контролю доступу для забезпечення того, що до даних мають доступ тільки авторизовані користувачі та сервіси.
- Регуляторні вимоги. Соціальні мережі повинні відповідати різним регуляторним вимогам, таким як GDPR в Європі або CCPA в Каліфорнії. Це включає управління персональними даними користувачів та забезпечення їх прав на конфіденційність.
- Безпекові стандарти. Дотримання галузевих стандартів безпеки, таких як ISO/IEC 27001, може допомогти в побудові надійної системи безпеки.

Економічні аспекти

- Моделі оплати. Хмарні платформи пропонують різні моделі оплати (pay-as-you-go, резервовані ресурси), які дозволяють оптимізувати витрати залежно від потреб проекту.
- Оцінка витрат. Аналіз та прогнозування витрат на використання хмарних ресурсів допомагає уникнути несподіваних витрат і оптимізувати бюджет проекту.
- Оптимізація використання. Впровадження практик оптимального використання ресурсів, таких як автоматичне вимкнення невикористовуваних ресурсів та оптимізація конфігурацій, дозволяє знизити загальні витрати.

- Використання спеціалізованих сервісів. Використання спеціалізованих хмарних сервісів для зберігання, обробки даних та інших функцій може бути більш економічно ефективним, ніж самостійна розробка аналогічних рішень.

Організаційні аспекти

- Безперервна інтеграція та розгортання (CI/CD). Впровадження CI/CD практик дозволяє забезпечити швидке і надійне розгортання нових версій додатку, мінімізуючи ризик виникнення помилок і простоїв.
- Агіл методології. Використання гнучких методологій розробки (Scrum, Kanban) сприяє ефективній координації команд та швидкому реагуванню на змінювані вимоги проекту.
- Координація команд. Забезпечення ефективної співпраці між різними командами (розробники, тестувальники, оператори) для досягнення загальних цілей проекту.
- Навчання та розвиток. Постійне навчання та розвиток навичок співробітників у сфері хмарних технологій для підтримки високого рівня професіоналізму та готовності до вирішення нових викликів.

Дослідження особливостей використання хмарних технологій для розгортання веб-орієнтованої соціальної мережі вимагає комплексного підходу, що охоплює технічні, безпекові, економічні та організаційні аспекти. Розуміння та вирішення цих питань дозволить створити ефективну, безпечну та економічно вигідну соціальну мережу, здатну задовольнити потреби користувачів і відповідати сучасним вимогам до якості програмного забезпечення.

1.3 Визначення вимог

Функціональні вимоги:

- Можливість створення нового облікового запису з верифікацією електронної пошти.
- Створення та редагування особистого профілю.
- Публікація текстових повідомлень.
- Можливість коментування та вподобання публікацій.

Нефункціональні вимоги:

- Система повинна підтримувати масштабованість для обслуговування великої кількості користувачів.
- Система повинна мати високу доступність, з мінімальними простоями.
- Час відгуку на запити користувачів повинен бути мінімальним.
- Інтуїтивно зрозумілий та зручний у використанні інтерфейс.
- Технічні вимоги
- Використання хмарних платформ (наприклад, AWS, Azure, Google Cloud) для розгортання додатків.

– Використання контейнеризації (наприклад, Docker, Kubernetes) для управління середовищем розгортання.
– Використання хмарних реляційних (наприклад, Amazon RDS, Google Cloud SQL) та нереляційних баз даних (наприклад, MongoDB Atlas).
– Використання RESTful API для інтеграції з іншими сервісами та клієнтськими додатками.

Вимоги до безпеки:

– Використання протоколів OAuth для аутентифікації користувачів.
– Захист даних користувачів шляхом шифрування.
– Захист даних в транзиті та на зберіганні за допомогою SSL/TLS.
– Регулярне резервне копіювання даних.
– Реалізація систем моніторингу безпеки та аудиту дій користувачів.

Додаткові вимоги:

– Використання Agile методології для гнучкого управління проектом.
– Використання сучасних фреймворків та бібліотек (наприклад, .NET).

Ці вимоги охоплюють основні аспекти, необхідні для дослідження та реалізації веб-орієнтованої соціальної мережі з використанням хмарних технологій в інженерії програмного забезпечення.

1.4 Моделювання БД за допомогою MS SQL Server

Сервер Microsoft SQL або більше відомий як сервер MS SQL — це сервер реляційної бази даних, розроблений корпорацією Microsoft. Сервер бази даних — це в основному програма бази даних, яка використовується для зберігання даних, а інші програмні програми отримують і зберігають дані за допомогою певної мови, яка називається SQL (мова структурованих запитів) у випадку з сервером MS SQL [1].

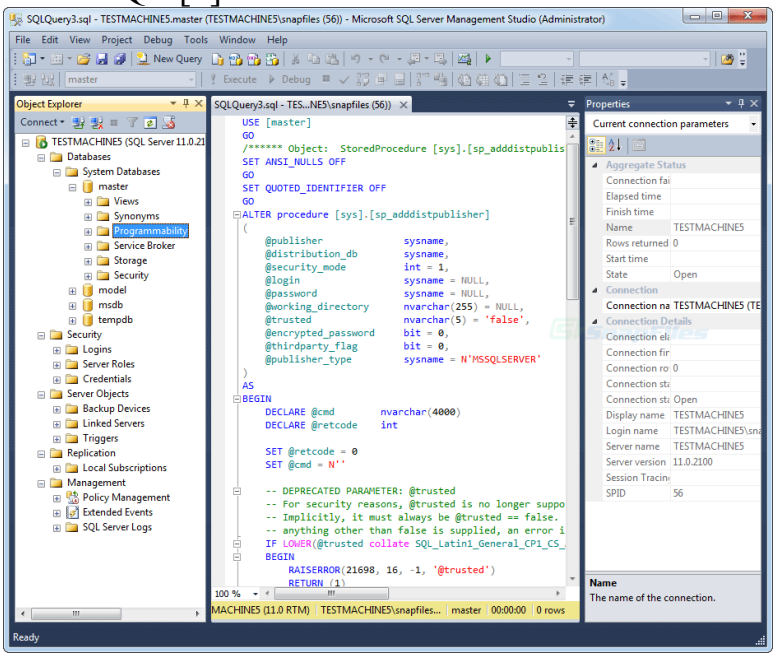


Рисунок 1.1 – Демонстрація Microsoft SQL Server

Основним інструментом інтерфейсу для MS SQL Server є SQL Server Management Studio (SSMS), який підтримує 64-розрядне та 32-розрядне операційне середовище. MS Server підтримує ANSI SQL. ANSI SQL є стандартною мовою структурованих запитів або SQL. Хоча MS SQL Server має власну програму мови SQL, тобто Transact-SQL або T-SQL.

Власна мова Microsoft, T-SQL, додатково надає можливості представлення збереженої процедури, обробки винятків, змінних тощо. Клієнтська програма, яка отримує доступ до даних із сервера бази даних, може бути на тому самому чи іншому комп'ютері та мати доступ до даних через набір протоколів та Інтернет. Корпорація Майкрософт розробила широкий вибір версій сервера MS SQL, зважаючи на тип аудиторії, яка використовує продукти сервера баз даних. Редакції MS SQL Server доступні для великих підприємств, середніх організацій і окремих осіб.

Його основне використання — забезпечення зв'язку з реляційними базами даних за допомогою мови програмування SQL.

За допомогою MS SQL Server ви можете надсилати запити до баз даних різними способами. Загалом, він використовує твердження на англійській мові. Крім того, він використовується для обробки даних і внутрішніх рішень для зберігання. Кілька найпопулярніших платформ використовують MS SQL Server, зокрема Microsoft, Accenture, Hepsiburada та Alibaba Travels. Крім того, він добре працює з програмами для відтворення музики та потокового передавання, а також мобільними банківськими програмами.

Крім того, будь-яка компанія, яка використовує реляційну базу даних, може захотіти використовувати запит MS SQL Server. Однак швидкість і ефективність запитів між різними платформами будуть різними. Наявність різних розширень є основною причиною різниці у швидкості. У будь-якому випадку MS SQL Server здатний виконувати дії, які включають ETL (створення, читання, оновлення та видалення) [2].

Інші способи використання MS SQL Server включають наступне.

- створення бази даних на основі конкретних специфікацій, виділених користувачем;
- після створення бази даних MS SQL Server може її підтримувати. Він також може підтримувати інші бази даних, створені іншими службами SQL;
- можно аналізувати та створювати звіти для сервера за допомогою відповідних параметрів;
- здатність виконувати операції ETL (вилучення, перетворення, завантаження) за допомогою певних функцій.

1.5 Моделювання архітектури додатку

Архітектура програми описує шаблони та методи, які використовуються для проектування та створення програми. Архітектура дає розробнику

дорожню карту та найкращі практики, яких слід дотримуватися під час створення програми, щоб клієнт отримав добре структуровану програму.

Шаблони проектування програмного забезпечення можуть допомогти клієнту створити програму. Шаблон описує повторюване рішення проблеми.

Патерни можна зв'язувати разом для створення більш загальних архітектур додатків. Замість того, щоб повністю створювати архітектуру самостійно, він може використовувати існуючі шаблони проектування, які також гарантують, що все працюватиме так, як повинно.

Як частина архітектури програми, існуватимуть як зовнішні, так і внутрішні служби. Фронтальна розробка пов'язана з користувальницьким досвідом роботи програми, а бек-енд розробка зосереджена на наданні доступу до даних, послуг та інших існуючих систем, які забезпечують роботу програми.

Архітектура є відправною точкою або дорожньою картою для створення програми, але розробнику потрібно буде зробити вибір реалізації, який не враховано в архітектурі. Наприклад, першим кроком є вибір мови програмування, на якій буде написано додаток.

Для розробки програмного забезпечення використовується багато мов програмування. Певні мови можуть використовуватися для створення певних типів додатків, наприклад Swift для мобільних додатків або JavaScript для інтерфейсної розробки.

JavaScript, який використовується з HTML і CSS, наразі є однією з найпопулярніших мов програмування для розробки веб-додатків.
Серед інших популярних мов програмування – Ruby, Python, Swift, TypeScript, Java, PHP і SQL. Мова, яка використовується під час створення програми, залежатиме від типу програми, доступних ресурсів розробки та вимог.

Історично склалося так, що програми писалися як єдиний код, де всі компоненти спільно використовують однакові ресурси та простір пам'яті. Цей стиль архітектури називають монолітним.

Сучасні архітектури додатків частіше слабко пов'язані, використовуючи мікросервіси та інтерфейси програмування додатків (API) для підключення служб, які забезпечують основу для хмарних програм.

Хмарна розробка — це спосіб пришвидшити створення нових програм, оптимізувати існуючі та забезпечити послідовну розробку й автоматизоване керування в приватних, загальнодоступних і гібридних хмарах [6].

Вирішуючи, яку архітектуру програми використовувати для нової програми, або оцінюючи поточну архітектуру, розробники починають із визначення своїх стратегічних цілей.

Тоді розробник зможе розробити архітектуру, яка підтримує його цілі, замість того, щоб спочатку вибирати архітектуру та намагатися зробити так, щоб програма відповідала цій структурі.

Розробник має продумати, як часто він хоче випускати оновлення, щоб задовольнити потреби клієнтів або операційні потреби, а також які функції вимагаються для бізнес-цілей або потреб розвитку.

Здатність швидко надавати клієнтам нові послуги та нові функції є одним із ключових конкурентних відмінностей, які може запропонувати компанія. А швидша розробка дозволяє компаніям частіше випускати нові функції та розгортати оновлення, щойно виявляється вразливість.

Існує багато різних типів архітектур додатків, але найпоширенішими сьогодні, заснованими на зв'язках між службами, є: моноліти та N-рівнева архітектура (тісно пов'язана), мікросервіси (роз'єднані), а також архітектура, керована подіями, та архітектура, орієнтована на послуги. (слабко з'єднані) [3].

Програми, які дотримуються принципу інверсії залежностей, а також принципів проектування, керованого доменом (DDD), як правило, мають схожу архітектуру. Протягом багатьох років ця архітектура мала багато імен. Однією з перших назв була Hexagonal Architecture, а потім Ports-and-Adapters. Зовсім недавно її називали цибулевою архітектурою або чистою архітектурою. Остання назва, Чиста архітектура, використовується як назва цієї архітектури в цій електронній книзі.

Довідкова програма eShopOnWeb використовує підхід Clean Architecture для організації коду в проекти. Ви можете знайти шаблон рішення, який можна використовувати як відправну точку для ваших власних рішень ASP.NET Core, у репозиторії ardalis/cleanarchitecture GitHub або встановивши шаблон із NuGet.

Чиста архітектура ставить бізнес-логіку та модель програми в центр програми. Замість того, щоб бізнес-логіка залежала від доступу до даних або інших питань інфраструктури, ця залежність інвертується: деталі інфраструктури та впровадження залежать від ядра програми. Ця функціональність досягається шляхом визначення абстракцій або інтерфейсів у прикладному ядрі, які потім реалізуються типами, визначеними на рівні інфраструктури. Поширеним способом візуалізації цієї архітектури є використання серії концентричних кіл, схожих на цибулю. На малюнку 5-7 показано приклад цього стилю архітектурного зображення.

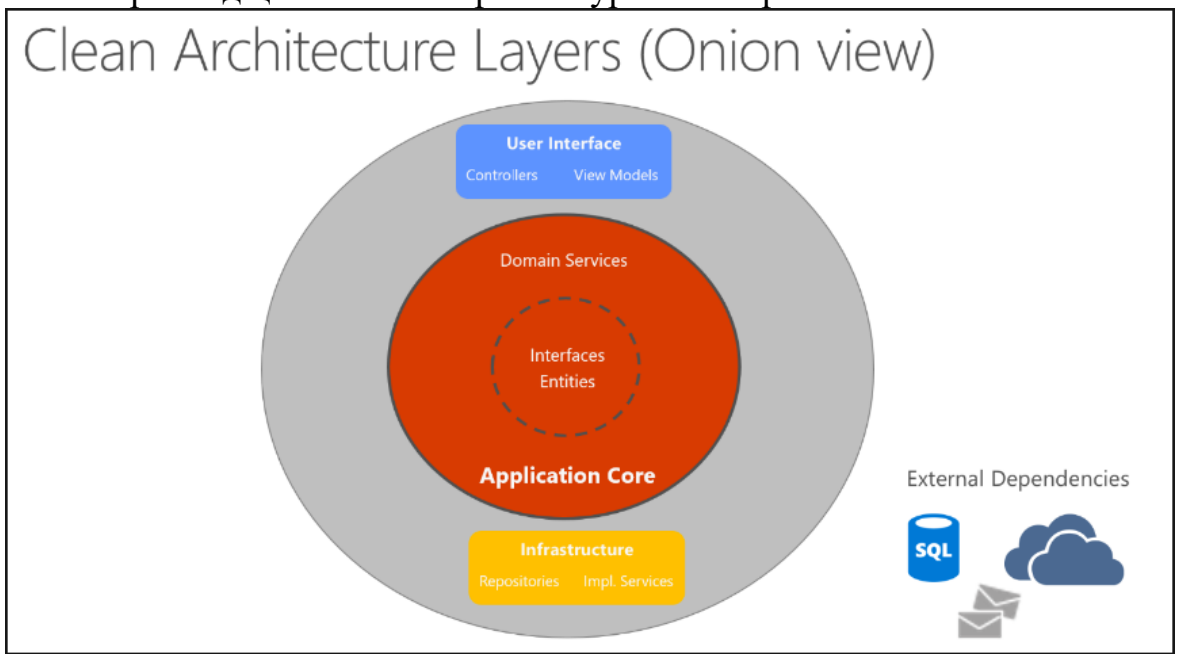


Рисунок 1.2– Демонстрація архітектури «Луковиця»

На цій діаграмі залежності прямують до самого внутрішнього кола. Ядро додатка отримав свою назву від свого положення в центрі цієї діаграми. І ви можете бачити на діаграмі, що Application Core не залежить від інших рівнів програми. Сутності та інтерфейси програми знаходяться в самому центрі. Зовні, але все ще в ядрі програми, знаходяться служби домену, які зазвичай реалізують інтерфейси, визначені у внутрішньому колі. За межами прикладного ядра інтерфейс користувача, і рівень інфраструктури залежать від прикладного ядра, але не один від одного (обов'язково).

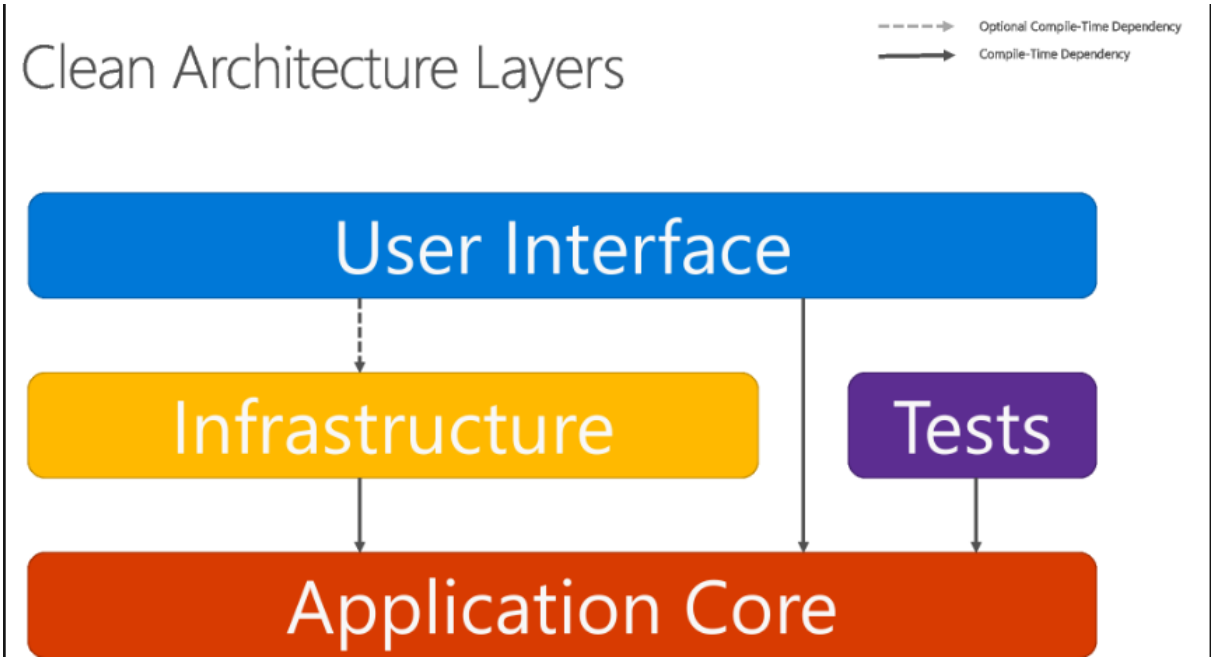


Рисунок 1.3 – Демонстрація слоїв «чистої архітектури»

Зауважте, що суцільні стрілки позначають залежності під час компіляції, тоді як пунктирна стрілка позначає залежності лише під час виконання. З чистою архітектурою рівень інтерфейсу користувача працює з інтерфейсами, визначеними в Application Core під час компіляції, і в ідеалі не повинен знати про типи реалізації, визначені на рівні інфраструктури. Однак під час виконання ці типи реалізації потрібні для виконання програми, тому вони повинні бути присутніми та підключені до інтерфейсів Application Core через впровадження залежностей [3].

Архітектурний шаблон Model-View-Controller (MVC) розділяє програму на три основні компоненти: модель, представлення та контролер. Структура ASP.NET MVC надає альтернативу шаблону ASP.NET Web Forms для створення веб-додатків на основі MVC. Фреймворк ASP.NET MVC — це легка структура презентацій, яка добре тестується, інтегрована з існуючими функціями ASP.NET, такими як головні сторінки та автентифікація на основі членства (як і програми на основі веб-форм). Структура MVC визначена в просторі імен System.Web.Mvc і є фундаментальною підтримуваною частиною простору імен System.Web [4].

MVC — це стандартний шаблон проектування, з яким знайомі багато розробників. Деякі типи веб-додатків виграють від інфраструктури MVC. Інші

продовжуватимуть використовувати традиційний шаблон програми ASP.NET, який базується на веб-формах і зворотній пересилці. Інші типи веб-додатків поєднуватимуть ці два підходи; жоден підхід не виключає інший.

Інфраструктура MVC включає наступні компоненти:

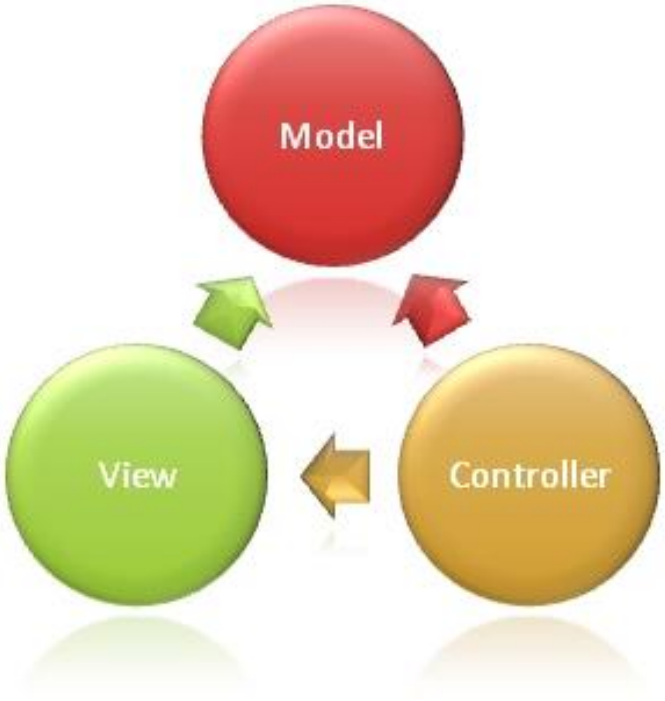


Рисунок 1.4 – Демонстрація принципу MVC

Об’єкти моделі – це частини програми, які реалізують логіку для домену даних програми. Часто об’єкти моделі отримують і зберігають стан моделі в базі даних. Наприклад, об’єкт Product може отримати інформацію з бази даних, працювати з нею, а потім записати оновлену інформацію назад до таблиці Products у SQL Server.

У невеликих програмах модель часто є концептуальним розділенням замість фізичного. Наприклад, якщо програма лише зчитує набір даних і надсилає його до представлення, програма не має рівня фізичної моделі та пов’язаних класів. У цьому випадку набір даних бере на себе роль модельного об’єкта.

Перегляди (View) — це компоненти, які відображають інтерфейс користувача програми (UI). Як правило, цей інтерфейс користувача створюється з даних моделі. Прикладом може бути подання редагування таблиці Products, яка відображає текстові поля, розкривні списки та прапорці на основі поточного стану об’єкта Products [4].

Контролери — це компоненти, які обробляють взаємодію з користувачем, працюють із моделлю та, зрештою, вибирають подання для візуалізації, яке відображає інтерфейс користувача. У програмі MVC перегляд відображає лише інформацію; контролер обробляє та реагує на введення та взаємодію користувача. Наприклад, контролер обробляє значення рядка запиту та передає ці значення моделі, яка, у свою чергу, запитує базу даних, використовуючи значення.

Шаблон MVC допомагає розробнику створювати програми, які розділяють різні аспекти програми (логіку введення, бізнес-логіку та логіку інтерфейсу користувача), водночас забезпечуючи слабкий зв’язок між цими елементами. Шаблон вказує, де кожен тип логіки повинен бути розташований у програмі. Логіка інтерфейсу користувача належить до представлення. Вхідна логіка належить до контролера. Бізнес-логіка належить до моделі. Це розділення допомагає керувати складністю під час створення програми, оскільки дає змогу зосередитися на одному аспекті реалізації за раз. Наприклад, ви можете зосередитися на поданні без залежності від бізнес-логіки.

На додаток до управління складністю, шаблон MVC полегшує тестування програм, ніж тестування веб-програми ASP.NET на основі веб-форм. Наприклад, у веб-додатку ASP.NET на основі веб-форм один клас використовується як для відображення вихідних даних, так і для відповіді на введення користувача. Написання автоматизованих тестів для додатків ASP.NET на основі веб-форм може бути складним, оскільки для тестування окремої сторінки ви повинні створити екземпляр класу сторінки, усіх його дочірніх елементів керування та додаткових залежних класів у додатку. Оскільки для запуску сторінки створюється дуже багато класів, може бути важко писати тести, які зосереджені виключно на окремих частинах програми. Тому тести для додатків ASP.NET на основі веб-форм можуть бути складнішими для впровадження, ніж тести в додатку MVC. Крім того, тести в програмі ASP.NET на основі веб-форм потребують веб-сервера. Фреймворк MVC роз’єднує компоненти та активно використовує інтерфейси, що дає змогу тестувати окремі компоненти окремо від решти фреймворку.

Слабкий зв’язок між трьома основними компонентами програми MVC також сприяє паралельній розробці. Наприклад, один розробник може працювати над представленням, другий розробник може працювати над логікою контролера, а третій розробник може зосередитися на бізнес-логіці в моделі.

Фреймворк ASP.NET MVC пропонує такі переваги [4]:

- це полегшує керування складністю, розділяючи програму на модель, представлення та контролер;
- не використовує форми стану перегляду або серверні форми. Це робить структуру MVC ідеальною для розробників, які хочуть повного контролю над поведінкою програми;
- використовує шаблон переднього контролера, який обробляє запити веб-додатків через один контролер. Це дає змогу розробити програму, яка підтримує розширену інфраструктуру маршрутизації. Щоб отримати додаткові відомості, перегляньте Front Controller на веб-сайті MSDN;
- забезпечує кращу підтримку розробки, керованої тестуванням (TDD);
- добре працює для веб-програм, які підтримуються великими командами розробників і веб-дизайнерів, яким потрібен високий ступінь контролю над поведінкою програми;

– переваги веб-програми на основі веб-форм.

Платформа на основі веб-форм пропонує такі переваги:

– підтримка модель подій, яка зберігає стан через HTTP, що сприяє розробці бізнес-додатків. Програма на основі веб-форм надає десятки подій, які підтримуються сотнями серверних елементів керування;
– використання шаблону Page Controller, який додає функціональність окремим сторінкам. Додаткову інформацію див. у розділі Page Controller на веб-сайті MSDN;
– використання форми стану перегляду або форми на основі сервера, що полегшує керування інформацією про стан;
– добре працює для невеликих груп веб-розробників і дизайнерів, які хочуть скористатися перевагами великої кількості доступних компонентів для швидкої розробки програм;
– загалом, це менш складно для розробки додатків, оскільки компоненти (клас Page, елементи керування тощо) тісно інтегровані та зазвичай потребують менше коду, ніж модель MVC.

1.6 Моделювання Авторизації/Автентифікації

Хоча автентифікація та авторизація часто використовуються як взаємозамінні, це окремі процеси, які використовуються для захисту організації від кібератак. Оскільки витоки даних продовжують зростати як за частотою, так і за масштабами, автентифікація та авторизація є першою лінією захисту, щоб запобігти потраплянню конфіденційних даних у чужі руки. Як наслідок, надійні методи автентифікації та авторизації мають бути важливою частиною загальної стратегії безпеки кожної організації.

Отже, різниця між автентифікацією та авторизацією. Автентифікація — це процес перевірки того, ким є хтось, тоді як авторизація — це процес перевірки, до яких конкретних програм, файлів і даних має доступ користувач. Ситуація схожа на ситуацію з авіакомпанією, якій потрібно визначити, хто з людей може сісти на борт. Перший крок — підтвердити особу пасажира, щоб переконатися, що він той, за кого себе видає. Після встановлення особи пасажира другим кроком є перевірка будь-яких спеціальних послуг, до яких пасажир має доступ, будь то політ першим класом або відвідування VIP-залу.

У цифровому світі автентифікація та авторизація досягають тих самих цілей. Автентифікація використовується для перевірки того, що користувачі дійсно є тими, за кого себе представляють. Після підтвердження авторизація використовується для надання користувачеві дозволу на доступ до різних рівнів інформації та виконання певних функцій залежно від правил, встановлених для різних типів користувачів.

Аутентифікація:

– автентифікація перевіряє, хто є користувачем;

- автентифікація здійснюється за допомогою паролів, одноразових PIN-кодів, біометричної інформації та іншої інформації, наданої або введеної користувачем;
- автентифікація є першим кроком ефективного процесу керування ідентифікацією та доступом;
- автентифікацію бачить користувач і частково змінює;
- приклад. Підтвердивши свою особу, співробітники можуть отримати доступ до програми кадрів (HR), яка містить особисту інформацію про зарплату, час відпустки та дані 401K.

Авторизація:

- авторизація визначає, до яких ресурсів може отримати доступ користувач;
- авторизація працює через налаштування, які впроваджує та підтримує організація;
- авторизація завжди відбувається після аутентифікації;
- користувач не бачить і не може змінити авторизацію;
- приклад: після авторизації рівня доступу співробітники та менеджери з персоналу можуть отримувати доступ до різних рівнів даних на основі дозволів, установлених організацією.

Загальні методи автентифікації

Хоча ідентифікаційні дані користувача історично перевірялися за допомогою комбінації імені користувача та пароля, сучасні методи автентифікації зазвичай спираються на три класи інформації:

- що ви знаєте: найчастіше це пароль. Але це також може бути відповідь на таємне запитання або одноразовий PIN-код, який надає користувачеві доступ лише до одного сеансу чи транзакції;
- що у вас є: це може бути мобільний пристрій або програма, маркер безпеки або цифрова ідентифікаційна картка;
- що ви: це біометричні дані, такі як відбиток пальця, сканування сітківки ока або розпізнавання обличчя.

Часто ці типи інформації поєднуються за допомогою кількох рівнів автентифікації. Наприклад, користувача можуть попросити ввести ім’я користувача та пароль, щоб завершити онлайн-покупку. Після підтвердження одноразовий PIN-код може бути надісланий на мобільний телефон користувача як другий рівень безпеки. Поєднуючи кілька методів автентифікації з узгодженими протоколами автентифікації, організації можуть забезпечити безпеку, а також сумісність між системами.

Загальні методи авторизації

Після автентифікації користувача застосовуються елементи керування авторизацією, щоб гарантувати, що користувачі можуть отримати доступ до потрібних їм даних і виконувати певні функції, наприклад додавати або видаляти інформацію, на основі дозволів, наданих організацією. Ці дозволи можна призначити на рівні програми, операційної системи або інфраструктури. Дві поширені методи авторизації включають:

Контроль доступу на основі ролей (RBAC): цей метод авторизації надає користувачам доступ до інформації на основі їх ролі в організації. Наприклад, усі співробітники компанії можуть переглядати, але не змінювати свою особисту інформацію, таку як зарплата, час відпустки та 401K даних. Тим не менш, менеджерам з персоналу (HR) може бути надано доступ до інформації про персонал усіх працівників із можливістю додавати, видаляти та змінювати ці дані. Призначаючи дозволи відповідно до ролі кожної особи, організації можуть забезпечити продуктивність кожного користувача, одночасно обмежуючи доступ до конфіденційної інформації.

Керування доступом на основі атрибутів (ABAC): ABAC надає користувачам дозволи на більш детальному рівні, ніж RBAC, використовуючи низку певних атрибутів. Це може включати такі атрибути користувача, як ім'я користувача, роль, організація, ідентифікатор і дозвіл безпеки. Він може містити такі атрибути середовища, як час доступу, місцезнаходження даних і поточні рівні організаційної загрози. Він може містити атрибути ресурсу, такі як власник ресурсу, ім'я файлу та рівень конфіденційності даних. ABAC — це більш складний процес авторизації, ніж RBAC, призначений для додаткового обмеження доступу. Наприклад, замість того, щоб дозволяти всім менеджерам з персоналу в організації змінювати кадрові дані співробітників, доступ можна обмежити певними географічними точками або годинами доби, щоб підтримувати жорсткі обмеження безпеки [5].

Надійна стратегія автентифікації та авторизації є важливою

Надійна стратегія безпеки вимагає захисту своїх ресурсів як за допомогою автентифікації, так і авторизації. Завдяки надійній стратегії автентифікації та авторизації організації можуть постійно перевіряти, хто є кожен користувач і до чого він має доступ, запобігаючи несанкціонованій діяльності, яка становить серйозну загрозу. Забезпечуючи належну ідентифікацію всіх користувачів і доступ лише до тих ресурсів, які їм потрібні, організації можуть максимізувати продуктивність, підвищуючи безпеку в той час, коли витоки даних позбавляють компаній їхніх доходів і репутації.

JWT або веб-токени JSON найчастіше використовуються для ідентифікації автентифікованого користувача. Вони видаються сервером автентифікації та споживаються клієнт-сервером (для захисту його API).

Веб-токен JSON — це відкритий галузевий стандарт, який використовується для обміну інформацією між двома об'єктами, як правило, клієнтом (наприклад, зовнішнім інтерфейсом програми) і сервером (сервером програми).

Вони містять об'єкти JSON, які містять інформацію, якою потрібно поділитися. Кожен JWT також підписується за допомогою криптографії (хешування), щоб гарантувати, що вміст JSON (також відомий як претензії JWT) не може бути змінений клієнтом або зловмисною стороною [5].

Наприклад, коли ви входите в обліковий запис Google, Google видає JWT, який містить такі претензії / корисне навантаження JSON:

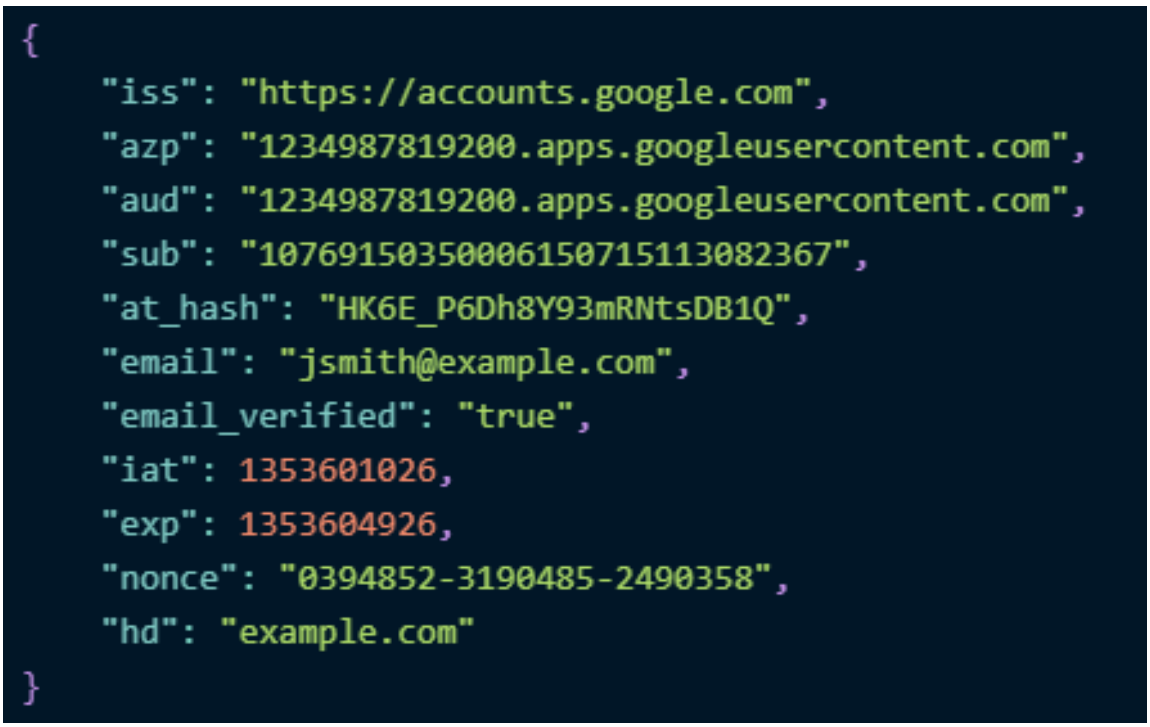


Рисунок 1.5 – Приклад JWT токену

Причина, чому сервер автентифікації не може просто надіслати інформацію як звичайний об’єкт JSON і чому йому потрібно перетворити її на «токен».

Якщо сервер автентифікації надсилає його як звичайний JSON, API клієнтської програми не матимуть можливості перевірити правильність вмісту, який вони отримують. Зловмисник може, наприклад, змінити ідентифікатор користувача (підзаявка у наведеному вище прикладі JSON), і API програми не зможуть дізнатися, що це сталося.

Через цю проблему безпеки сервер автентифікації повинен передавати цю інформацію таким чином, щоб її могла перевірити клієнтська програма, і тут з’являється концепція «токена».

Простіше кажучи, токен — це рядок, який містить деяку інформацію, яку можна безпечно перевірити. Це може бути випадковий набір буквено-цифрових символів, які вказують на ідентифікатор у базі даних, або це може бути закодований JSON, який клієнт може самостійно перевірити (відомий як JWT) [6].

JWT складається з трьох частин:

- заголовок, складається з двох частин;
- алгоритм підпису;
- тип токена, який в даному випадку здебільшого є «JWT»;
- корисне навантаження: корисне навантаження містить претензії або об’єкт JSON;
- підпис: Рядок, створений за допомогою криптографічного алгоритму, який можна використовувати для перевірки цілісності корисного навантаження JSON.

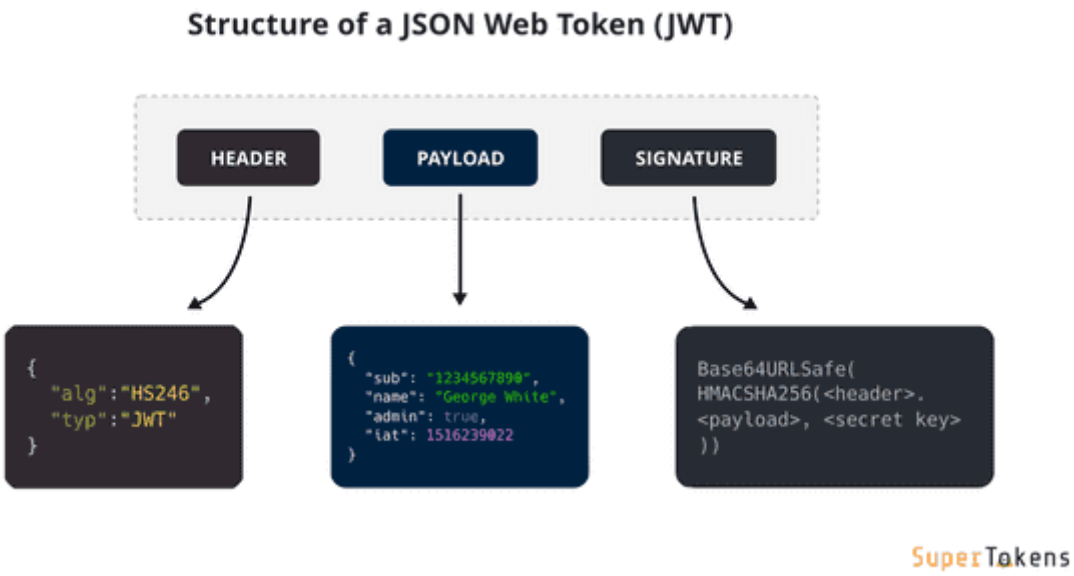


Рисунок 1.6 – Структура JWT токену

Найпростіший спосіб пояснити, як працює JWT, це на прикладі. Ми почнемо зі створення JWT для конкретного корисного навантаження JSON, а потім перевіримо його:

1. Створіть JSON

Візьмемо таке мінімальне корисне навантаження JSON:

```
{
    "userId": "abcd123",
    "expiry": 1646635611301
}
```

Рисунок 1.7 – Приклад JWT токену

2. Створіть ключ підпису JWT і виберіть алгоритм підпису

По-перше, нам потрібен ключ підпису та алгоритм для використання. Ми можемо створити ключ підпису за допомогою будь-якого захищеного випадкового джерела. Для цілей цієї публікації давайте використаємо:

– ключ підпису: NTNv7j0TuYARvmNMmWXo6fKvM4o6nv/aUi9ryX38ZH+L1bkrnD1ObOQ8JAUmHCBq7Iy7otZcyAagBLHVKvvYaIpmMuxmARQ97jUVG16Jkpkp1wXOPsrF9zwew6TpczyHkHgX5 EuLg2MeBuiT/qJACs1J0apruOOJCg/gOtkjB4c=
– алгоритм підпису: HMAC + SHA256, також відомий як HS256.

3. Створення «Заголовка»

Тут міститься інформація про те, який алгоритм підпису використовується. Як і корисне навантаження, це також JSON і буде додано до початку JWT (звідси заголовок назви):

Рисунок 1.8 – Заголовок JWT токену

4. Створіть підпис

– спочатку ми видаляємо всі пробіли з корисного навантаження JSON, а потім кодуємо його в base64, щоб отримати eyJ1c2VySWQiOiJhYmNkMTizIiwiZXhwaXJ5IjoxNjQ2NjM1NjExMzAxfQ. Ви можете спробувати вставити цей рядок у онлайн-декодер base64, щоб отримати наш JSON;
– так само ми видаляємо пробіли із заголовка JSON і кодуємо його в base64, щоб отримати: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9;
– ми об'єднуємо обидва базові 64 рядки з . посередині, наприклад <header>.<payload>, що дає нам eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQiOiJhYmNkMTizIiwiZXhwaXJ5IjoxNjQ2NjM1NjExMzAxfQ. Немає особливої причини робити це таким чином, окрім встановлення конвенції, якої галузь може дотримуватися;
– Тепер ми запускаємо функцію Base64 + HMACSHA256 на наведеному вище конкатенованому рядку та секреті, щоб дати нам підпис:

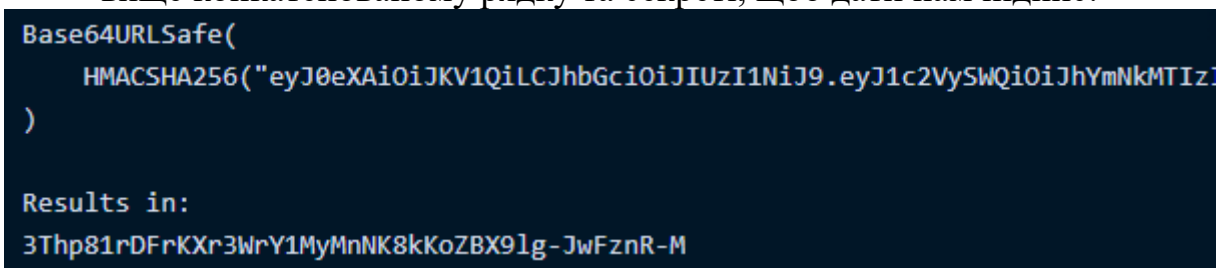
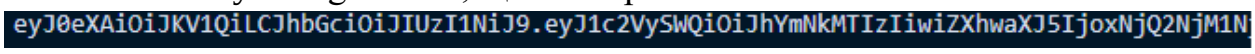


Рисунок 1.9 – Приклад підпису JWT токену

5. Створення JWT

Нарешті, ми додаємо згенерований підпис, наприклад <header>.<body>.<signature>, щоб створити наш JWT:



6. Перевірка JWT

Сервер автентифікації надішле JWT назад до інтерфейсу клієнта. Інтерфейс підключатиме JWT до мережевих запитів до рівня API клієнта. Рівень API виконає наступні кроки для перевірки JWT:

– отримує частину заголовка JWT (eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9);
– виконує декодування base64 для отримання простого тексту JSON: {"typ":"JWT","alg":"HS256"};
– перевіряє, чи значення поля typ — JWT, а alg — HS256. Якщо ні, JWT буде відхилено;

- отримує секретний ключ підпису та виконує ту саму операцію Base64URLSafe(HMACSHA256(...)) як крок номер (4) у заголовку та тілі вхідного JWT. Зауважте, що якщо тіло вхідного JWT відрізняється, цей крок створить інший підпис, ніж на кроці (4);
- перевіряє, чи згенерований підпис збігається з підписом із вхідного JWT. Якщо ні, тоді JWT відхиляється;
- ми base64 декодуємо тіло JWT (eyJ1c2VySWQiOiJhYmNkMTizIiwiZXhwaXJ5IjoxNjQ2NjM1NjExMzAxfQ), щоб отримати {"userId":"abcd123","expiry":1646635611301};
- ми відхиляємо JWT, якщо поточний час (у мілісекундах) перевищує термін дії JSON (оскільки термін дії JWT минув).

Ми можемо довіряти вхідному JWT, лише якщо він пройшов усі перевірки, наведені вище.

Використання JWT має чимало переваг [6]:

- безпечно: JWT мають цифровий підпис із використанням секретного (HMAC) або пари відкритих/приватних ключів (RSA або ECDSA), що захищає їх від зміни клієнтом або зловмисником;
- зберігається лише на клієнті: ви створюєте JWT на сервері та надсилаєте їх клієнту. Потім клієнт надсилає JWT з кожним запитом. Це економить місце в базі даних;
- ефективність / відсутність стану: JWT можна швидко перевірити, оскільки для цього не потрібен пошук у базі даних. Це особливо корисно у великих розподілених системах.

Однак деякі з недоліків:

- не підлягає відкликанню: через їх самодостатній характер і процес перевірки без збереження стану може бути важко відкликати JWT до закінчення терміну його дії. Тому такі дії, як миттєва заборона користувача, не можуть бути реалізовані легко. З огляду на це, існує спосіб підтримувати заборону/чорний список JWT, і завдяки цьому ми можемо негайно їх відкликати;
- залежить від одного секретного ключа: створення JWT залежить від одного секретного ключа. Якщо цей ключ зламано, зловмисник може створити власний JWT, який прийме рівень API. Це, у свою чергу, означає, що якщо секретний ключ скомпрометовано, зловмисник може підробити особу будь-якого користувача. Ми можемо зменшити цей ризик, час від часу змінюючи секретний ключ.

Підводячи підсумок, JWT є найбільш корисним для великомасштабних програм, які не потребують таких дій, як негайна заборона користувача.

Висновок до розділу

Проаналізована основна проблема обраної теми. Проаналізований ринок провайдерів хмарних сервісів. Опрацьовано науковий апарат та досліджено особливості предметної галузі. Визначені вимоги та промодельовано і прототиповано додаток який буде задеплоєний до хмари.

2 Поняття хмарних технологій

Хмарні обчислення – це надання обчислювальних послуг через Інтернет. Обчислювальні послуги включають загальну IT-інфраструктуру, наприклад віртуальні машини, сховище, бази даних та мережу. Хмарні послуги також розширюють традиційні IT-рішення, включаючи Інтернет речей (IoT), машинне навчання (ML) та штучний інтелект (AI).

Оскільки хмарні обчислення використовують Інтернет для доставки таких послуг, їм не доведеться обмежуватися фізичною інфраструктурою, як у традиційному центрі обробки даних. Це означає, що якщо потрібно швидко збільшити IT-інфраструктуру, то не доведеться чекати створення нового центру обробки даних. Можна використовувати хмару для швидкого розширення IT-ресурсів [7].

2.1 Опис моделі спільної відповідальності

Почнемо з традиційного корпоративного центру обробки даних. Компанія відповідає за обслуговування фізичного простору, забезпечення безпеки та обслуговування або заміну серверів у разі виходу з ладу. IT-відділ відповідає за обслуговування всієї інфраструктури та програмного забезпечення, необхідного для роботи центру обробки даних. Ймовірно, IT-відділ також відповідає за встановлення всіх виправлень та оновлення ПО до необхідної версії.

При роботі за моделлю спільної відповідальності ці обов’язки розподіляються між постачальником хмарних послуг та клієнтом. За фізичну безпеку, живлення, охолодження та мережеве підключення відповідає постачальник хмарних послуг. Споживач не прив’язаний до цього центру обробки даних, тому логічно, що ці обов’язки лежать не на ньому.

Водночас споживач відповідає за дані та інформацію, що зберігаються в хмарі. (споживач не захоче, щоб постачальник хмарних послуг міг прочитати його інформацію). Споживач також відповідає за безпеку доступу, тобто надає доступ тільки тим, хто потребує його.

Іноді обов’язки сторін залежать від ситуації. Якщо ви використовуєте хмарну базу даних SQL, постачальник хмарних послуг буде відповідати за її обслуговування. Однак ви все одно відповідаєте за дані, які в неї потрапляють. Якщо ви розгорнули віртуальну машину та встановили на ній базу даних SQL, ви будете відповідати за встановлення виправлень та оновлень в цій базі даних, а також за обслуговування даних та інформації, що зберігаються в базі

даних.

При використанні локального центру обробки даних ви відповідаєте за все. При використанні хмарних обчислень ці обов'язки змінюються. Модель спільної відповідальності тісно пов'язана з такими типами хмарних послуг (вони розглядаються далі в цій схемі навчання): інфраструктура як послуга (IaaS), платформа як послуга (PaaS) та програмне забезпечення як послуга (SaaS). IaaS покладає більшу частину відповідальності на споживача, а постачальник хмарних послуг відповідає за базові характеристики фізичної безпеки, живлення та підключення. На іншому кінці спектру — модель SaaS, в якій більша частина відповідальності покладається на постачальника хмарних послуг [9].

При використанні моделі PaaS — "золота середина" між моделями IaaS та SaaS — відповідальність рівномірно розподілена між постачальником хмарних послуг та споживачем.

На наступній схемі показано, як залежно від типу хмарної послуги розподіляється відповідальність за моделлю спільної відповідальності.

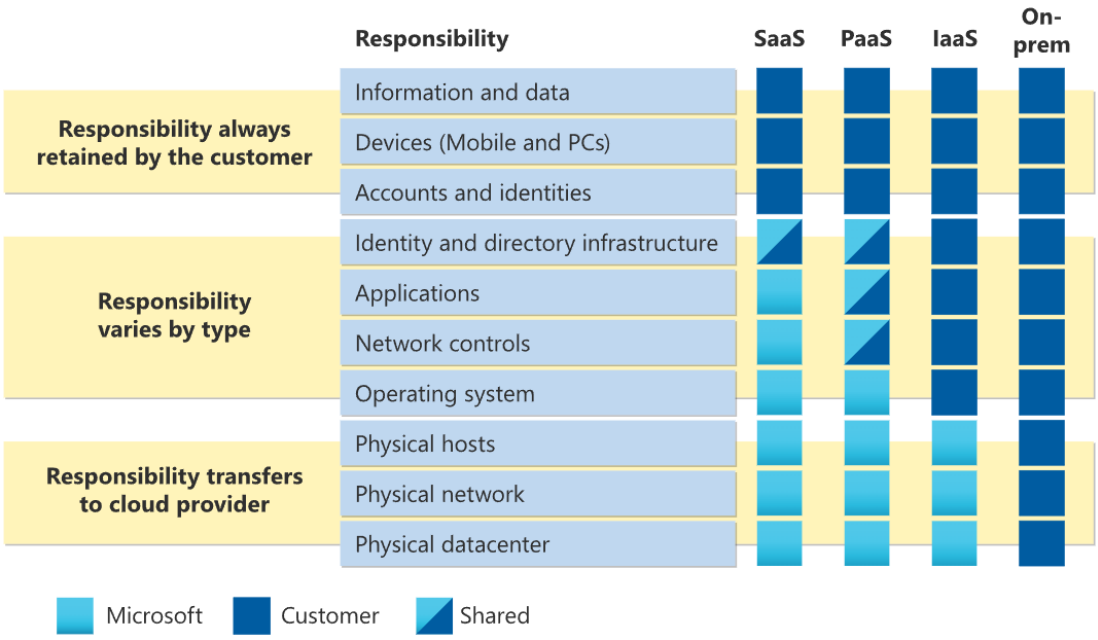


Рисунок 2.1 – Демонстрація Microsoft Azure послуг

Використовуючи постачальника хмарних служб, ви завжди будете відповідати за:

- Відомості та дані, що зберігаються у хмарі
- Пристрої, яким можна підключатися до хмари (мобільні телефони, комп'ютери тощо)
- Облікові записи та посвідчення користувачів, служб та пристроїв в організації

Постачальник хмарних служб завжди відповідає за таке:

- Фізичний центр обробки даних
- Фізична мережа
- Фізичні вузли

Модель обслуговування визначає відповідальність за такі речі, як:

- Операційні системи
- Елементи керування мережею
- Програми
- Ідентифікація та інфраструктура

2.2 Визначення хмарних моделей

Хмарні моделі визначають тип розгортання хмарних ресурсів. Три основні хмарні моделі: приватні, загальнодоступні та гібридні хмари.

Приватна хмара

По перше, приватна хмара. Приватна хмара — це, певною мірою, продукт природної еволюції корпоративного центру обробки даних. Це хмара (доставка IT-служб через Інтернет), яка використовується однією сутністю. Приватна хмара забезпечує набагато більший контроль над компанією та IT-відділом. Тим не менш, витрати в цьому випадку підвищені, а переваг загальнодоступної хмари менше. І, нарешті, приватну хмару можна розмістити з локального центру обробки даних. Його також можна розмістити у виділеному автономному центрі обробки даних; це може зробити навіть сторонній постачальник, який виділив цей центр обробки даних для вашої компанії [8].

Загальнодоступна хмара

Загальнодоступна хмара створюється, контролюється та обслуговується стороннім постачальником хмарних служб. З загальнодоступною хмарою будь-хто, хто хоче придбати хмарні служби, може отримати доступ до ресурсів і використовувати їх. Загальна доступність — це ключова різниця між загальнодоступними та приватними хмарами.

Гібридна хмара

Гібридна хмара – це обчислювальне середовище, яке використовує як загальнодоступні, так і приватні хмари у підключеному середовищі. Гібридне хмарне середовище можна використовувати для масштабування приватної хмари відповідно до тимчасових коливань попиту шляхом розгортання загальнодоступних хмарних ресурсів. Гібридна хмара може бути використана для забезпечення додаткового рівня безпеки. Наприклад, користувачі отримують гнучкі можливості для вибору служб для розміщення в загальнодоступній хмарі та їхнього розгортання в інфраструктурі приватної хмари.

Багатохмарні рішення

Четвертий сценарій, що набирає популярності, — це багатохмарне середовище. У багатохмарному середовищі використовується кілька постачальників загальнодоступних служб хмар. Можливо, клієнт використовує різні функції від різних постачальників хмарних служб. Або, можливо, він розпочав своє дослідження хмар з одним постачальником і в процесі дойшли до рішення іншого постачальника. У будь-якому випадку, у багатохмарному середовищі клієнт має справу з двома (або більше)

загальнодоступними постачальниками хмарних служб і керує ресурсами та безпекою в обох середовищах.

Azure Arc

Azure Arc – це набір технологій, які допомагають керувати хмарним середовищем. Azure Arc допомагає керувати хмарним середовищем, будь то загальнодоступна хмара виключно в Azure, приватна хмара в центрі обробки даних, гібридною конфігурацією або навіть багатохмарним середовищем, запущеним відразу на декількох постачальниках хмарних служб.

Рішення Azure VMware

Це рішення існує якщо клієнт вже встановив vMware в приватній хмарі, але хочете перенести його в загальнодоступну або гібридну хмару. Рішення Azure VMware дозволяє запускати робочі навантаження VMware в Azure за допомогою простої інтеграції та масштабованості [8].

2.3 Опис моделі, що базується на споживанні

При порівнянні моделей IT-інфраструктури можна виділити два типи витрат: капітальні (CapEx) та операційні (OpEx).

Капітальні витрати - це, як правило, одноразові попередні витрати на придбання матеріальних ресурсів. Прикладами капітальних витрат є витрати на будівництво нової будівлі, ремонт паркування, будівництво центру обробки даних або придбання автомобіля компанії.

А операційні витрати, навпаки, — це витрати на послуги чи продукти протягом певного періоду. Оренда конференц-центру, прокат автомобіля компанії або передплата хмарних служб — усе це приклади операційних витрат.

Оплата хмарних обчислень відноситься до операційних витрат, так як хмарні обчислення виконуються за моделлю на основі споживання. При використанні хмарних обчислень ви не платите за фізичну інфраструктуру, електрику, безпеку або будь-що інше, пов'язане з обслуговуванням центру обробки даних. Ви платите лише за IT-ресурси, що споживаються. Якщо цього місяця ви не користувалися IT-ресурсами, ви не платите за них.

Така модель на основі споживання має безліч переваг, включаючи такі:

- Жодних початкових витрат.
- Немає необхідності набувати дорогої інфраструктури, потенціал якої, можливо, не буде реалізований користувачами повною мірою, та керувати нею.
- Можливість оплачувати додаткові ресурси у разі потреби у них.
- Можливість припинити оплачувати ресурси, які більше не потрібні.

У традиційному центрі обробки даних потрібно прогнозувати майбутню потребу у ресурсах. Якщо ви переоціните цю потребу, ви витратите на свій центр обробки даних більше, ніж потрібно, і можливо втратите гроші. Якщо ви недооцінюєте цю потребу, ваш центр обробки даних швидко досягне ліміту

ресурсів і ваші програми та служби можуть постраждати від зниження продуктивності. Вирішення цієї проблеми може зайняти тривалий час. Можливо, потрібно замовити, отримати та встановити більше обладнання. Також потрібно додати живлення, охолодження та мережеві ресурси для додаткового обладнання.

При використанні хмарної моделі прогнозувати потребу в ресурсах так точно не потрібно. Якщо розробник вияве, що йому потрібно більше віртуальних машин, він додасть їх. Якщо попит на ресурси знизиться і йому не потрібно стільки віртуальних машин, він їх видалить. У будь-якому випадку клієнт платить лише за віртуальні машини, що використовуються, а не за додаткову ємність, яка є у постачальника хмарних служб.

2.4 Порівняння моделей ціноутворення для хмари

Хмарні обчислення – це надання обчислювальних служб через Інтернет за допомогою цінової моделі з оплатою у міру використання. Зазвичай ви платите тільки за хмарні служби, що використовуються. Такий підхід дозволяє:

- планувати та контролювати операційні витрати клієнатам;
- ефективно обслуговувати інфраструктуру;
- змінювати масштаб залежно від потреб бізнесу.

Іншими словами, хмарні обчислення – це спосіб орендувати обчислювальні ресурси та сховища у сторонньому центрі обробки даних. Поставтеся до хмарних ресурсів так само, як до ресурсів свого центру обробки даних. Однак, на відміну від власного центру обробки даних, коли ви закінчите використовувати хмарні ресурси, ви повернете їх назад. Рахунки виставляються лише за використані ресурси.

Замість використання фізичних ЦП та сховища у своєму центрі обробки даних, клієнт орендує їх на потрібний йому час. Всі турботи, пов'язані з підтримкою його інфраструктури, беруть на себе постачальник хмари. Хмара дозволяє швидко вирішувати складні бізнес-завдання та пропонує користувачам найсучасніші рішення.

2.5 Опис ринку провайдерів хмарних платформ

За даними Synergy Research Group, світовий ринок хмарних послуг продовжує демонструвати стійке зростання, незважаючи на деяке уповільнення темпів у четвертому кварталі 2022 року. Витрати підприємств на послуги хмарної інфраструктури перевищили 61 мільярд доларів США, що свідчить про високий попит на такі рішення. Однак, порівняно з попереднім кварталом, темпи зростання знизилися, що може бути пов'язано з низкою факторів, включаючи глобальну економічну нестабільність, зміни валютних курсів та насичення ринку.

Історично сильний долар США та обмеження китайського ринку значно вплинули на загальну динаміку зростання. Однак, якщо розглядати ринок США окремо, то темпи зростання виявилися більш стійкими, що підтверджує високий попит на хмарні послуги в цій країні [10].

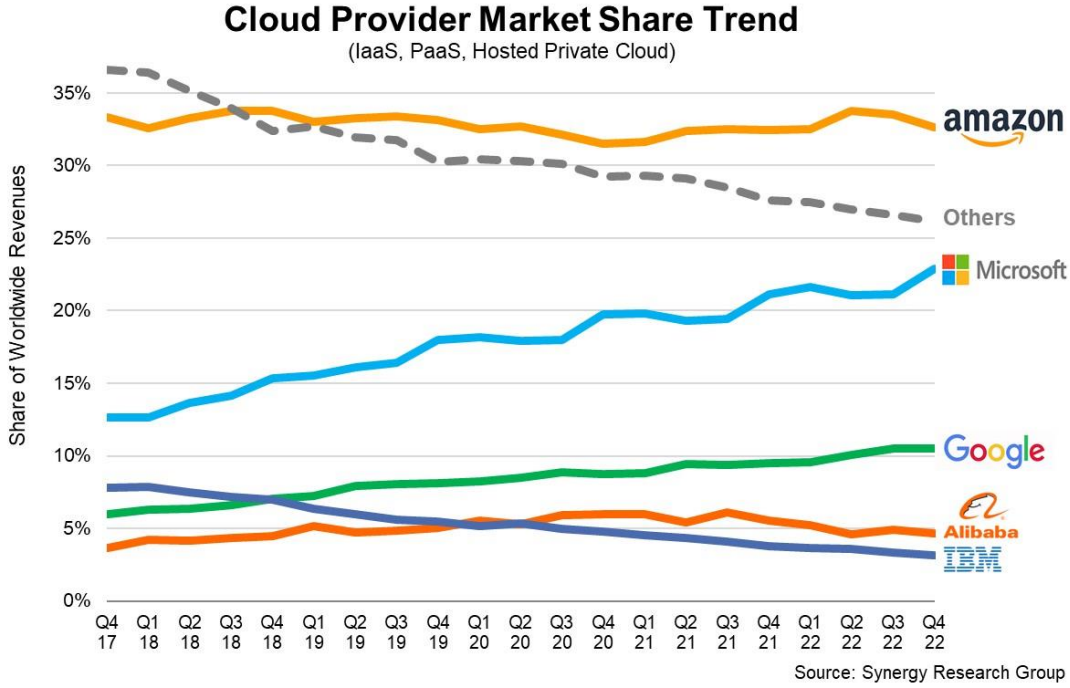


Рисунок 2.2 – ринок провайдерів хмарних технологій з 17 по 22 рік

Серед провідних постачальників хмарних послуг особливо виділяється Microsoft, яка демонструє значне зростання своєї ринкової частки. Amazon, як і раніше, залишається лідером ринку, тоді як Google також продовжує нарощувати свою присутність. Варто зазначити, що трійка лідерів контролює значну частку світового ринку, що свідчить про високу концентрацію.

Публічні послуги IaaS і PaaS залишаються основною рушійною силою ринку. Їх частка в загальному обсязі доходів продовжує зростати, що підтверджує тенденцію до міграції підприємств в хмару.

Хмарний ринок продовжує активно розвиватися в усіх регіонах світу, хоча темпи зростання можуть відрізнятися. США залишається найбільшим і найдинамічнішим ринком, але інші регіони також демонструють значний потенціал зростання.

Незважаючи на деяке уповільнення темпів зростання в четвертому кварталі 2022 року, Synergy Research Group прогнозує, що світовий ринок хмарних послуг продовжить демонструвати стійку динаміку в довгостроковій перспективі. З поліпшенням глобальної економічної ситуації та стабілізацією валютних ринків очікується прискорення темпів зростання.

Ринок хмарних послуг проходить період трансформації, пов'язаний з впливом макроекономічних факторів, зміною поведінки споживачів та посиленням конкуренції. Незважаючи на ці виклики, ринок залишається привабливим для інвесторів і продовжує демонструвати високий потенціал зростання.

Ключові тенденції:

– Зростання витрат на хмарні послуги
– Уповільнення темпів зростання в деяких регіонах
– Зміни в розподілі ринкової частки між провідними постачальниками
– Зростання попиту на публічні послуги IaaS і PaaS
– Географічна диверсифікація ринку

Canalys переглянула свої історичні оцінки доходів від послуг хмарної інфраструктури після всебічного перегляду визначень, категорій послуг і вихідних даних. Це безпосередньо вплинуло на частки ринку, звітовані з 1 кварталу 2022 року по 1 квартал 2024 року, але рейтинг і темпи зростання залишаються незмінними [10].

Worldwide cloud infrastructure services market

Market shares from Q1 2022 to Q2 2024

	Amazon Web Services	Microsoft Azure	Google Cloud	Others
Q1 2022	34%	17%	8%	41%
Q2 2022	34%	17%	9%	41%
Q3 2022	35%	16%	9%	40%
Q4 2022	35%	17%	9%	38%
Q1 2023	34%	19%	9%	39%
Q2 2023	33%	18%	9%	39%
Q3 2023	33%	18%	9%	39%
Q4 2023	34%	19%	10%	38%
Q1 2024	33%	21%	10%	37%
Q2 2024	33%	20%	10%	37%

Note: percentages may not add up to 100% due to rounding

Source: Canalys estimates, August 2024

Рисунок 2.3 – ринок провайдерів хмарних технологій з 21 по 24 рік

Очікується, що зростаючий попит на штучний інтелект стане потужним стимулом для розвитку хмарних сервісів. Запровадження технологій ШІ вимагає значних обчислювальних ресурсів та масштабованої інфраструктури, що, в свою чергу, стимулює інвестиції в хмарні рішення. Незважаючи на деякі побоювання щодо надмірного інвестування в ШІ, гіперскейлери продовжують збільшувати свої капітальні витрати, розуміючи, що потенційні ризики можуть бути менш згубними, ніж пропущені можливості.

Всі три провідні гіперскейлери демонструють значну активність у сфері штучного інтелекту. Вони активно інвестують у розширення своїх центрів обробки даних, розробку нових продуктів на основі ШІ та зміцнення партнерських екосистем. Запуск таких інноваційних продуктів, як Gemini 1.5 від Google Cloud та GPT-4o mini від Azure, свідчить про прагнення гіперскейлерів задовольнити зростаючий попит на рішення штучного інтелекту [11].

Гіперскейлери також активно співпрацюють зі стартапами, що спеціалізуються на штучному інтелекті. Ці партнерства дозволяють гіперскейлерам отримати доступ до нових технологій, талантів та ринків. Запуск таких ініціатив, як Generative AI Accelerator від AWS та Google Cloud для стартапів від Google Cloud, свідчить про розуміння гіперскейлерами важливості підтримки інноваційних компаній.

Ключові фактори, що впливають на розвиток ринку:

- зростаючий попит на ШІ: Штучний інтелект стає все більш інтегрованим у різні сфери діяльності, що створює високий попит на обчислювальні ресурси.
- інвестиції гіперскейлерів: Значні інвестиції гіперскейлерів у розвиток інфраструктури та розробку нових продуктів на основі ШІ забезпечують стійкий розвиток ринку.
- партнерство з стартапами: Співпраця з стартапами дозволяє гіперскейлерам прискорити інновації та вийти на нові ринки.
- конкуренція: Конкуренція між гіперскейлерами стимулює розвиток нових продуктів і послуг, що сприяє зниженню цін і підвищенню якості.

Взаємодія штучного інтелекту та хмарних сервісів створює синергію, яка прискорює цифрову трансформацію бізнесу. Гіперскейлери, інвестуючи в розвиток ШІ та співпрацюючи зі стартапами, формують нову екосистему, яка відкриває широкі можливості для інновацій і зростання.

Висновок до розділу

У даній роботіі була розглянута розробка додатку. По-перше, обрана СУБД. Проаналізовані сильні та слабкі стороні MS SQL СУБД. По-друге, проаналізовано сучасні типи архітектури веб-додатків. Було детально проаналізовано поняття «чиста архітектура» та MVC. По-третє, детально розглянуто поняття Авторизації/Автентифікації, способи реалізації. Слабкі та сильні сторони JWT.

3. Процес розгортання веб додатку

3.1 Вибір провайдеру хмарних технологій

Серед усіх хмарних платформ було обрано Azure Microsoft. Ця платформа чудово підходить для мого проекту через глибоку інтеграцію та сумісність з технологіями від Microsoft. Більш того, Azure була розроблена з урахуванням .NET. Це значно спрощує розгортання і керування. Також, Azure має пряму інтеграцію з основною IDE платформи .NET - Visual Studio. Також, Azure підтримує як .NET Framework , так і .NET Core. Це дає розробникам гнучкість у виборі стеку технологій.

Azure пропонує своїм клієнтам широкий спектр служб .NET служб. Вони задовольняють різноманітні потреби користувачів. Azure App Service – повнісю керована платформа для створення, розгартання та машстабування веб додатків і інших серверних програм. Azure Functions дозволяють розробникам швидко запускати код не турбуючись за керування інфраструктурою пропонуючи безсерверне (serverless) обчислювальне рішення. Бази даних Azure SQL і Azure Cosmos DB надають розробникам надійні варіанти баз для SQL і NoSQL рішень.

Можливості хмарної платформи Azure пропонує клієнтам ефективно використовувати локальні і хмарні ресурси. Azure Stack надає функції Azure локально і дозволяє розробникам розширити вже існуючу інфраструктуру. За допомогою Azure Arc можливо узгоджено керувати локальними, переферійними та мультихмарними середовищами.

Azure надає комплексну екосистему, яка підтримує розробку .NET. Azure DevOps – хмарна платформа для планування, стфорення, тестування та розгортання програм. Azure Monitor дає можливість розробникам відстежувати виконання коду, працездатність і продуктивність. як Azure Security Center надає розширений захист від загроз і рекомендації щодо безпеки.

Масштабованість і гнучкість Azure роблять його придатною платформою для широкого спектру програм. Його можливості автоматичного масштабування забезпечують оптимальну продуктивність і економічну ефективність шляхом автоматичного коригування ресурсів на основі попиту. Глобальне охоплення Azure з центрами обробки даних по всьому світу дозволяє розробникам розгортати програми ближче до своїх користувачів і зменшувати затримку.

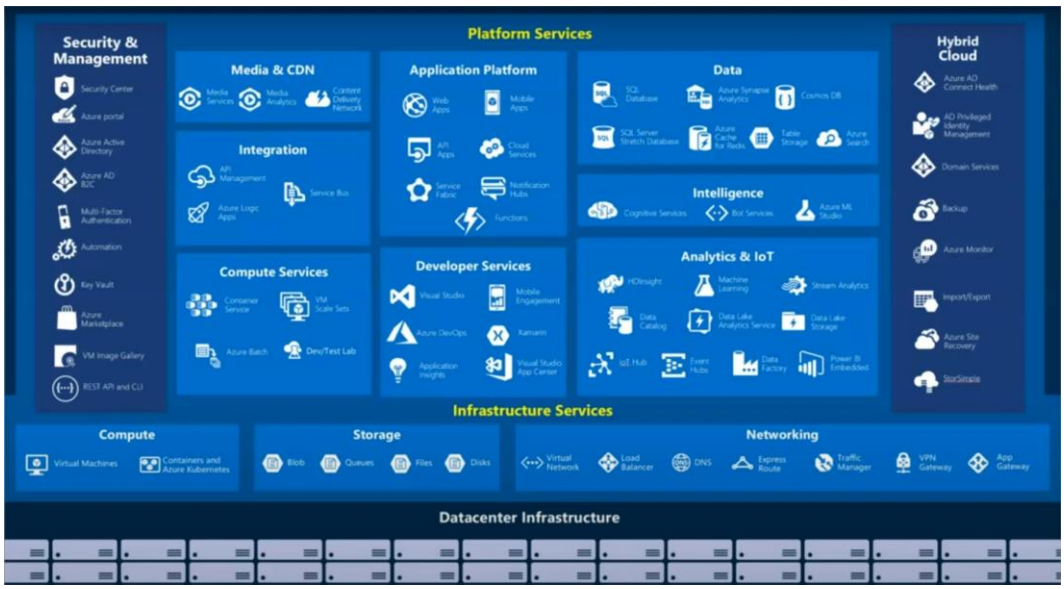


Рисунок 3.1 – схематичне зображення сервісів Azure

Підсумовуючи, слід зазначити, що глибока інтеграція Azure із .NET, розширені специфічні для .NET служби, можливості гібридної хмари, комплексний набір інструментів і масштабованість роблять його чудовим вибором для розробників, які створюють програми .NET.

3.2 Azure Free Services

Безкоштовні послуги Azure призначені для зниження вартості інфраструктури хмарних обчислень для малого та середнього бізнесу. Він пропонує клієнтам безкоштовні послуги рівня для тестування нових програм і оцінки переваг хмарних обчислень.

Коли користувач починає використовувати Azure із безкоштовним обліковим записом, він отримує кредит у розмірі 200 доларів США, який можна витратити протягом перших 30 днів після реєстрації. Крім того, отримується безкоштовні щомісячні суми двох груп послуг: популярних послуг, які безкоштовні протягом 12 місяців, і більше 40 інших послуг, які безкоштовні завжди.

Буде надано:

- €170 або $200 безкоштовний кредит протягом першого місяця. Після закінчення першого місяця термін дії кредиту закінчується, і буде запропоновано вибрати одну з доступних підписок
- 12 місяців популярних сервісів безкоштовно.
- 25 послуг самостійно безкоштовно. Пропозиція може відрізнятися в різних регіонах.

3.3 Безкоштовний обліковий запис Azure з обмеженим часом

Обмежений безкоштовний обліковий запис Azure – це чудова пропозиція спробувати рішення хмарних служб Microsoft. Популярні служби

безкоштовні протягом 12 місяців, а потім можливо користуватися ними зі знижкою 50% з кредитом Azure. Починаючи з 200 доларів США кредиту Azure, можливо створювати, розгортати та масштабувати свої програми на різних платформах, включаючи веб-програми, мобільні серверні програми та рішення для великих даних.

Безкоштовно протягом 12 місяців — це пропозиція з обмеженим часом, за допомогою якої ви можете спробувати платні послуги Azure без необхідності платити. За цією пропозицією можна користуватися будь-якою з понад 40 послуг Azure безкоштовно протягом 12 місяців і отримати 200 доларів США у своєму першому рахунку.

3.4 Огляд Azure Portal

Портал Azure представляє собою єдину інтегровану платформу, що забезпечує централізований доступ до множини хмарних сервісів. За допомогою цього інструменту користувачі можуть здійснювати повний цикл управління програмним забезпеченням: від розробки та розгортання до моніторингу та масштабування. Портал надає зручний графічний інтерфейс, який дозволяє ефективно взаємодіяти з різноманітними хмарними рішеннями, починаючи від простих веб-додатків і закінчуючи складними розподіленими системами.

Для отримання доступу до можливостей платформи Azure необхідно пройти процедуру реєстрації. Новим користувачам надається значний пакет початкових переваг, включаючи:

- безкоштовний доступ до базового набору сервісів протягом перших 12 місяців використання.
- кредит на певну суму для оплати додаткових ресурсів, який діє протягом 30 днів з моменту реєстрації. Сума кредиту може варіюватися залежно від регіону.
- безкоштовний доступ до деяких сервісів на постійній основі.

Перед реєстрацією рекомендується детально ознайомитися з переліком доступних сервісів та спланувати їх використання з урахуванням тимчасових обмежень безкоштовного кредиту. Це дозволить максимально ефективно використати надані можливості.

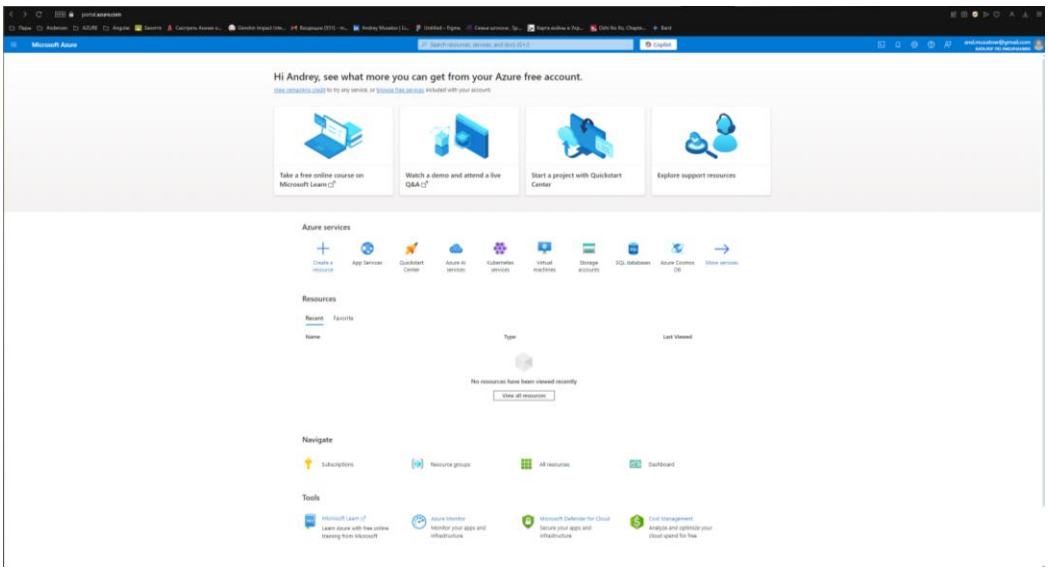


Рисунок 3.2 – Azure Portal

3.5 Azure Підписки (Subscriptions)

Підписка Azure являє собою самостійну ізольовану область всередині хмарної платформи, яка асоціюється з конкретним обліковим записом користувача. Вона функціонує як логічна одиниця, що об'єднує сукупність хмарних ресурсів, які можуть бути використані в рамках даної підписки. Наявність активної підписки є обов'язковою умовою для доступу до широкого спектру хмарних послуг, наданих компанією Microsoft Azure.

Кожна підписка Azure однозначно прив'язана до конкретного облікового запису користувача, який ініціював її створення. Цей обліковий запис використовується як для аутентифікації доступу до підписки, так і для здійснення фінансових операцій, пов'язаних з її використанням. В межах однієї підписки можуть бути розгорнуті різноманітні хмарні сервіси та ресурси, що забезпечують гнучкість та масштабованість розв'язань.

Для більшої зручності управління та обліку витрат передбачена можливість створення декількох підписок Azure. Кожна з них веде самостійну історію використання ресурсів та формує окремий рахунок. Даний підхід дозволяє розділити відповідальність за різні проекти або департаменти всередині організації.

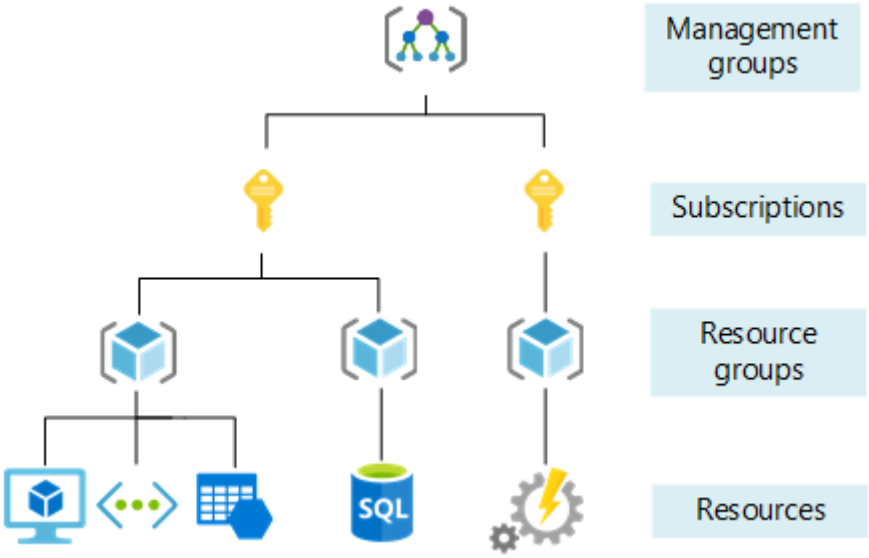


Рисунок 3.3 – ієрархія сутностей Azure

Ініціатор створення підписки автоматично набуває статусу глобального адміністратора, отримуючи найвищий рівень доступу до всіх ресурсів та налаштувань даної підписки. Механізм створення множинних підписок може бути ефективним інструментом для делегування повноважень та розподілу відповідальності за управління різними хмарними середовищами всередині організації.

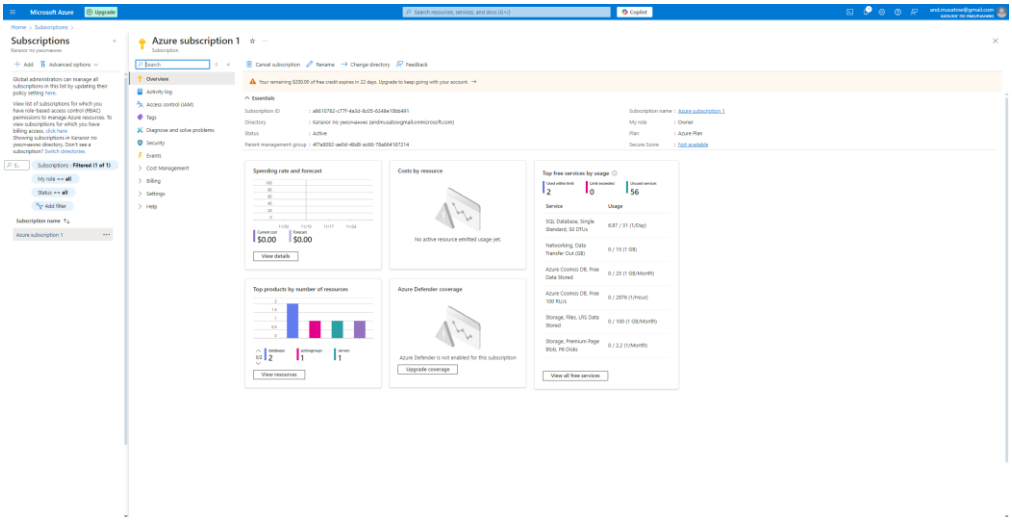


Рисунок 3.4 – Azure Subscriptions

3.6 Групи ресурсів (Resource Group) та Ресурса (Resource)

Портал містить не лише послуги, які надає Azure, але й послуги, що надаються сторонніми постачальниками на платформі Azure. Вони використовували ЦП або віртуальні машини Azure і розгортали на ньому свою платформу та пропонували цю платформу як Послугу на основі оплати по мірі використання.

Група ресурсів: контейнер, який містить пов’язані ресурси для рішення Azure. Він може містити всі ресурси для рішення або містити лише ті ресурси, якими потрібно керувати як групою. Групи ресурсів — це контейнери

ресурсів, які мають спільний життєвий цикл або спільний атрибут, наприклад «усі сервери SQL» або «Відвідуваність програми».

На цьому кроці я створюю ресурсну групу «Diploma_Project». Подальший дейплоймнт веб сервісу та бази даних буде створено саме в цій ресурсній групі.

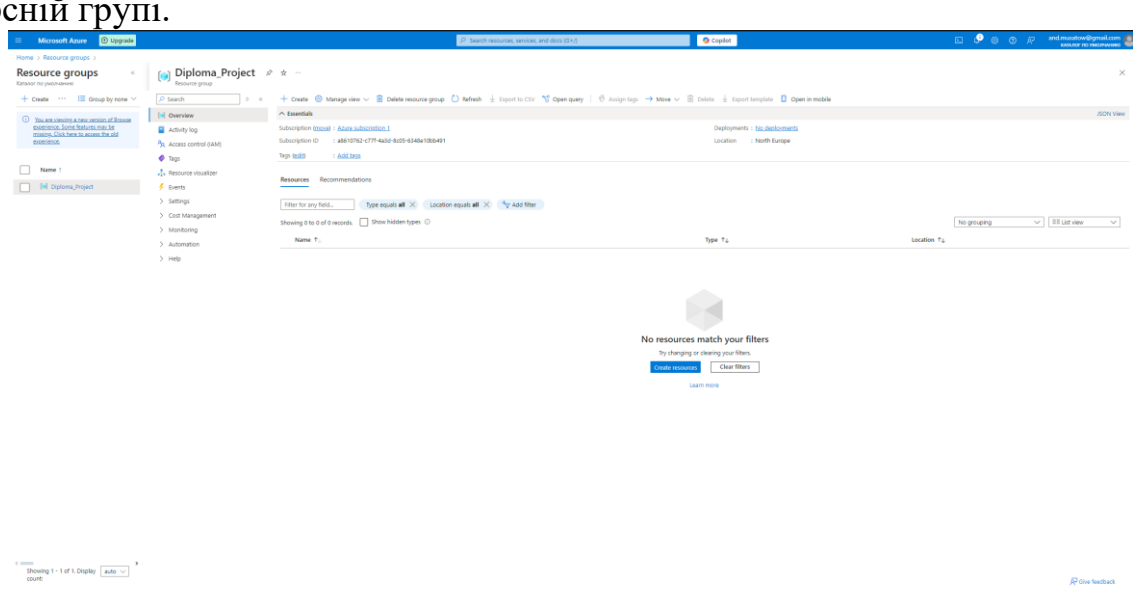


Рисунок 3.5 – Azure Resource Group

3.7 Azure App Service для контейнерізації

Azure App Service є багатофункціональною платформою як послуга (PaaS), спеціально розробленою для розміщення та масштабування веб-додатків, зокрема веб-сайтів та API. З моменту появи підтримки контейнерів, платформа набула значної гнучкості, дозволяючи розробникам запускати як традиційні веб-додатки, так і контейнеризовані рішення.

Однією з ключових переваг Azure App Service є спрощення процесу розробки та розгортання. Платформа автоматично керує інфраструктурою контейнерів, звільняючи розробників від рутинних операцій з підтримки середовища виконання. Крім того, тісна інтеграція з системами контролю версій, такими як GitHub і Azure DevOps, значно спрощує процес безперервної інтеграції та доставки.

Сильні сторони Azure App Service:

- використання Azure App Service означає, що ми маємо доступ до механізмів масштабування плану Azure App Service (масштабування по вертикалі або горизонталі). Ви можете зробити це на основі метрики (наприклад, використання ЦП або пам’яті) або на основі розкладу.
- це повністю керована платформа (сервіс PaaS). Треба дбати тільки про свій імідж.
- інтеграція з Azure DevOps і GitHub дуже тісна. Використовуючи функцію Центру розгортання, ви можете зв’язати джерело (наприклад, GitHub), де ми можемо вибрати, де буде розміщено наш

код/конфігурацію контейнера, і дозволити Azure автоматично генерувати робочий процес GitHub для дій GitHub.

- підтримка Windows і Linux. Важливим доповненням, про яке ми повинні знати, є те, що операційні системи вибираються під час розгортання плану App Service (ще одна служба, яка автоматично розгортається під час створення веб-програми).
- чудова інтеграція з Visual Studio та VS Code.
- він надає хороші інструменти для усунення несправностей і діагностики. Хоча спочатку це може здатися досить важким, з часом ви знаходитимете в ньому все більшу й більшу користь.

Слабкі сторони Azure App Service:

- обмеження масштабування. Сам план Azure App Service просто забезпечує додатковий екземпляр, не знаючи про час виконання. Служба програм Azure не враховує, що це контейнер, який працює на службі програм.
- тоді як звичайна служба додатків Azure запускає код безпосередньо та має тісну інтеграцію з такими рішеннями, як Application Insights і сама консоль Kudu, веб-програми для контейнерів цього не роблять. Деталі інфраструктури та журналювання все ще доступні, але якщо ми хочемо налагодити наш контейнер, нам потрібно змінити наше рішення, щоб забезпечити правильний тип журналювання для платформи Azure.
- опція з кількома контейнерами все ще знаходиться на етапі попереднього перегляду.
- підходить лише для веб-рішень (веб-сайтів і API). Коли ми починаємо запускати складніші сценарії, ми не просто хочемо запустити веб-сайт і набір серверних API. Служба додатків Azure надає доступ лише до портів 80 і 443. Якщо ви хочете запустити Rabbit MQ у службі додатків Azure, єдиним варіантом буде використання Docker Compose та надання іншому контейнеру доступу до контейнера RabbitMQ через внутрішню мережу.

Azure App Service є потужним інструментом для розробки та розгортання веб-додатків, як традиційних, так і контейнеризованих. Однак, перед вибором цієї платформи слід ретельно оцінити її можливості та обмеження в контексті конкретного проекту.

Процес деплойменту веб додатку за допомогою Azure App Service. Спочатку ми обираємо App Service в Search барі. Як ви бачите на цьому скріншоті, на данний момент на моєму акаунті не створено жодних App Services:

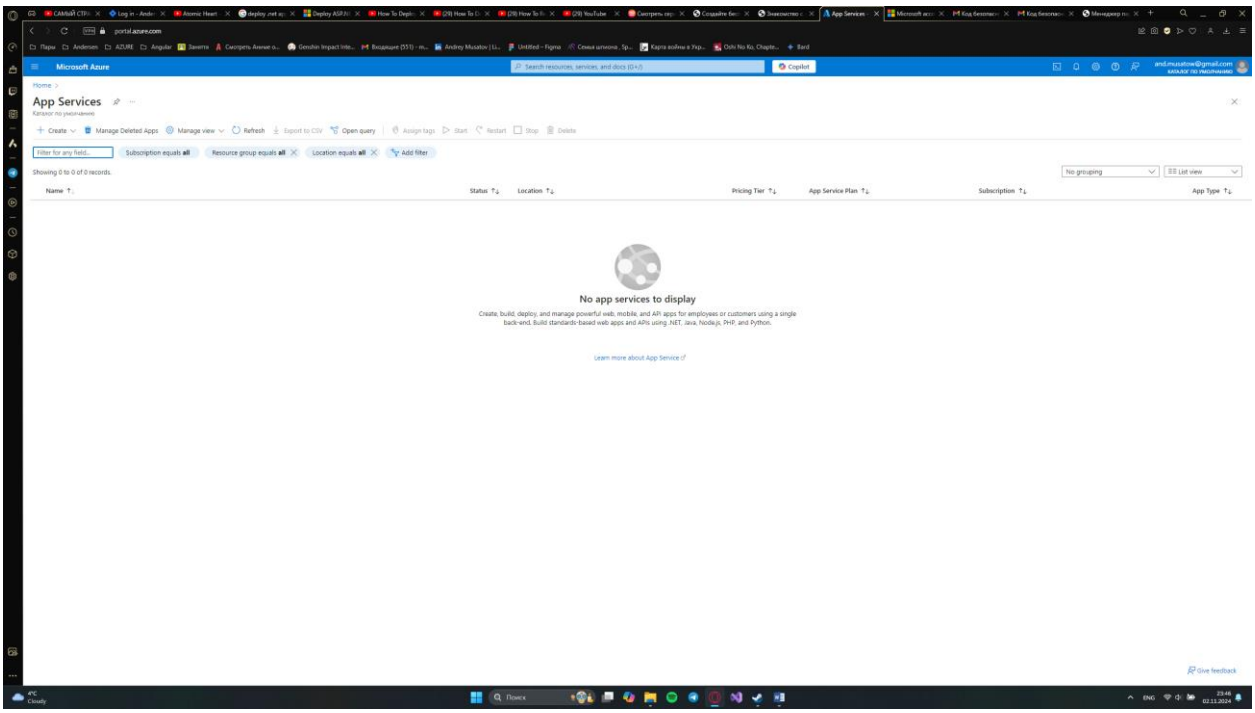


Рисунок 3.6 – Azure App Services

Для того щоб створити App Service, потрібно нажити на кнопку Create та заповнити потрібні данні, як на цьому скріншоті:

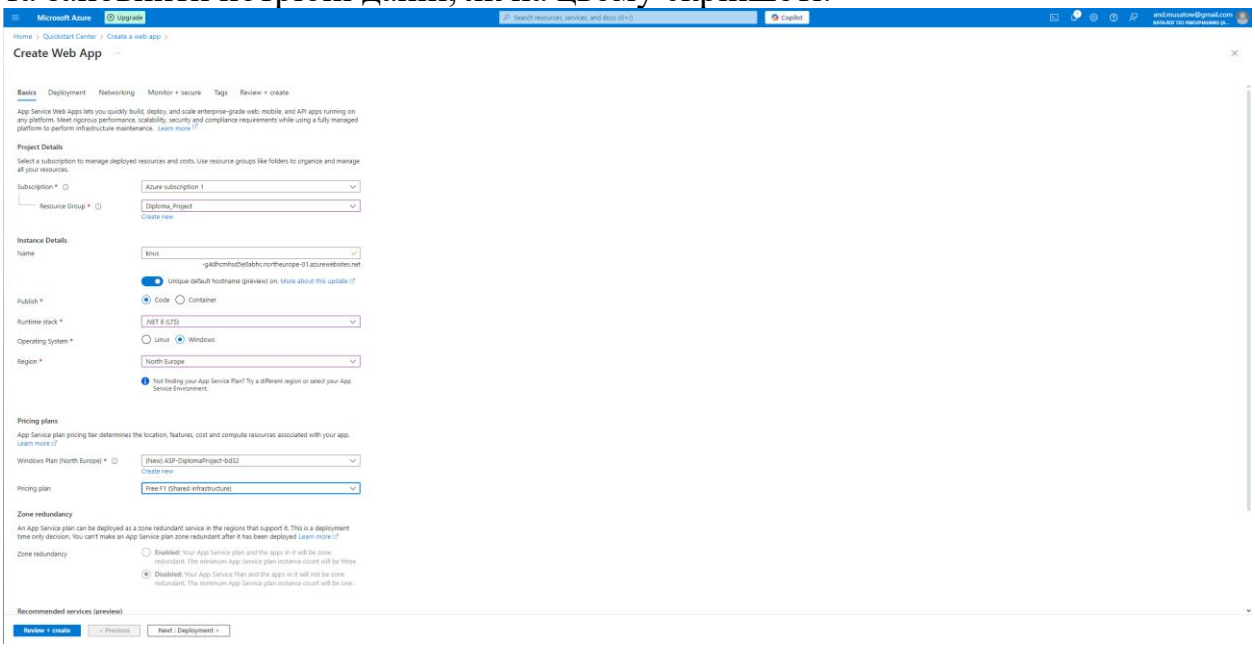


Рисунок 3.7 – створення контейнеру Azure App Services

Після заповнення форми і підтвердження, Майкрасофт демонструє результат. Ця секція поділена на 4 сегмента:

- Деталі(Details) – демонструє основні деталі створеного контейнеру. обрана підписка, ресурсна група, до якої додається контейнер. Ім’я створеного контейнеру, для цього проекту я обрав ім’я knus. Та версія .NET, обрана остання – восьма.
- App Service Plan – сгенероване внутрішнє ім’я, операційна система, обраний регіон та виділена оперативна пам’ять (1 ГБ – це максимальна кількість для безкоштовної підписки).

– Відстеження(Monitor) – цей сегмент частично дублює інформацію з минулих сегментів. Але є важлива інформація щодо роботи Application Insights. В мене ця опція включена.
– Деплоймент(Deployment) – деталі самого деплойменту.

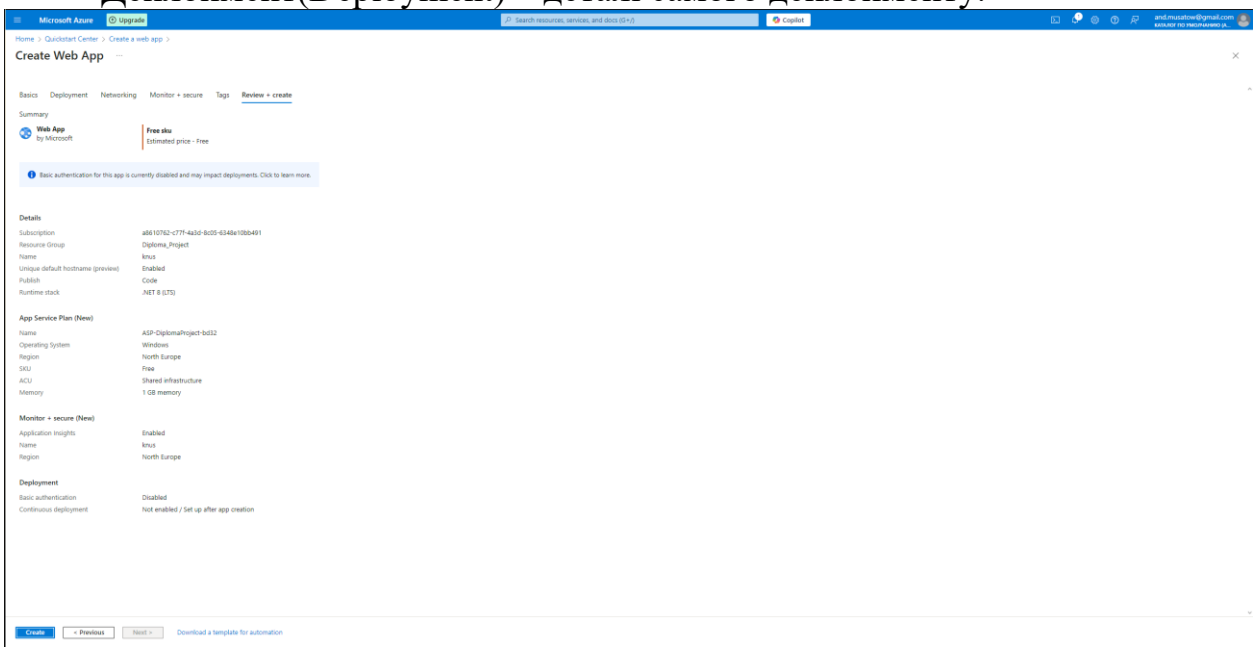


Рисунок 3.8 – огляд контейнеру Azure App Services

Після підтвердження починається процес створення контейнеру:

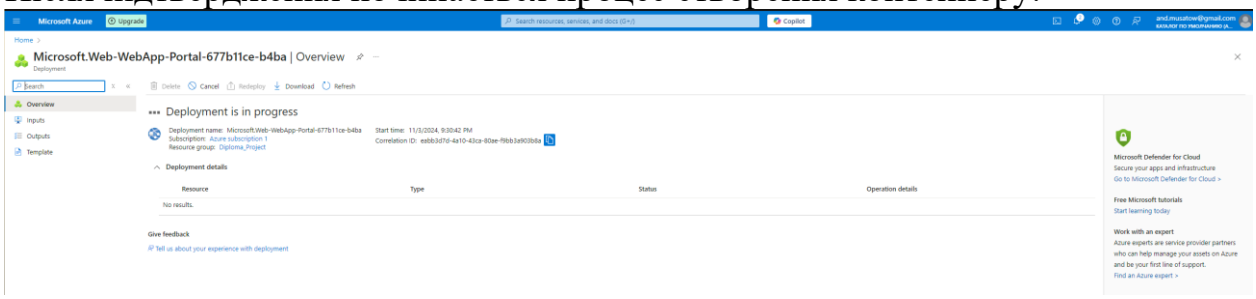


Рисунок 3.9 – процес розгортання контейнеру Azure App Services

Через декілька хвилин отримано результат розвертання:

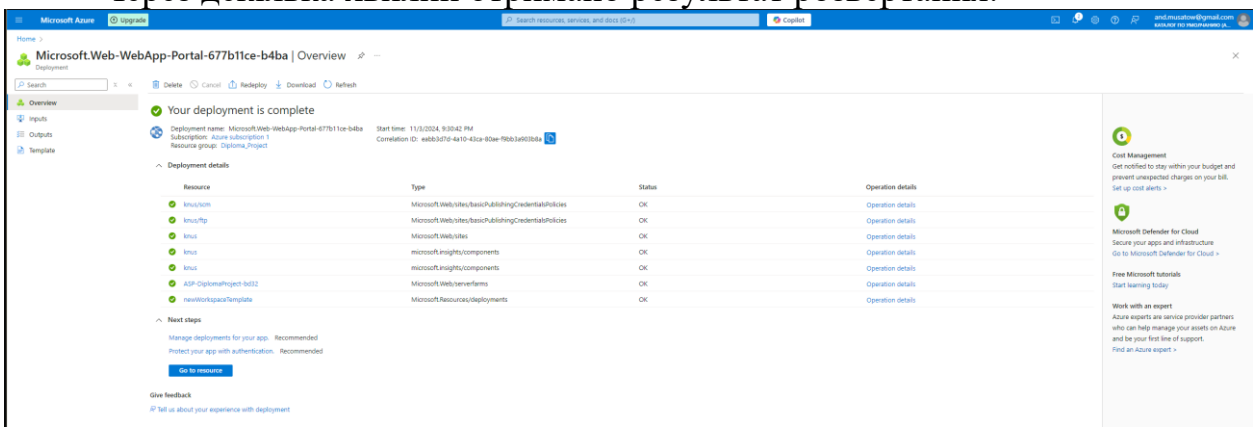


Рисунок 3.10 – демонстрація уіспішного розгортання контейнеру Azure App Services

Зараз Azure App Service може продемонструвати створений та розвернутий контейнер.

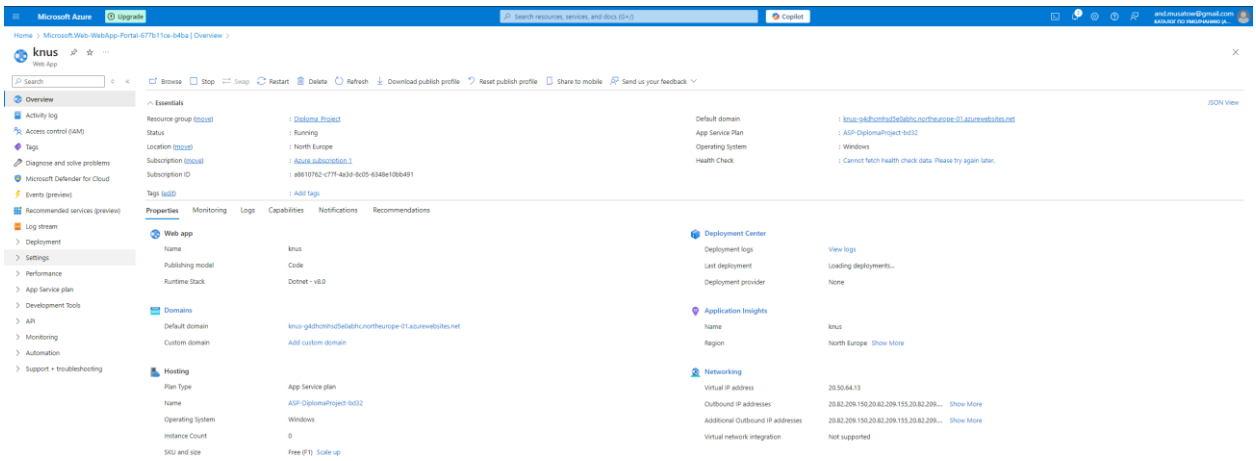


Рисунок 3.11 – огляд створеного контейнеру Azure App Services

Переходячи по посиланню, можна побачити що розвертання дійсно відбулося.

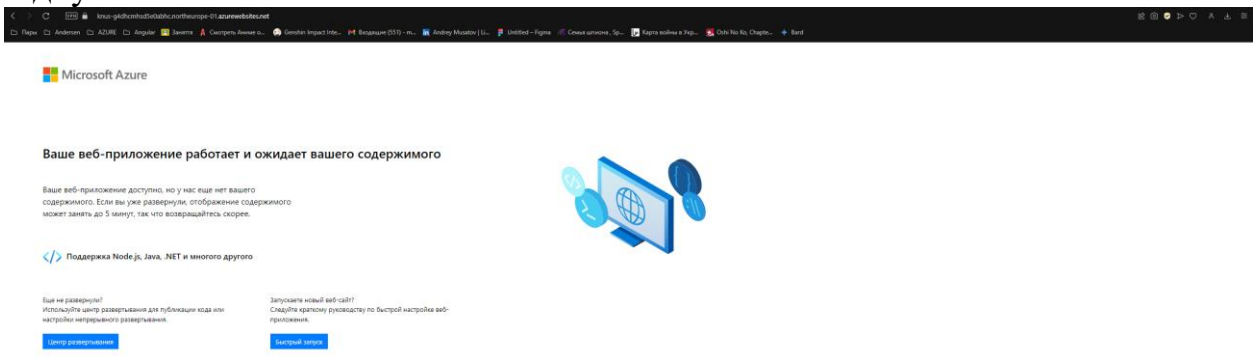


Рисунок 3.12 – перехід на домейн створеного Azure App Services

Тепер переходимо до деплойменту веб додатку. По перше обираємо необхідний варіант в Visual Studio:

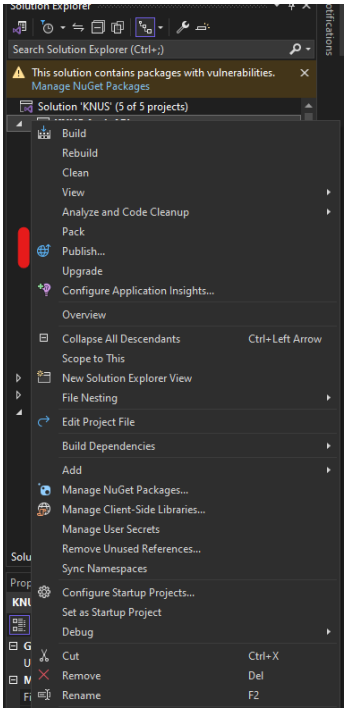


Рисунок 3.13 – початок розгортання додатку в Visual Studio

Обираємо варіант “Publish”:

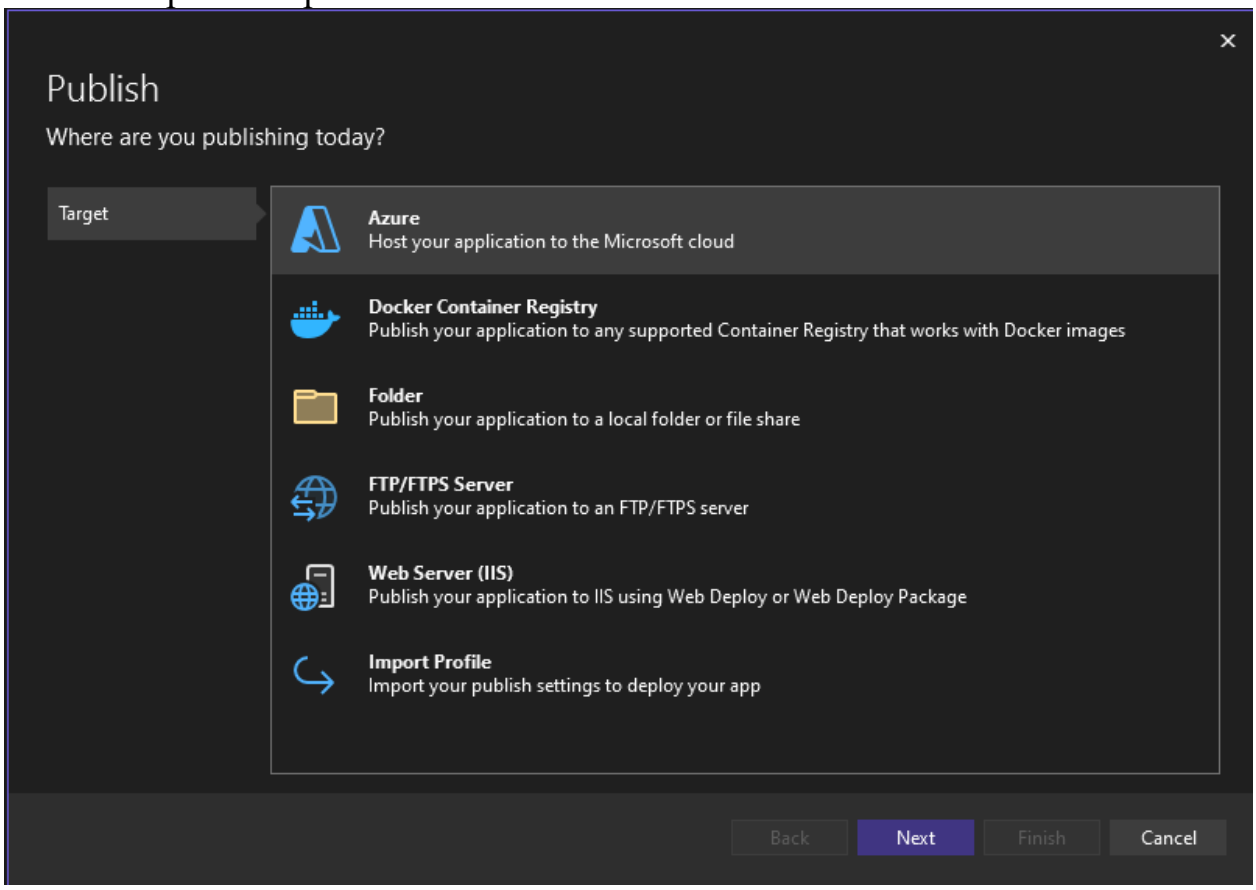


Рисунок 3.14 – обирання способу розгортання додатку в Visual Studio

Після цього відкривається декілька варіантів і обираємо варіант Azure

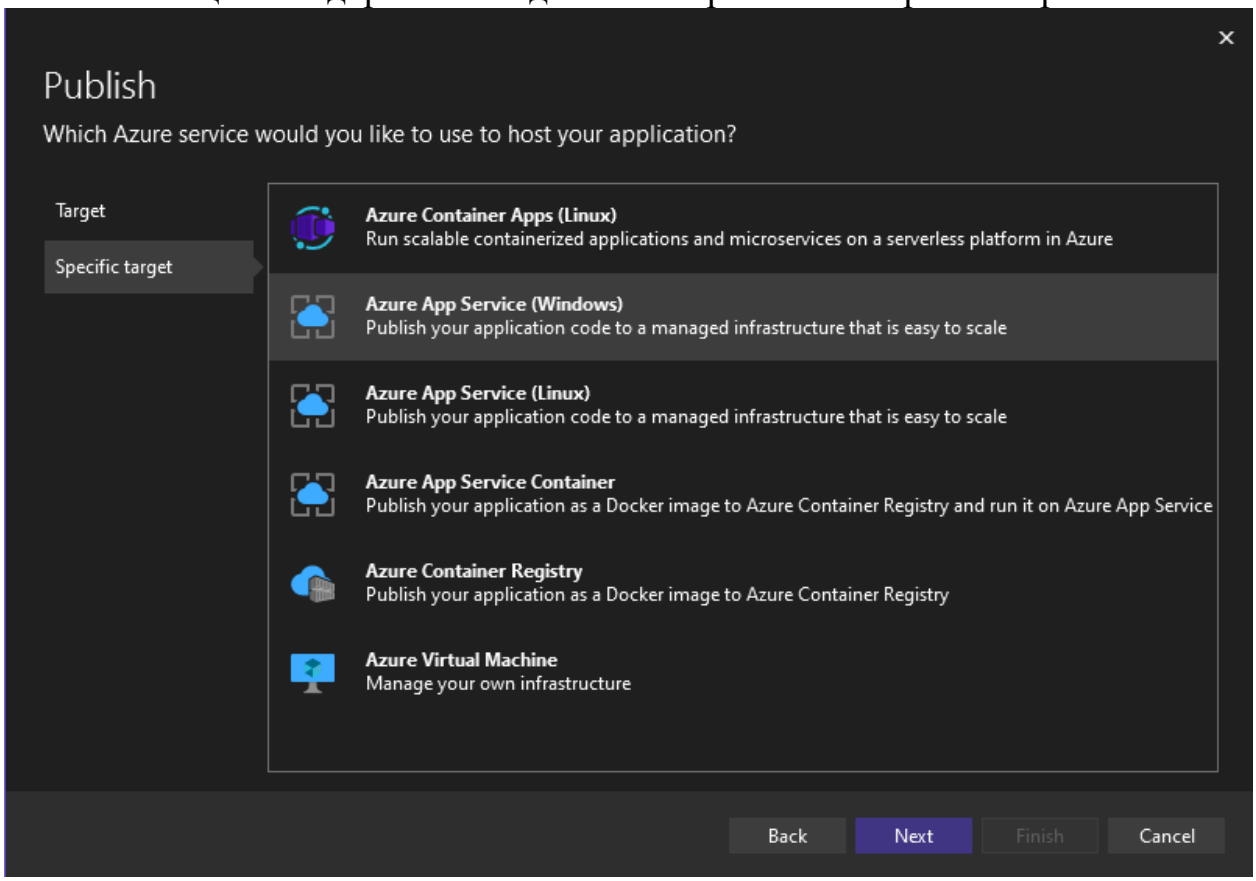


Рисунок 3.15 – Azure App Service в Visual Studio

Обираємо раніше описаний варіант «Azure App Service».

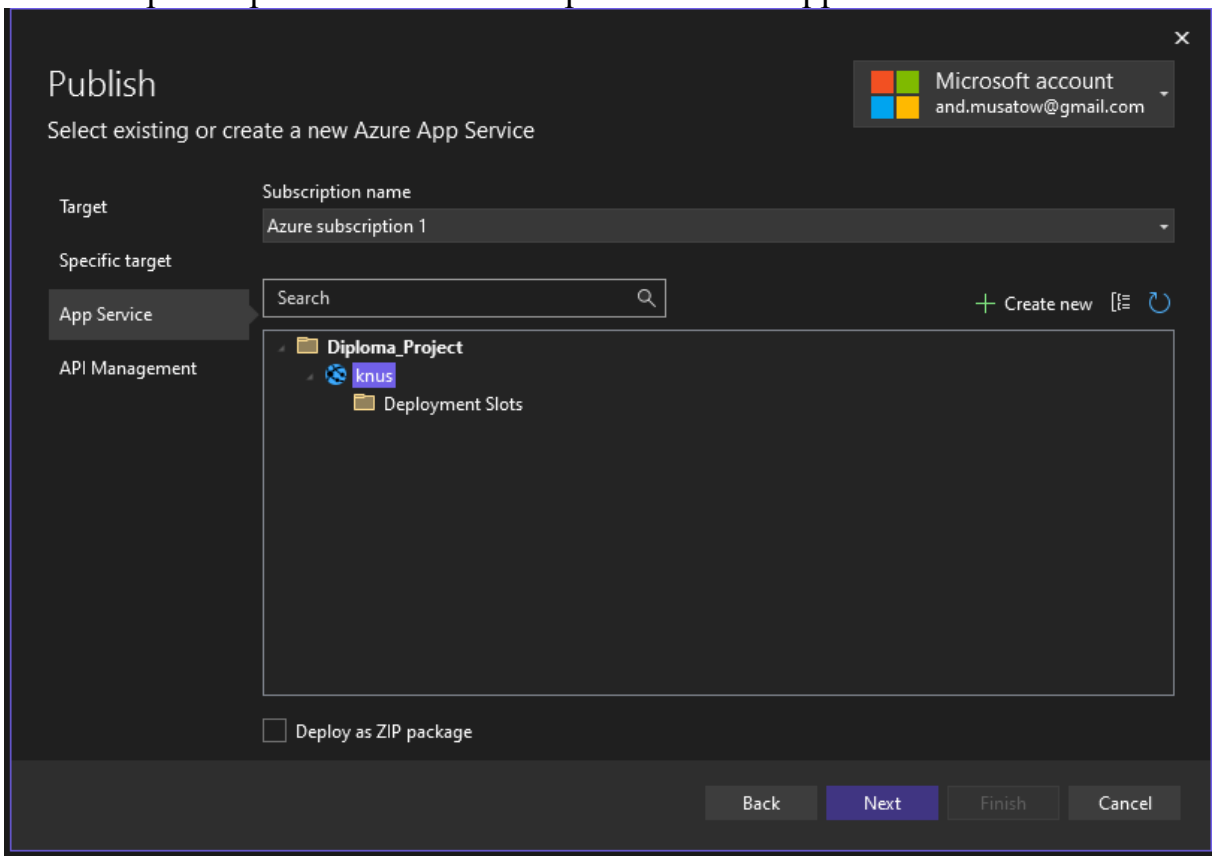


Рисунок 3.16 – отримання створеного Azure App Service в Visual Studio

Visual Studio отримала створений раніше Azure App Service контейнер для розгортання.

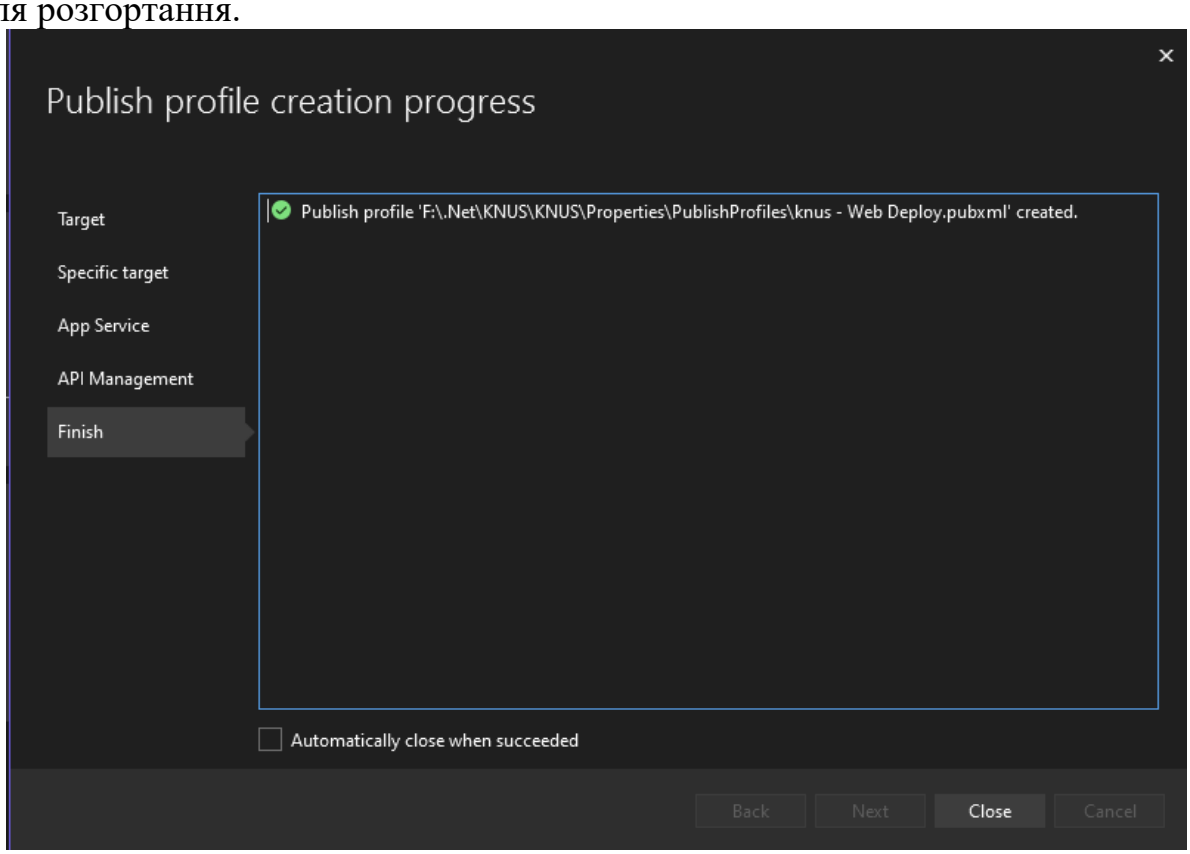


Рисунок 3.17 – створення профайлу Azure App Service в Visual Studio

Після запуску процесу був створений профайл який тепер можливо розгорнути у хмарі.

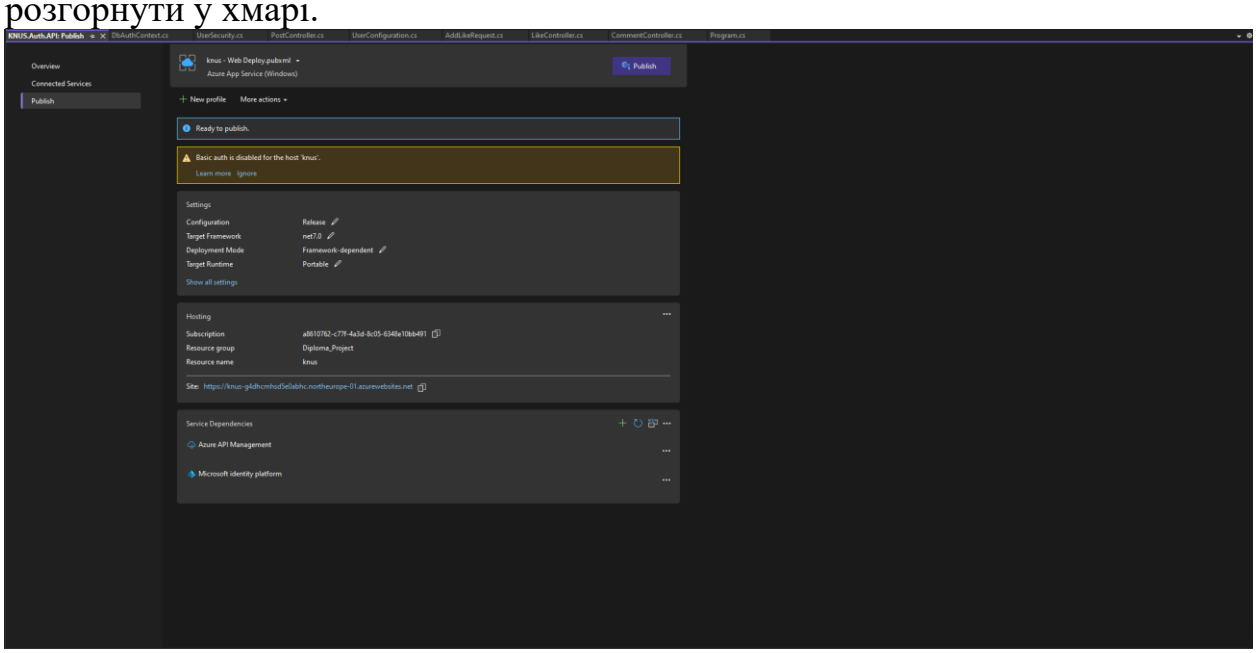


Рисунок 3.18 – створення профайлу Azure App Service в Visual Studio

3.8 Azure SQL

Тепер треба додати підтримку бази даних для веб додатку. Це можна зробити за допомогою сервісу Azure SQL.

Azure SQL — це централізовано керована платформа як сервісний механізм бази даних (PaaS), який виконує більшість служб адміністрування, наданих базою даних, наприклад, резервне копіювання, виправлення, оновлення та моніторинг, з дуже незначною дією користувача.

Для Azure SQL потрібна належним чином виправлена операційна система та надійна інсталяція SQL Server, бажано останньої версії. Крім того, це допомагає розробити високодоступний і швидкий рівень зберігання даних.

Нижче наведено деякі типи баз даних Azure SQL:

- база даних SQL Azure: це повністю керована хмарна служба реляційної бази даних. Висока доступність і вбудований штучний інтелект завжди зберігають свою довговічність і продуктивність із 99,95% SLA.
- SQL Server на віртуальних машинах Azure: цей тип бази даних допомагає перенести ваш сервер SQL за допомогою гібридного підключення та гнучкості Azure. Ви можете зареєструвати свої віртуальні машини та скористатися перевагами автоматизованого керування та вбудованої безпеки.
- керований екземпляр Azure SQL: розширена хмарна база даних, яка поєднує в собі сумісність механізму для найширшого SQL Server і надає переваги сучасного PaaS.
- Azure SQL Edge: це база даних периферійних обчислень IoT, яка поєднує часові ряди та потокове передавання даних із вбудованими функціями графіків і машинним навчанням.

База даних SQL Azure — це служба реляційної бази даних для хмарних інфраструктур, побудована на технології Microsoft SQL Server і пропонує всі

переваги традиційної бази даних SQL Server, включаючи високу продуктивність і масштабованість, надійну безпеку та широкі можливості керування.

Крім того, це дає змогу скористатися перевагами гнучкості та ефективності хмари, щоб масштабувати свою базу даних на вимогу та платити лише за ресурси, які використовуються.

Нижче наведено деякі функції бази даних Microsoft Azure SQL:

- автоматичне налаштування: База даних SQL Azure автоматично налаштовується на основі шаблонів робочого навантаження, що полегшує підтримку продуктивності в масштабі.
- динамічна масштабованість: базу даних Microsoft Azure SQL можна масштабувати вгору та вниз відповідно до ваших потреб без необхідності надання будь-якої інфраструктури чи керування нею.
- висока доступність: База даних Azure SQL забезпечує вбудовані можливості високої доступності (HA) і аварійного відновлення (DR), щоб зберегти ваші дані в безпеці та завжди доступними навіть під час збою або катастрофи.
- корпоративна безпека: База даних Microsoft Azure SQL пропонує повний набір функцій безпеки, які допомагають зберегти ваші дані в безпеці, включаючи шифрування, автентифікацію та авторизацію.
- гнучке ціноутворення: з базами даних SQL Azure ви можете мати різні варіанти ціноутворення відповідно до свого бюджету та вимог. Ви можете вибрати модель ціноутворення на основі оплати за використання або на основі передплати та збільшувати або зменшувати масштаб залежно від потреб

Як згадувалося раніше, база даних Azure SQL — це служба реляційної бази даних, яка не залежить від хмари, створена на базі надійної технології Microsoft SQL Server. Вона пропонує всі переваги SQL Server, включаючи високу доступність, безпеку та масштабованість. Крім того, вона проста у використанні та управлінні, тому можн швидко розпочати роботу та зосередитися на бізнесі, а не на базі даних.

Переходимо до розгортання БД. Для цього потрібно додати залежність Azure SQL Database.

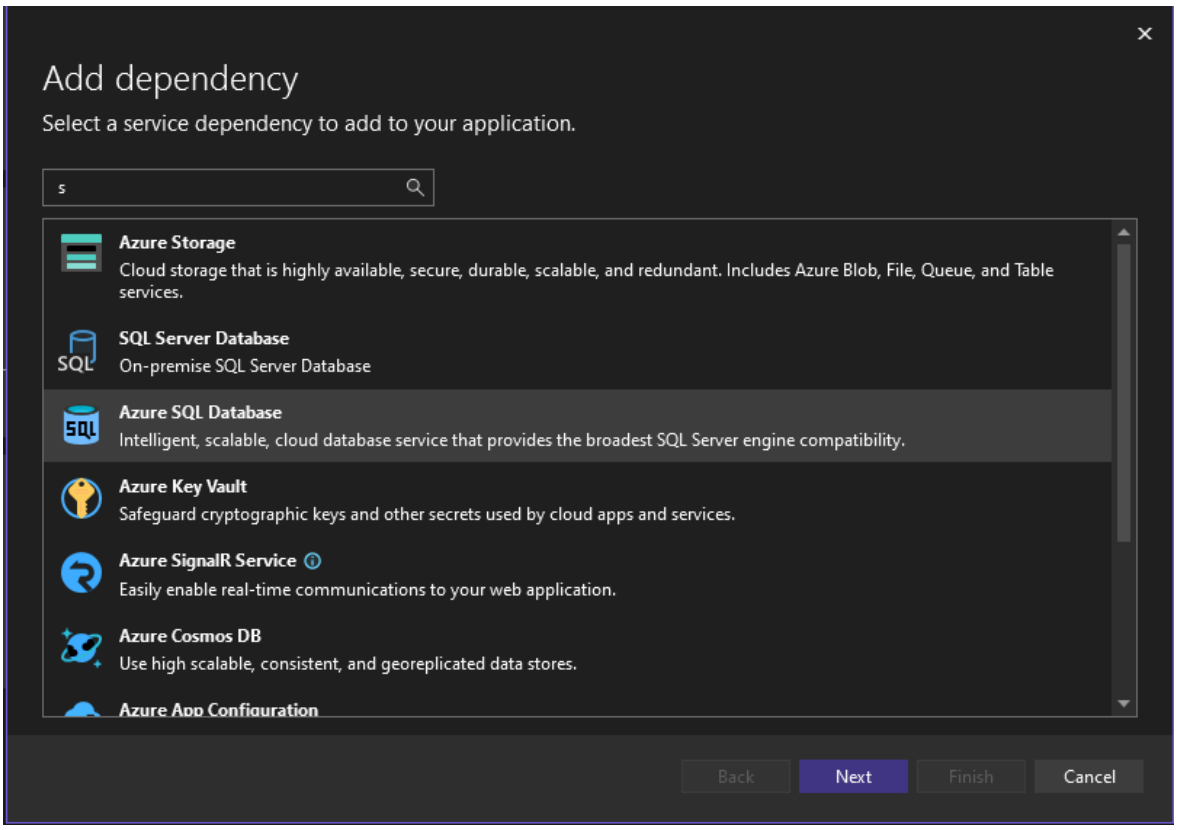


Рисунок 3.20 – обрання Azure SQL Database в Visual Studio

Після цього Microsoft намагається знайти існуючі бази даних. Але на даний момент не було створено жодної БД, саме тому було повернуто пустий список.

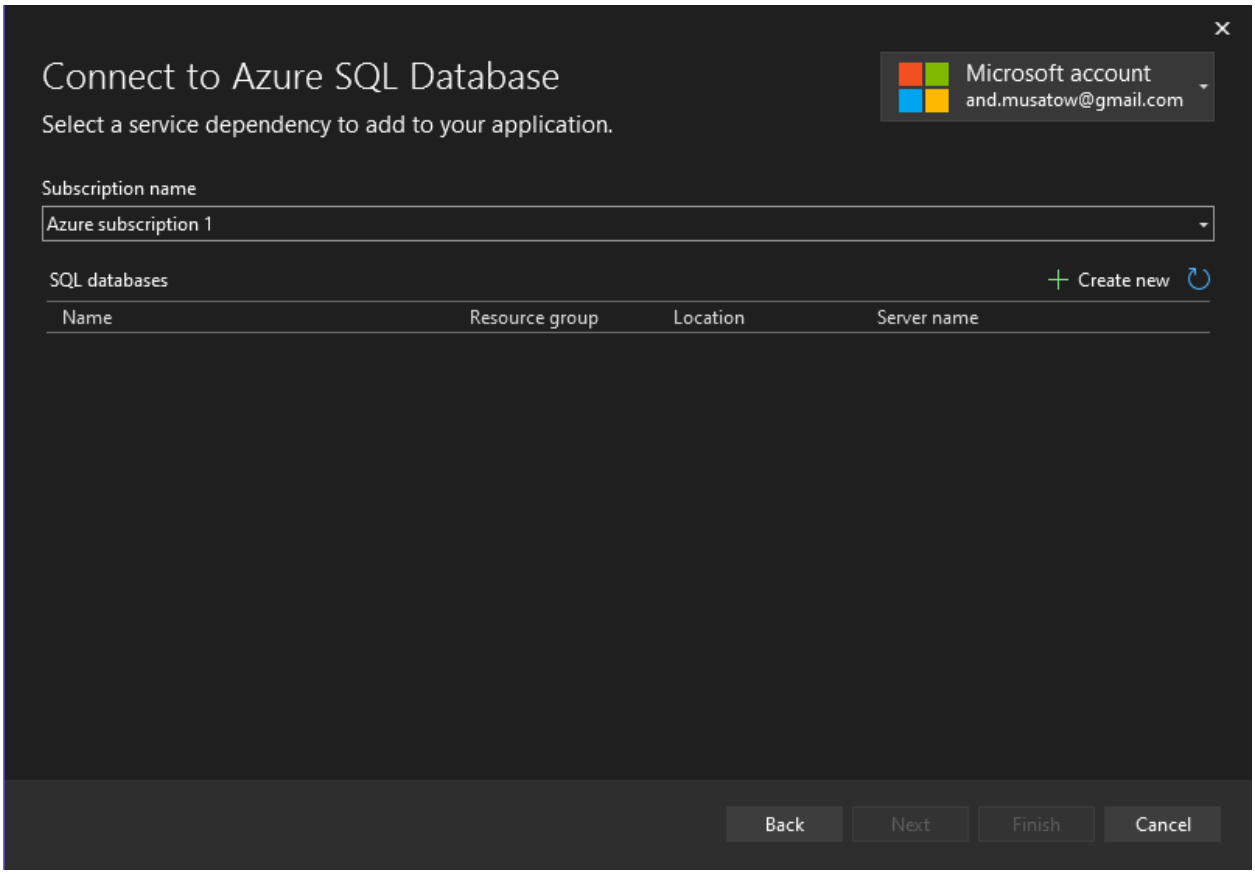


Рисунок 3.21 – пустий список Azure SQL Database в Visual Studio

Створюємо БД. Для цього треба заповнити цю форму:

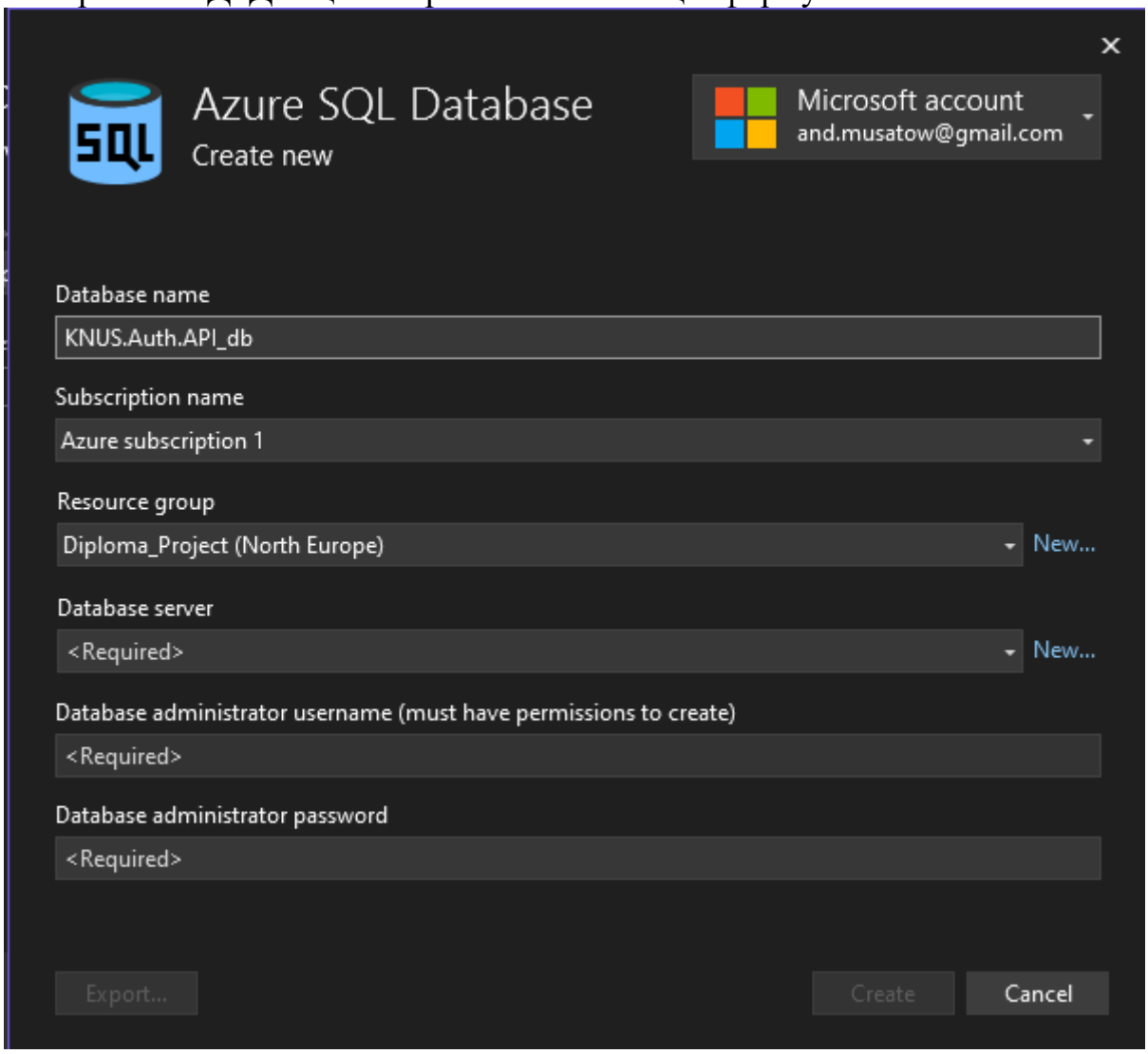


Рисунок 3.22 – створення БД в Azure SQL Database за допомогою Visual Studio

Але спочатку треба створити Database Server:

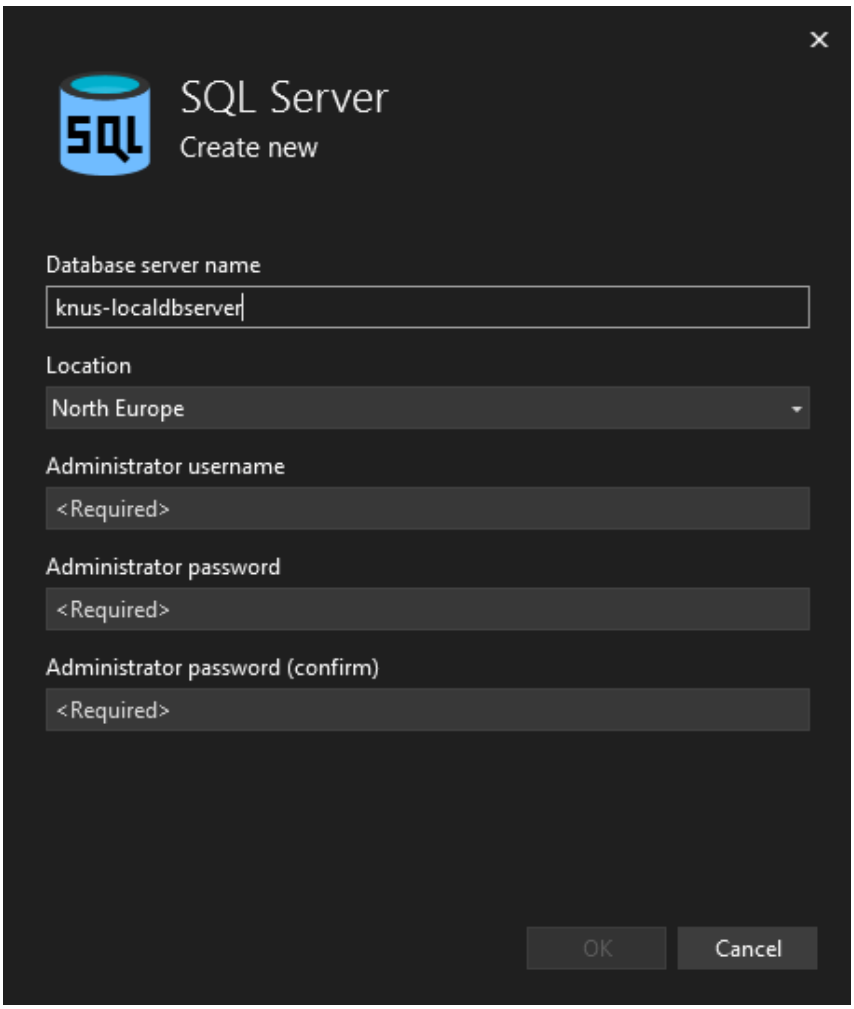


Рисунок 3.23 – створення Серверу в Azure SQL Database за допомогою Visual Studio

Після створення серверу – повертаємося до самого розвертання БД:

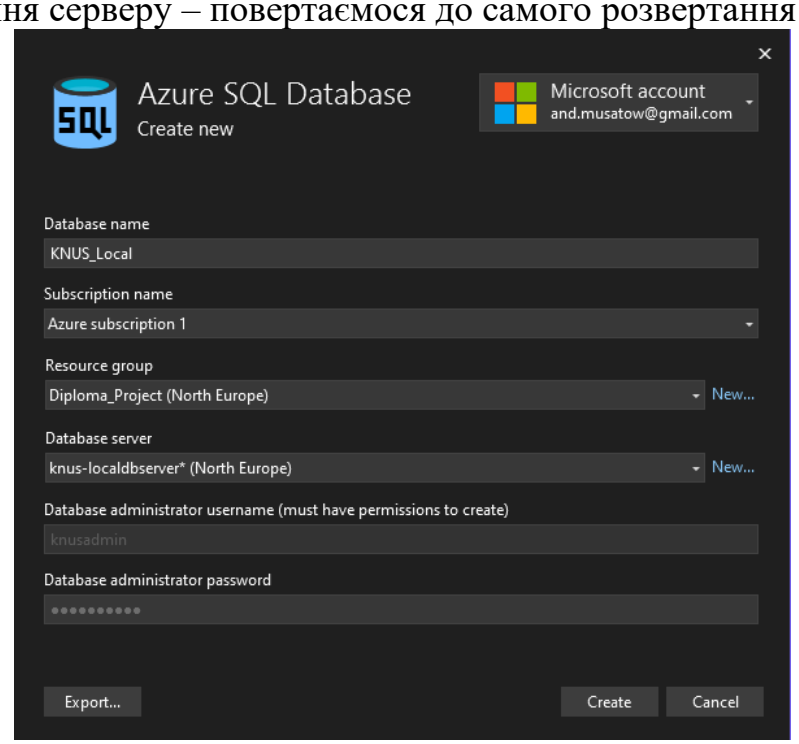


Рисунок 3.24 – створення БД в Azure SQL Database за допомогою Visual Studio

І тепер Microsoft повертає нашу створену БД.

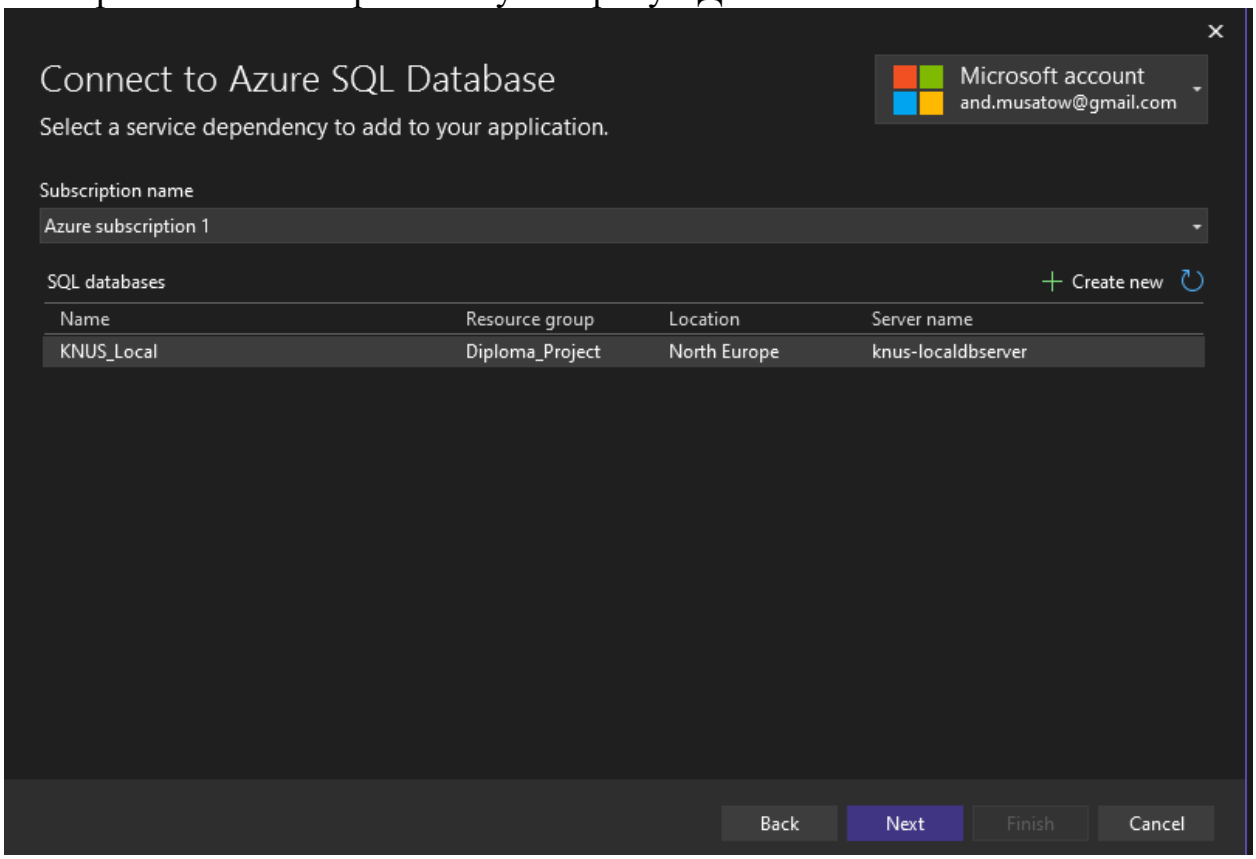


Рисунок 3.25 – створена БД в списку Azure SQL Database в Visual Studio

Також її тепер можна побачити на сайті Azure Portal:

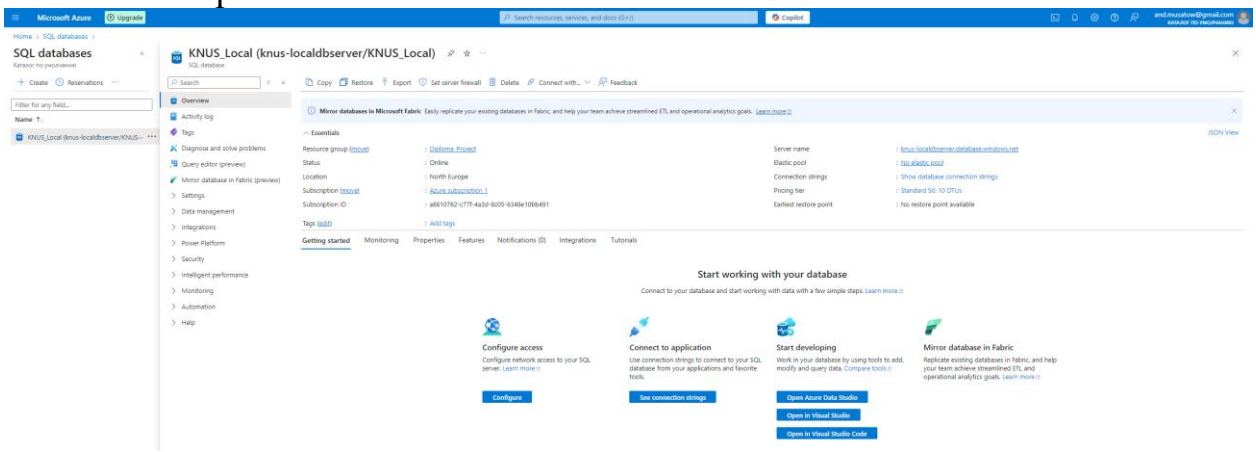


Рисунок 3.26 – створена БД в Azure Portal

Наступним кроком буде підключення бази даних до профайлу веб додатку:

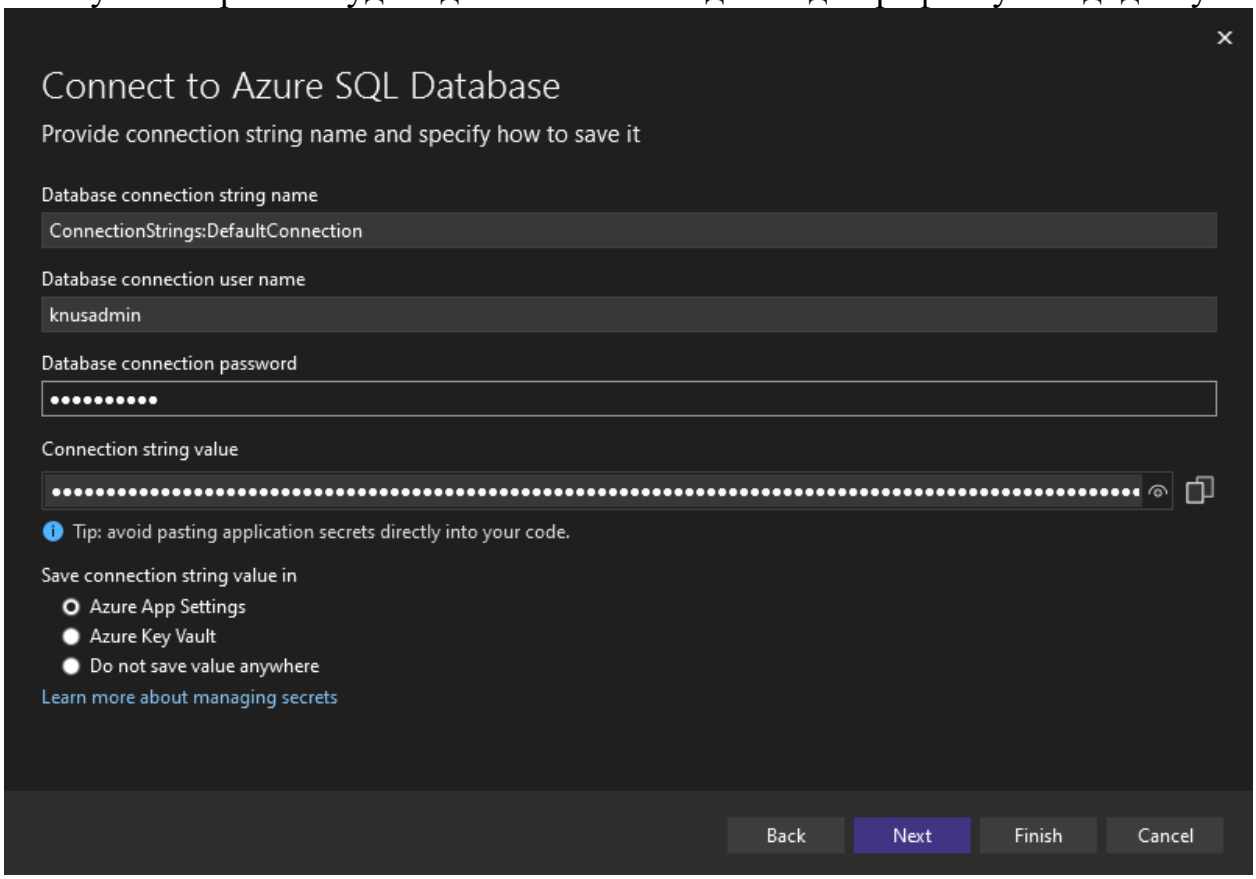


Рисунок 3.27 – підключення БД до профайлу в Visual Studio

Як можна побачити на наступному скріншоті, створена база даних «підтягнулася» до додатку і відмічіна, що готова до розгортання.

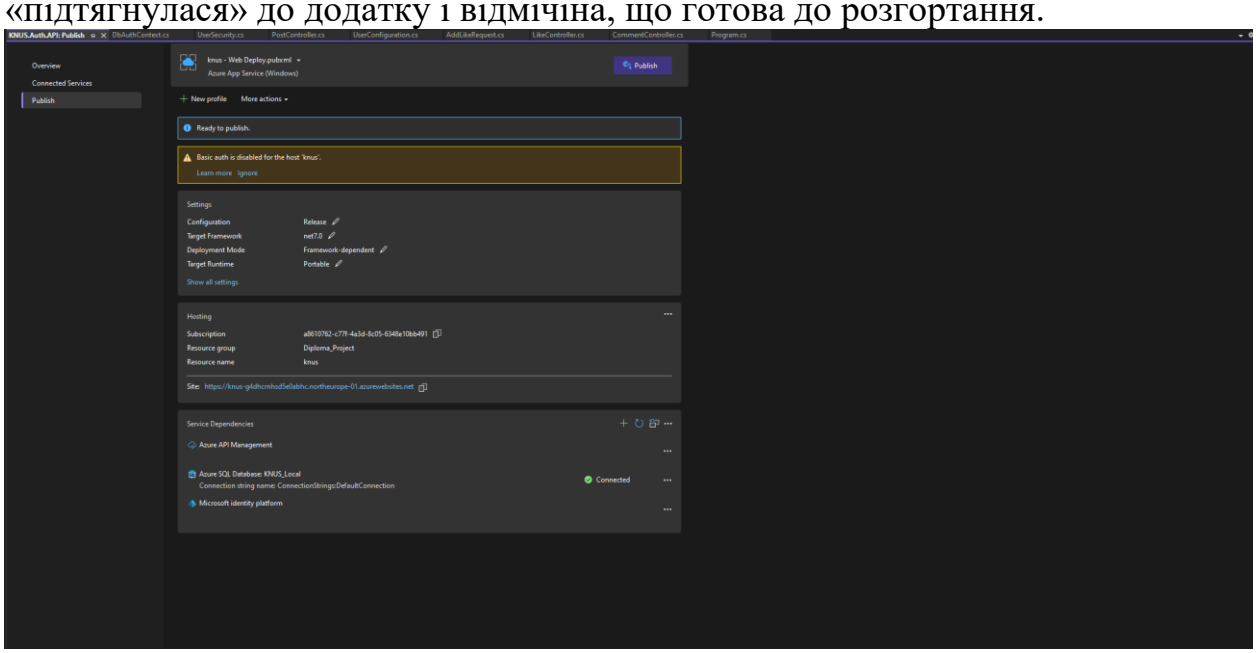


Рисунок 3.28 – демонстрація успішного підключення БД до профайлу в Visual Studio

Тепер перевіримо підключення бази даних через MS SQL Server. Введені потрібні данні для підключення, тобто дублюємо данні які використовувалися для створення SQL Server.

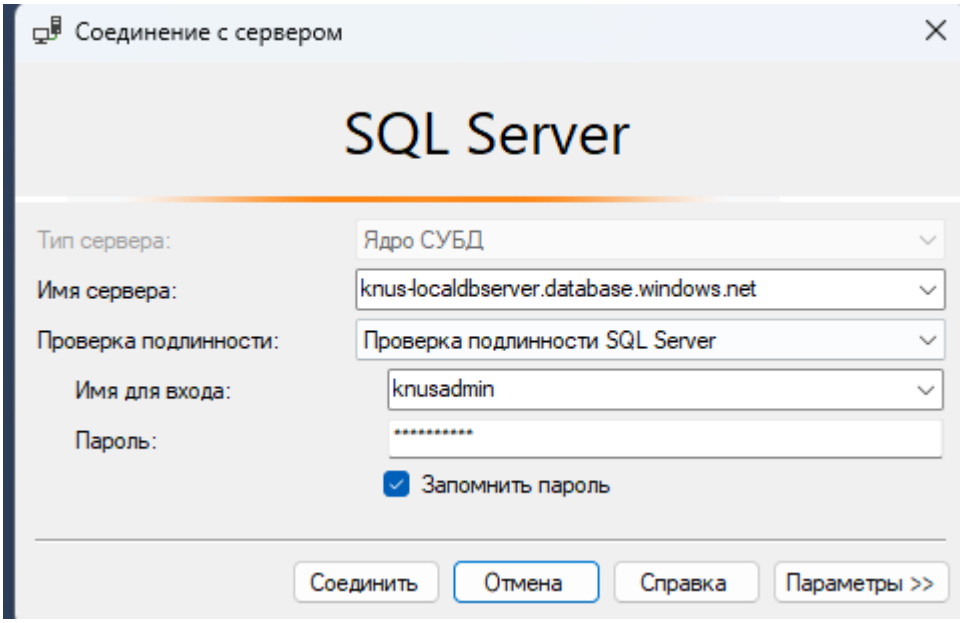


Рисунок 3.29 – підключення до БД розгорнутої на сервері за допомогою MS SQL

Але ми не здатні підключитися через обмеження хмарного провайдеру. Щоб вирішити цю проблему потрібно додати мою IP адресу в білий список.

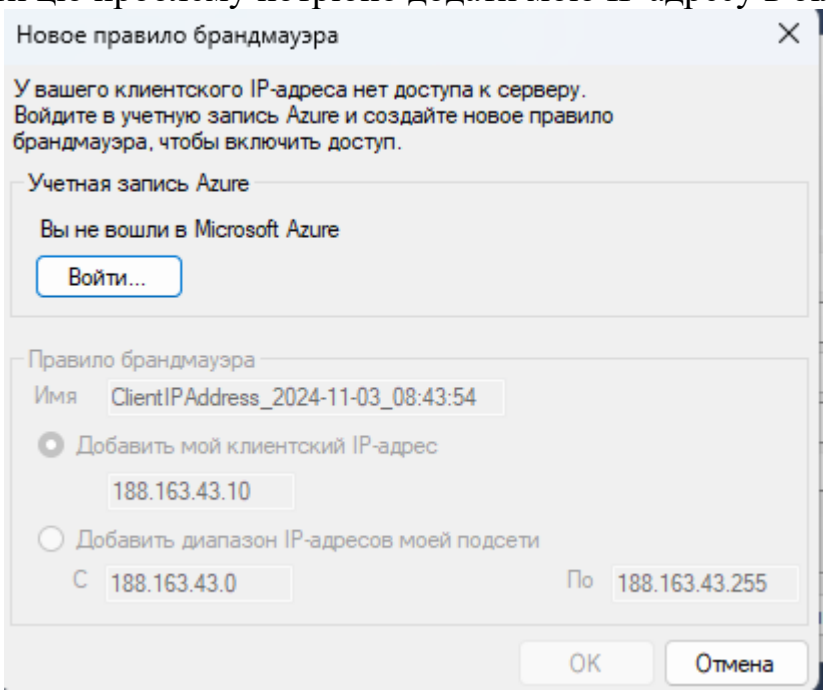


Рисунок 3.30 – помилка підключення до БД розгорнутої на сервері за допомогою MS SQL

Необхідно перейти до розділу «Networking» і додати IP адресу в білий список.

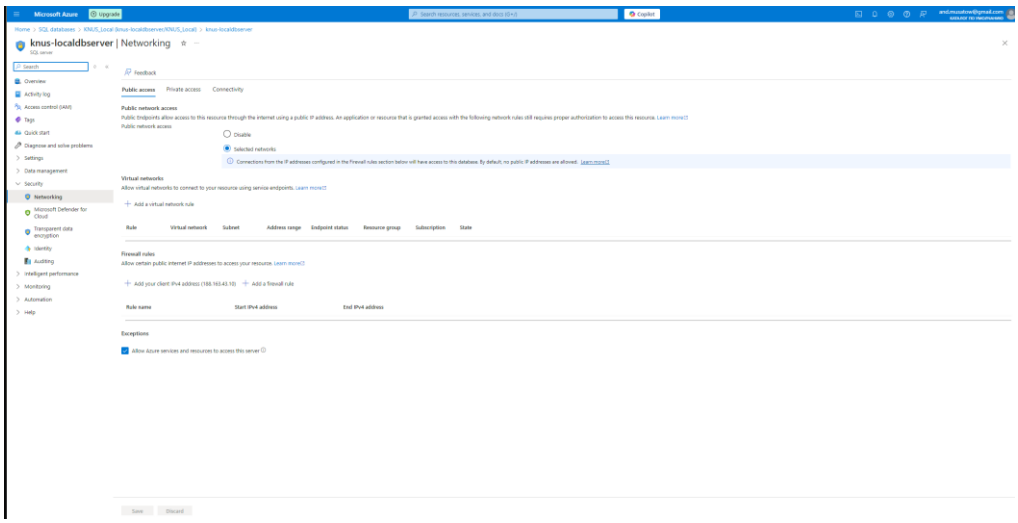


Рисунок 3.31 – білий список серверу

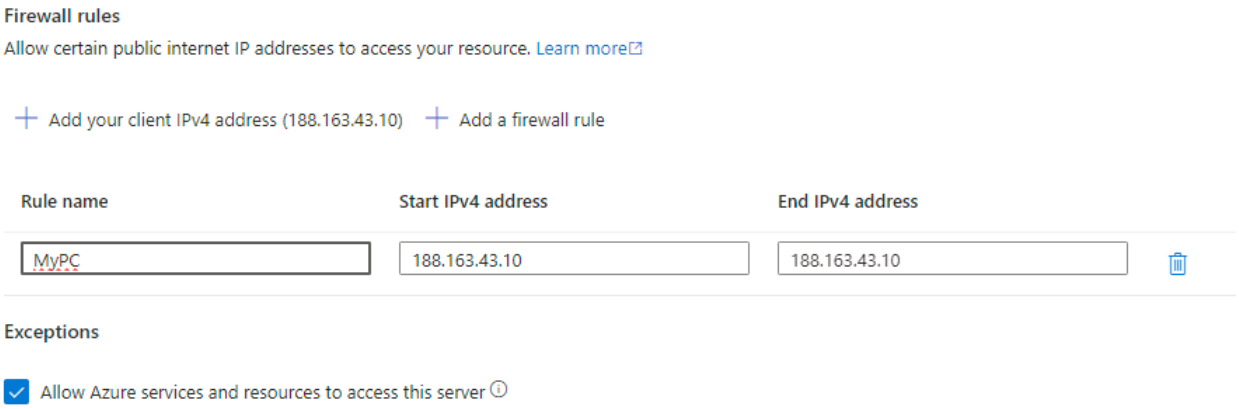


Рисунок 3.32 – додання до білого списку моєї IP адреси

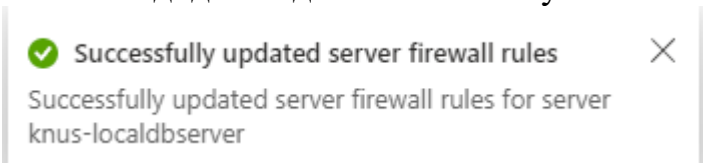


Рисунок 3.33 – IP адреса успішно додана

Після додання IP адреси в білий список і ще одної спроби приєднатися до створеного серверу ми отримуємо відповідь:

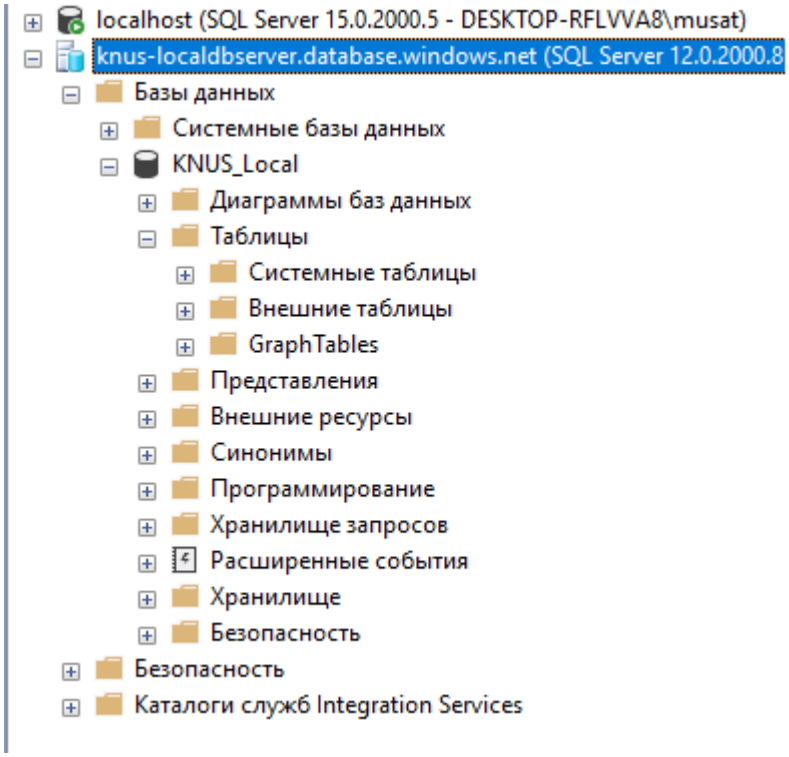


Рисунок 3.34 – підключено до розгорнутої БД

Після всіх пройдених кроків ми нарешті можемо провести розгортання веб додатку:

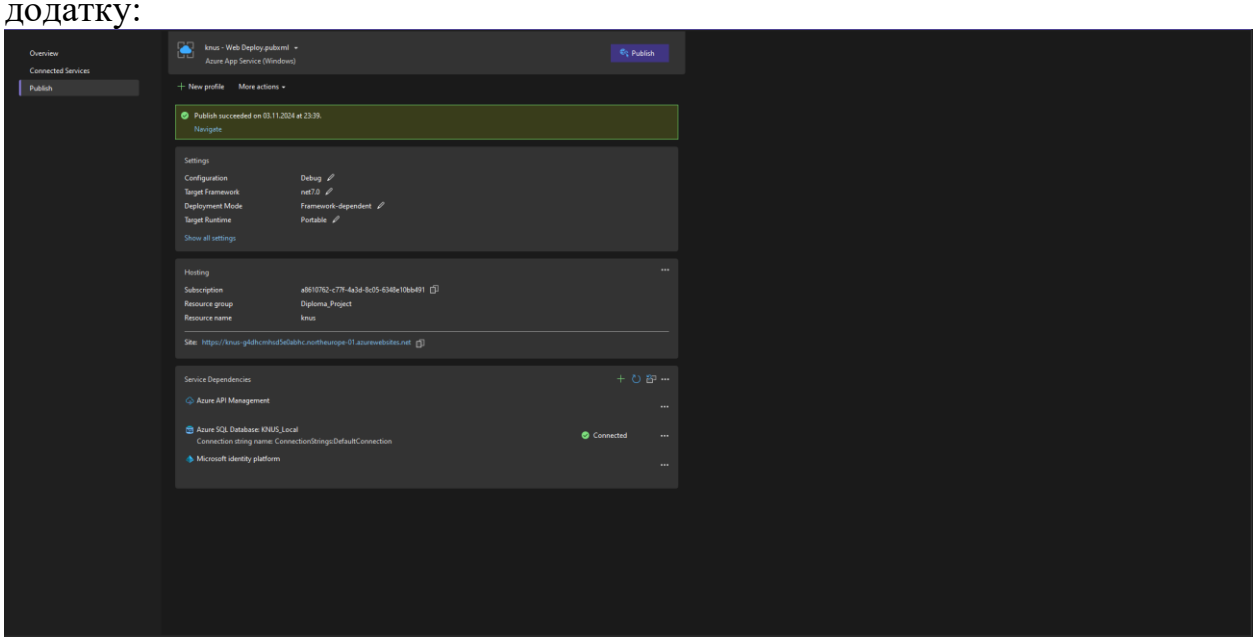


Рисунок 3.35 – підключуспішне розгортання веб додатку

Висновок до розділу

У цьому розділі було обрано провайдер хмарних технологій. Azure було обрано саме через тісну інтеграцію з технологією .NET і іншими платформами Microsoft. Також, Azure – один із найбільших провайдерів хмарних технологій. Також, було описано декілька сервісів Microsoft Azure. Такі як: Azure App Service, Azure Function, Azure SQL і інші.

Було пройдені шляхи створення аккаунту з пробною підпискою для студентів. Був проведений детальний опис Azure Portal.

Було проведено розгортання веб додатку за допомогою Azure App Service. Також було створено сервер бази даних і саму базу даних. Також було додано IP адресу до білого списку.

4. Ціноутворення хмарних сервісів Azure

4.1 Загальна ціна хмарних сервісів Azure

Azure — це загальнодоступна хмарна платформа від Microsoft, яка пропонує широкий спектр хмарних служб, таких як мережа, обчислення, зберігання та аналітика. Ви можете вибирати зі служб Azure для створення та масштабування своїх програм або переміщення існуючих програм у публічне хмарне середовище Microsoft.

Хоча хмарні обчислення можуть бути економічно ефективними, особливо для масштабованих і непередбачуваних робочих навантажень, витрати можуть швидко зрости, якщо ви не плануєте використання хмари. Перш ніж вибрати певний продукт або план, важливо знати свої потреби в хмарі.

На щастя, у вас є кілька способів запобігти тому, щоб рахунок організації за Azure перевищив запланований бюджет. Окрім вибору правильних регіонів, моделей ціноутворення та ресурсів для потреб, можливо відстежувати використання та використовувати інструменти розрахунку вартості й керування, щоб підтримувати витрати на хмару на потрібному рівні.

Azure надає моделі ціноутворення, щоб допомогти розробнику досягти оптимальної ціни для характеристик кожного робочого навантаження. Ось основні моделі:

- Pay-per-use pricing (оплата за використання)— ця модель виставляє рахунок за фактичне використання хмари за секунду. Ця гнучка модель дозволяє швидко надавати ресурси на вимогу, але за ринковою ціною, що означає, що це найдорожчий варіант.
- Azure spot instances (точкові віртуальні машини) — Azure надає знижки до 90% від ціни на вимогу, коли купляється резервна ємність. Однак Azure може перервати точкові віртуальні машини із завчасним сповіщенням за 30 секунд.
- Azure reserved instances (Зарезервовані екземпляри Azure) — якщо ви зобов’язуєтеся використовувати ресурси Azure протягом 1 або 3 років, можна отримати знижку до 72% від ціни за запитом.
- Azure hybrid benefit (Гібридна перевага Azure) — ця модель власної ліцензії (BYOL) дозволяє використовувати існуючі локальні розгортання Microsoft у хмарі та отримувати знижки на ресурси Windows, розгорнуті в хмарі Azure.

4.2 Тип ресурсу

Azure пропонує різні ціни для кожного типу ресурсу. Це означає, що використання, яке відстежує кожен лічильник, і кількість лічильників, пов'язаних з кожним ресурсом, залежать від типу ресурсу. Кожен лічильник відстежує певне використання, наприклад використання пропускної здатності відповідно до вхідного чи вихідного мережевого трафіку в бітах за секунду або розмір відповідно до ємності пам'яті в байтах. Azure співвідносить використання, яке відстежує лічильник, з одиницями, що підлягають оплаті, і застосовує ці витрати до вашого облікового запису за кожен розрахунковий період.

Коефіцієнти використання Azure та розрахункові періоди можуть відрізнятися для підписок Enterprise, Cloud Solution Provider (CSP) і Web Direct. Деякі підписки включають надбавки на використання, які впливають на вартість. Різні структури виставлення рахунків застосовуються до рідних і сторонніх служб Azure з Azure Marketplace.

Центри обробки даних Azure розташовані по всьому світу. Клієнт може використовувати різні місця, але витрати на використання відрізняються, оскільки кожне місце пропонує різні послуги, продукти та ресурси Azure відповідно до попиту, популярності та вартості місцевої інфраструктури. Наприклад, вибір місця з найнижчою ціною може не призвести до економії коштів, якщо клієнту також потрібно сплатити комісію за передачу даних у це місце.

Azure не стягує плату за вхідну передачу даних, але вимагає плату за вихідну передачу даних, яка визначається відповідно до кожної зони Azure. В Azure зона складається з кількох регіонів Azure, згрупованих для виставлення рахунків.

Таблиця 4.1 - Azure зони:

Zone 1	USA, Canada, Europe, UK
DE Zone 1	Germany
Zone 2	Japan, Korea, Australia, India
Zone 3	Brazil

У деяких зонах не стягується вартість початкових вихідних 5 ГБ на місяць, але стягується фіксована ціна за ГБ після цієї початкової передачі.

Платіжна зона – це не те саме, що зона доступності. Azure використовує термін «зона» лише для позначення виставлення рахунків. Термін «зона доступності» стосується механізму захисту від збоїв, який Azure використовує для центрів обробки даних.

Питання про те, як зменшити витрати на Azure, зазвичай виникають на третьому етапі циклу впровадження хмари, відомому як хмарна обізнаність. Після того, як організації визначають переваги створення хмари (перший етап) і розширюють використання служб хмарних обчислень (другий етап), вони зазвичай починають оптимізувати свої витрати на Azure VM.

Економія коштів не менш важлива на ранніх етапах впровадження хмарних технологій. Якщо це можливо, організації повинні запровадити стратегію контролю за витратами якомога раніше, перш ніж витрати вийдуть з-під контролю.

Якщо компанія встановлює ліміт використання для свого облікового запису Azure, Microsoft призупиняє послуги, якщо вони перевищують заздалегідь визначений місячний ліміт. Це може бути крайнім заходом, але може бути важливою гарантією запобігання надмірним витратам. Важливо пам'ятайти, що планування майбутніх проектів, потужностей і бюджетів неможливо без контролю витрат.

Є кілька причин, чому витрати на Azure можуть вийти з-під контролю. Ось три основні причини марних витрат у хмарі Azure:

- неактивні ресурси — багато облікових записів запускають неактивні ресурси, коли вони фактично не використовуються, що призводить до марнотратства. Наприклад, ресурси, які розробники не змогли закрити і більше не потрібні. Ви можете шукати та видаляти неактивні віртуальні машини Azure, екземпляри бази даних SQL Azure та масштабні набори Azure.
- загублені ресурси. Загублені ресурси виникають, коли віртуальна машина вимикається, але ресурси, пов'язані з цією машиною, все ще працюють або існують, що коштує грошей. Спробуйте визначити втратили віртуальні диски Azure, неприкріплені статичні загальнодоступні IP-адреси, втратили блочні блоки Azure та завершіть або перепризначте їх.
- надлишок ресурсів — можливо, ви платите за ресурси, які наразі не потрібні, або через те, що ваші потреби змінилися, або через те, що за замовчуванням встановлено максимальний розмір. Це означає, що ці ресурси перевантажені. Перевірте дані про використання недостатньо використовуваних екземплярів Microsoft Azure SQL Data Warehouse, екземплярів Azure SQL Database і віртуальних машин Azure. Також стежте за інструментами кешування неактивних хостів, такими як Azure Cache.

4.3 Методи оптимізації витрат Azure

Azure надає кілька інструментів для оцінки та оптимізації витрат на хмару. Ці інструменти включають:

- Azure Pricing Calculator – сервіс калькулятор цін був створений, щоб оцінити витрати на робоче навантаження перед розгортанням в Azure. Він дає змогу експериментувати з різними типами послуг або конфігураціями в калькуляторі, щоб знайти можливості для економії коштів.

- Azure Cost Analysis – ця служба в Azure була створена, для того щоб зрозуміти вартість кожного робочого навантаження. Інструмент також може передбачити майбутні витрати на основі поточної конфігурації.
- Azure Budgets – щоб реалізувати організаційні цілі, пов'язані з витратами на хмару, можна налаштувати бюджет на порталі Azure. Бюджети дають змогу встановлювати пороги витрат для користувачів і груп Azure. Коли порогове значення перевищено, служба генерує сповіщення.
- Azure Advisor – надає рекомендації щодо оптимізації витрат. Ці рекомендації зосереджені на зниженні витрат на екземпляр віртуальної машини.
- Azure cost alerts – сповіщення генеруються автоматично, коли користувачі досягають певних порогових значень використання або вартості. Існує три типи сповіщень: сповіщення про бюджет, які надходять, коли витрати досягають або перевищують заздалегідь визначену суму на основі використання ресурсів або вартості; кредитні сповіщення, що запускаються, коли передоплата Azure вичерпується; і сповіщення відділу, які надсилаються власнику відділу, коли користувач досягає певного порогу витрат.
- Azure Resource Manager – цей інструмент забезпечує виконання правил керування ресурсами Azure, наприклад, хто може створювати ресурси та як вони позначаються. Resource Manager сам по собі не оптимізує витрати, але він може зменшити витрати, уникаючи тіньових IT. Це також допомагає адміністраторам визначити робочі навантаження, які виконуються, коли вони більше не потрібні.

Якщо програми працюють у віртуальних машинах - можливо заощадити гроші на хмарному хостингу, перейшовши до контейнерів. Це пояснюється тим, що контейнери витрачають менше ресурсів на витрати віртуалізації, ніж віртуальні машини. Ви можете встановити більше контейнерів на одному хості.

Для прикладу розглянемо робоче навантаження з 12 серверами. Кожен сервер працює на різних віртуальних машинах Azure. Розгортання серверів як контейнерів за допомогою Azure Kubernetes Service (AKS) дозволяє об'єднати сервери в 3-4 віртуальні машини, значно скорочуючи витрати.

Ціни AKS на хост-сервер такі ж, як стандартні ціни Azure VM, що дозволяє заощадити до 75% витрат на хостинг. AKS коштує додатково 0,10 доларів США за годину, якщо ви виберете безперебійну роботу SLA.

Безсерверні обчислення з функціями Azure — ще один спосіб зменшити витрати. Програмне забезпечення, розгорнуте як безсерверна функція, працює на вимогу на основі визначених користувачем тригерів. Клієнти платять лише під час роботи програмного забезпечення. Безсерверні обчислення чудово підходять для періодичного виконання інтенсивних обчислювальних навантажень.

Це рентабельніше, ніж розміщення служби на віртуальній машині та оплата великої кількості невикористаних ресурсів ЦП. Виконуючи робочі навантаження на віртуальних машинах, клієнти платять за підтримку роботи віртуальної машини, навіть якщо не використовується повністю.

Ще одна найкраща практика оптимізації витрат Azure – це створення стратегії тегування ресурсів. Такі інструменти, як Azure Resource Manager, використовують цей метод.

Теги допомагають ідентифікувати хмарні ресурси. Можливо ідентифікувати користувача, який створив ресурс, і організаційний центр витрат, до якого належить ресурс. За потреби адміністратори можуть створювати список і шукати теги. Це чудовий спосіб відстежувати запущені ресурси та знаходити робочі навантаження, які можна припинити, щоб скоротити витрати.

Наприклад, можна позначати віртуальні машини, які використовуються для розробки та тестування. Надалі, коли вся компанія піде у відпустку, то можна шукати ресурси з цим тегом і закривати їх.

Сховище зазвичай становить значну частину поточних витрат на розгортання Azure. Ціни Azure Blob Storage пропонують рівні сховища Premium, Hot, Cool і Archive (холодніші рівні пропонують нижчу вартість за ГБ-місяць). і кілька варіантів резервування, де менше резервування зменшує витрати на зберігання.

Заощадження грошей, перемістивши менш конфіденційні дані або дані, до яких рідко звертаються, на більш дешевий рівень або менш зайвий варіант зберігання. Вбудувати автоматизацію багаторівневого зберігання у свої програми, щоб дані, до яких рідше звертаються, автоматично переміщувалися на дешевший рівень.

4.4 Azure Calculator

Калькулятор цін Azure може допомогти зрозуміти витрати на переміщення робочих навантажень у хмару Azure. Він оцінює ціни Azure, коли всі дані та програми знаходяться в Azure. Він відображає ціну для різних конфігурацій і розмірів віртуальних машин Azure (AVM), враховуючи пам’ять, ЦП, розташування, сховище та години використання.

Калькулятор може надати оцінку повного рішення за допомогою комбінації служб Azure. Тоді організації можуть прийняти обґрунтоване рішення, розглядаючи можливість переходу до хмари Azure або розширення своєї присутності в Azure.

Azure також надає калькулятор загальної вартості володіння, який може допомогти визначити робочі навантаження, зробити припущення та створити економічний аналіз загальної вартості очікуваного розгортання Azure у різних можливих сценаріях.

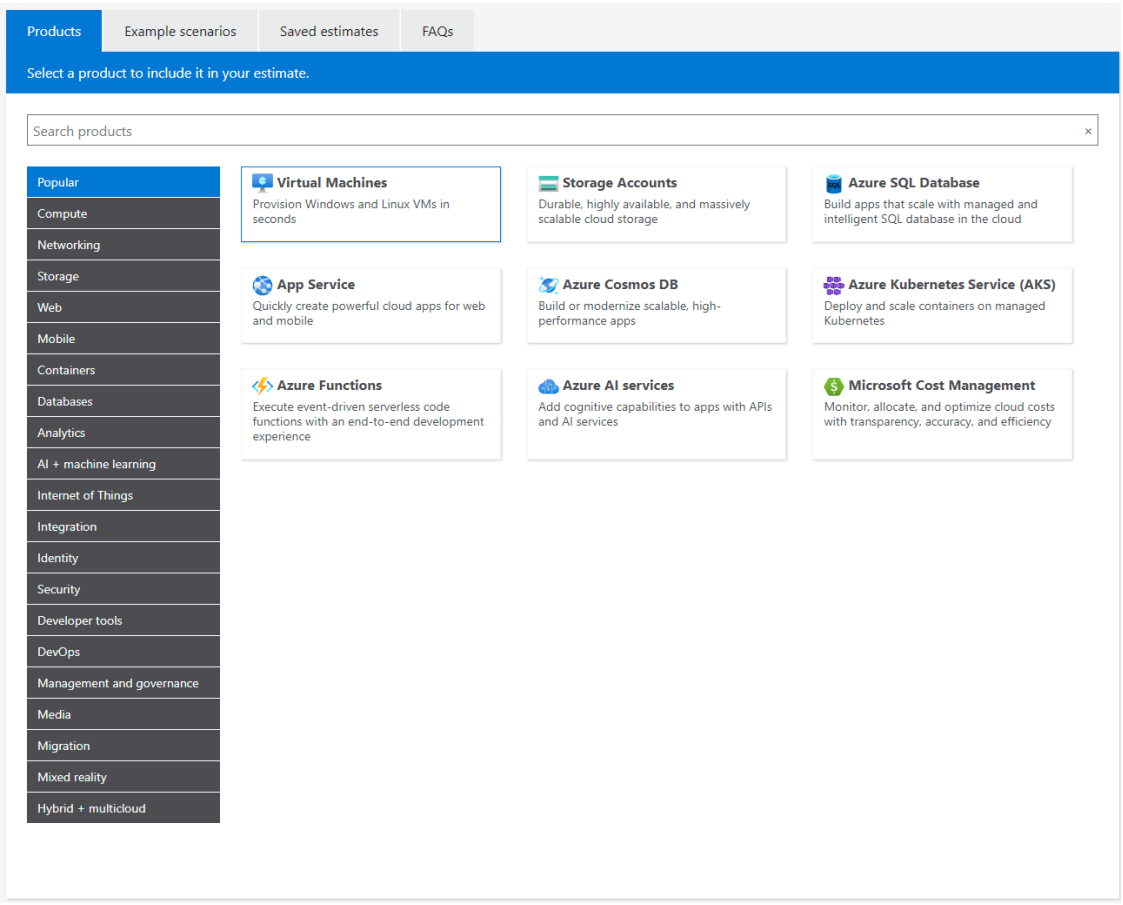


Рисунок 4.1– огляд Azure Calculator

Зараз я спробую провести поверховий разрахунок приблизної ціни мого веб додатку. Було розгорнуто два сервісу: Azure App Service и Azure SQL.

Спочатку розраховано сервіс Azure App Service.

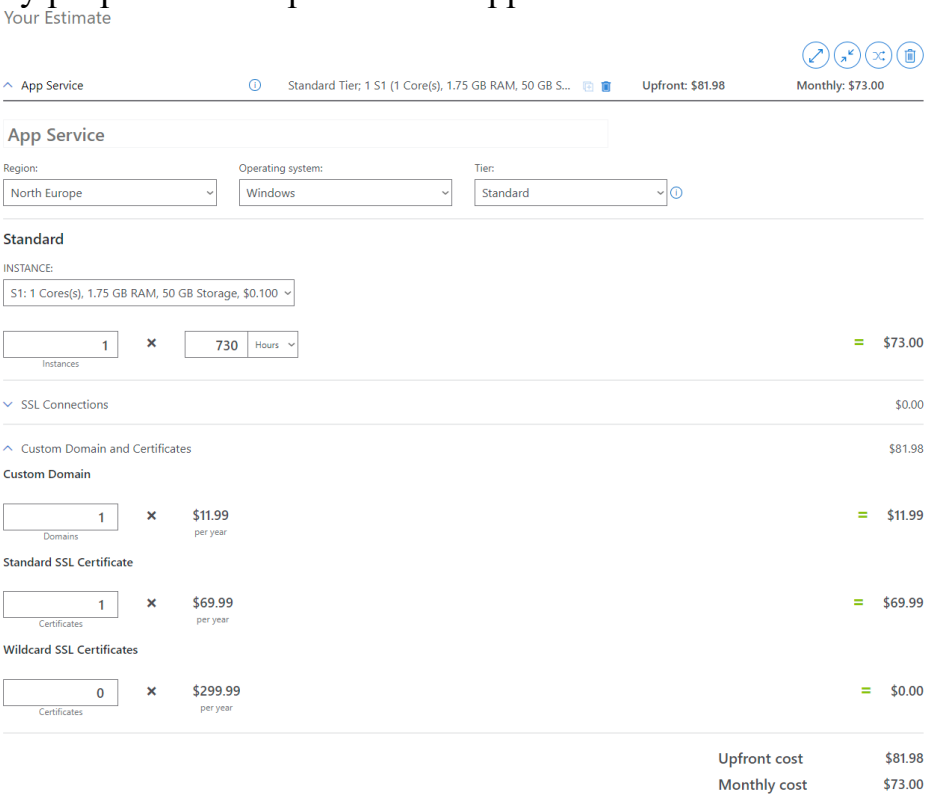


Рисунок 4.2 – розрахування вартості Azure App Service в Azure Calculator

Були обрані приблизні дані, які були зареєстровані при створені контейнеру у третьому розділі. А саме:

регіон північної Європи

операційна система Віндовс

тип надання послуг

Також була додана SSL сертифікація. Ця сертифікація потрібна для використання HTTPS протоколу. В 2024 році ця сертифікація є необхідною для просування створеного сайту браузерами та пошуковими системами. І з такими характеристиками ми отрумуємо щомісячну ціну – 73 доларів США.

Переходимо до оцінювання Azure SQL Database.

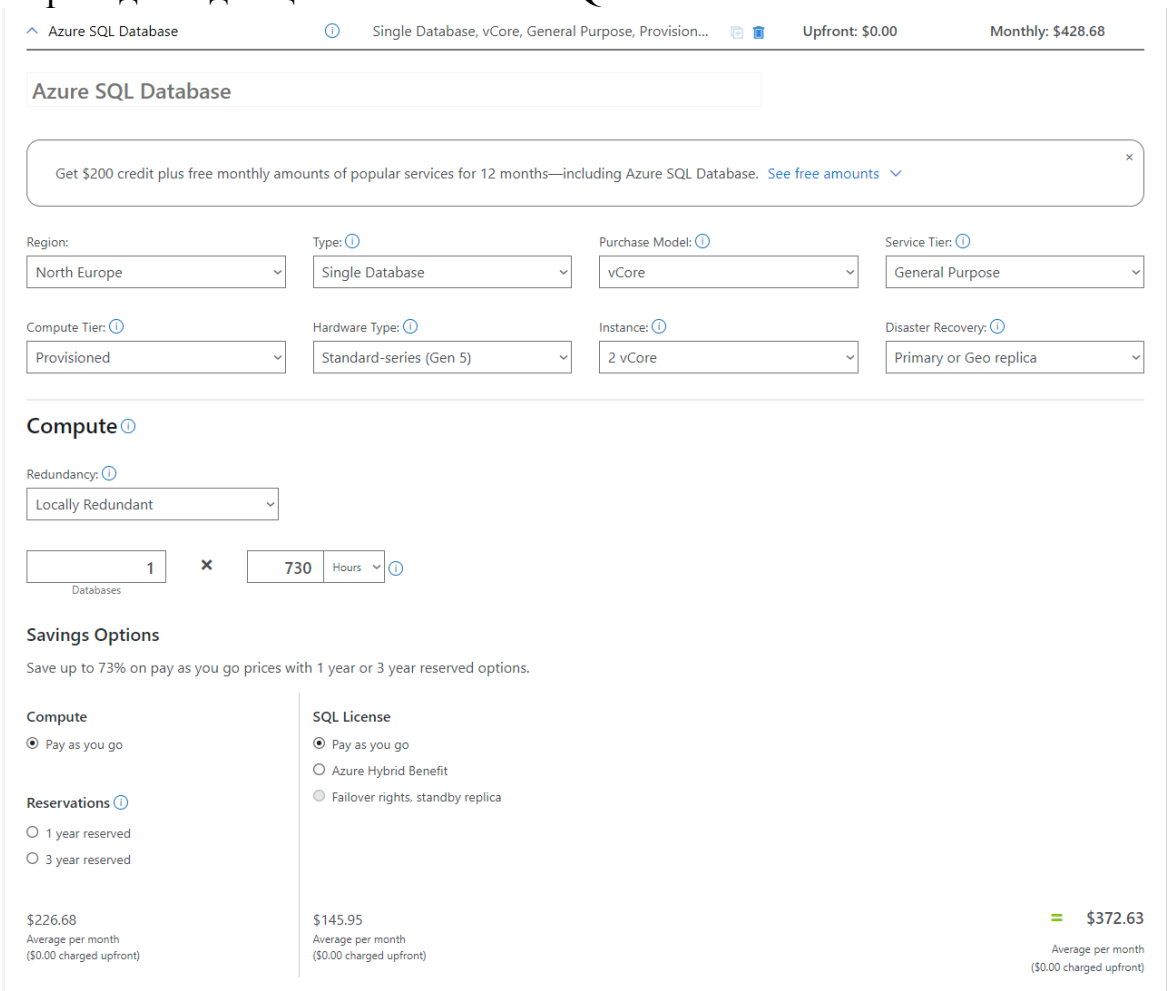


Рисунок 4.3 – розрахування вартості Azure SQL Database Compute в Azure Calculator без предоплати

Існує два типи оплати за Azure SQL Database сервіс. З можливістю часткової передоплати та платити кожен місяць по тарифу.

Спочатку прорахуємо другий тип оплати (рисунок зверху), де не вносяться попередні кошти. Microsoft не вимагає передоплати, але вимогає 428 доллари США кожен місяць. Якщо уявити ситуцію, де Azure SQL Database використовується з такою конфігурацією протягом 3 років, то ми отримуємо рахунок в 15408 долларів США тільки за Azure SQL Database сервіс.

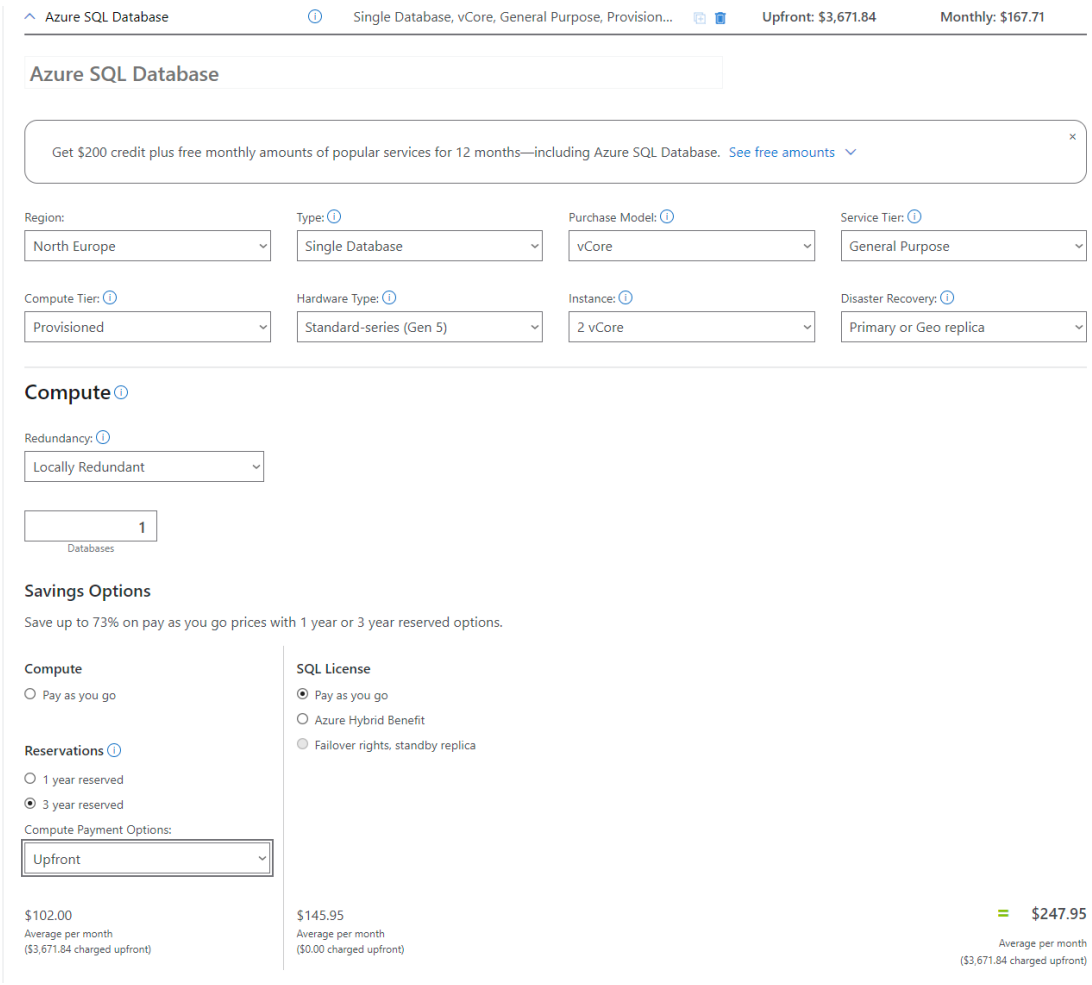


Рисунок 4.4 – розрахування вартості Azure SQL Database Compute в Azure Calculator з предоплатою

Тепер переходимо до типу з передоплатою. Обрана варіація показана на рисунку зверху. За 3 роки використання Microsoft вимогає 3671 доллар США передоплати та 167 долларів США. Тобто зашальна ціна за 3 роки: 9683 даллара США.

Також була обрана політика створення резервних данних та ємність БД:

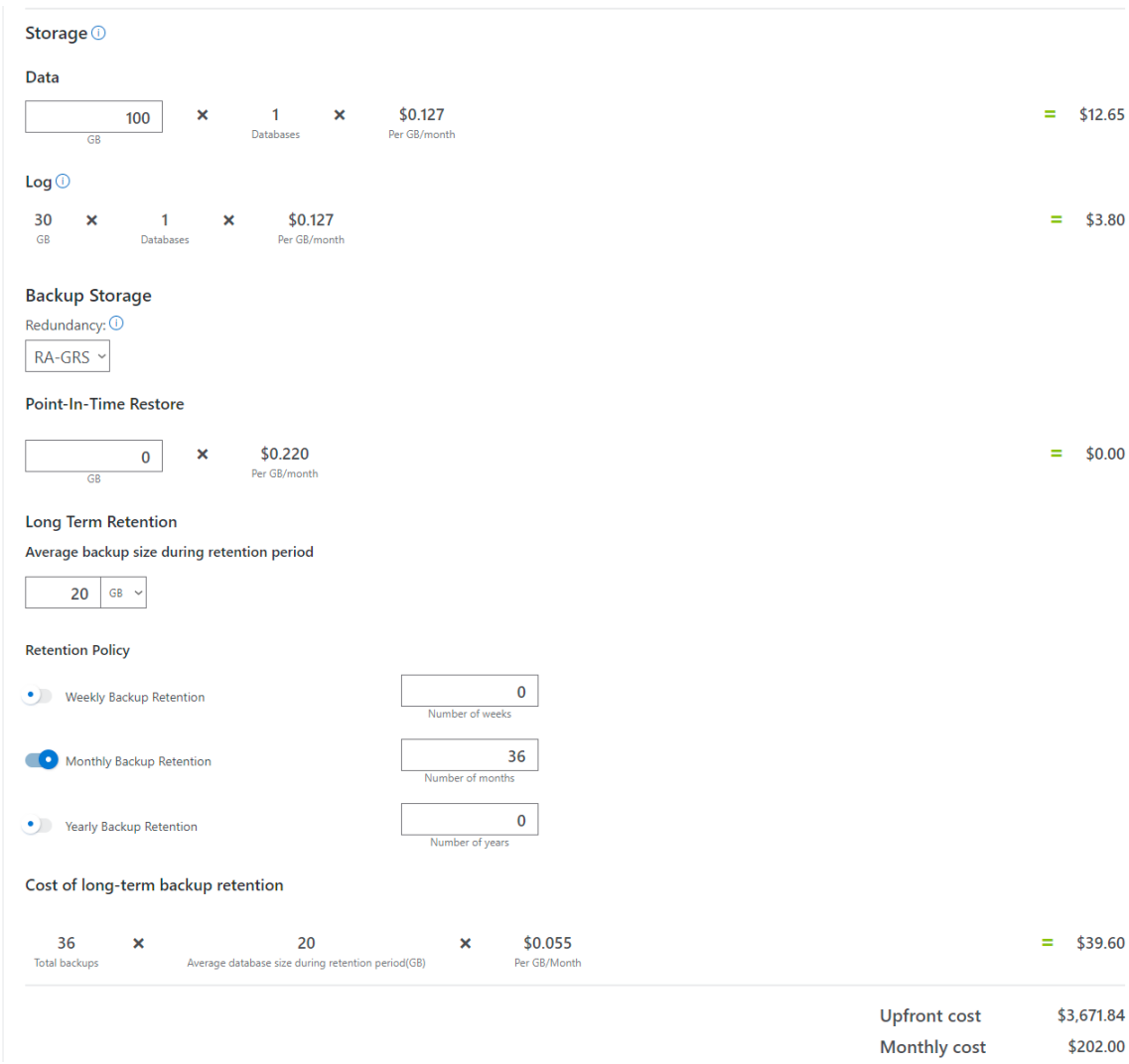


Рисунок 4.5 – розрахування вартості Azure SQL Database Storage в Azure Calculator з предоплатою

Була обрана ємність в 100 ГБ та резервне копіювання даних кожен місяць.

Висновок до розділу

В цьому розділі була оцінена вартість Microsoft Azure. Були оцінені типи ресурсів провайдеру хмарних технологій. Також були оцінені методи оптимізації витрат в Azure за допомогою вбудованих інструментів Microsoft. Таких як: Azure Pricing Calculator, Azure Cost Analysis, Azure Budgets, Azure Advisor, Azure cost alerts, Azure Resource Manager. В кінці розділу було проведено прорахування в Azure Pricing Calculator приблизної вартості розгорнутого веб додатку.

Висновки

У данній роботі магістра була проведенна комплексна та масивна робота з дослідження хмарних технологій, ринку провайдерів хмарних технологій та розгортання серверної частини веб-додатку на базі платформи .NET та хмарного провайдеру Azure.

У першому розділі досліджено основні поняття платформи .NET. Описано додаток який буде розгорнутий на платформі Azure. Розібрані основні поняття, архітектура та компоненти. Продемонстровано СУБД, використана для розробки та адміністрування БД. Промодельовано архітектуру додатку та авторизація.

Другий розділ був присвячений поняттю хмарних технологій. Розділ був поділений на 5 основних частини. У першій частині було описано моделі спільної відповідальності. У другій визначено хмарні моделі. Такі як хмарні або гібридні хмари. В інших частинах були описані різні моделі оплати та сам ринок провайдерів хмарних технологій.

В третьому розділі описан сам процес розгортання веб додатку. Обран провайдер хмарних технологій. Було вирішено використовувати Azure через тісну інтеграцію з технологіями Microsoft та в частності з платформою .NET. Було описані вимоги та можливості пробного аккаунту Azure. Такождетально проаналізовано різні сервіси Azure. Такі як: Azure SQL, Azure Portal, Azure App Services. Також розписан процес розгортання та підготовки до розгортання, окремо веб додатку та БД.

В остонньому розділі проаналізовано ціноутворення платформи Azure. Можливі методи економії. Типи ресурсів. Також промодельована можлива ціна хостингу розробленого веб додатку впродовж трьох років. Проаналізований найвигідніший можливий тариф від платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Introduction of MS SQL Server.
URL: https://www.geeksforgeeks.org/introduction-of-ms-sql-server/ (дата звернення: 24.10.2024).
2) Microsoft SQL Server: Everything you need to know.
URL: https://datascientest.com/en/microsoft-sql-server-everything-you-need-to-know (дата звернення: 24.10.2024).
3) Common web application architectures. URL: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures
4) ASP.NET MVC Overview. URL: https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview
5) What is a JWT? Understanding JSON Web Tokens. URL: https://supertokens.com/blog/what-is-jwt
6) JSON Web Tokens URL: https://auth0.com/docs/secure/tokens/json-web-tokens#:~:text=JSON%20web%20token%20(JWT)%2C,parties%20as%20a%20JSON%20object
7) What is Cloud Computing? URL: https://www.simplilearn.com/tutorials/azure-tutorial/what-is-azure#:~:text=Azure%20is%20a%20cloud%20computing,and%20resources%20provided%20by%20Microsoft.
8) Хмарна піраміда. URL: https://gigacloud.ua/ru/blog/navchannja/hmarna-piramida-iaas-paas-i-saas
9) What is SaaS? URL: https://aws.amazon.com/ru/what-is/saas/# ()
10) Cloud Spending Growth Rate Slows But Q4 Still Up By $10 Billion from 2021; Microsoft Gains Market Share. URL: https://www.srgresearch.com/articles/cloud-spending-growth-rate-slows-but-q4-still-up-by-10-billion-from-2021-microsoft-gains-market-share
11) Canalys Newsroom - Global cloud spending grows 21% in Q1 2024 as market leader AWS makes CEO change URL: https://www.canalys.com/
12) Робота Бакалвра, Мусатова А. В.: РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ СОЦІАЛЬНОЇ МЕРЕЖІ НА БАЗІ ПЛАТФОРМИ .NET