

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

СНІЖИНСЬКИЙ МАКСИМ ВАДИМОВИЧ

Допускається до захисту:

завідувач кафедри
інформаційних технологій,
доктор технічних наук, доцент

Т. В. Нескородева

« ____ » _____ 20__ р.

**РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ДОДАТКУ ВЕБ-ФОРУМУ ДЛЯ ІТ-
СПЕЦІАЛІСТІВ**

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (бакалаврська) робота

Керівник:

Мартянова Т.А., старший викладач
кафедри інформаційних технологій,
старший викладач

Оцінка:

____ / ____ / ____

(бали за шкалою ЕКТС/за національною шкалою)

Голова

ЕК:

(підпис)

АНОТАЦІЯ

Сніжинський М.В. Розробка серверної частини додатку веб-форуму для ІТ-спеціалістів. Спеціальність 122 «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2022.

Основною метою виконання кваліфікаційної роботи є створення серверної частини веб-форуму для ІТ-спеціалістів, оскільки більшість досліджень показують, що основна мета з якою люди використовують глобальну мережу Інтернет – пошук інформації.

У вступі наведено актуальність розробки додатків веб-форуму. У першому розділі розглянуті існуючі аналоги, описані їхні переваги та недоліки, проаналізована предметна область.

У другому розділі розглянуті основні інструменти та технології, які використовувались при створенні додатку, обґрунтовано вибір кожного інструменту.

Третій розділ присвячений розробці та аналізу самого програмного продукту, тобто серверної частини додатку.

Ключові слова: API, веб-форум, .NET, Server Side, Client-Server

ABSTRACT

Snizhinskyi M.V. Development of the server part of the web forum application for IT-specialists. Specialty 122 "Computer Science". Vasyl Stus Donetsk National University, Vinnytsia, 2022.

The main purpose of the qualification work is to create a server part of a web forum for IT-specialists, as most studies show that the main purpose for which people use the global Internet is to search for information.

The introduction shows the relevance of web forum application development. The first section considers the existing analogues, describes their advantages and disadvantages, analyzes the subject area.

The second section discusses the main tools and technologies used in creating the application, justifies the choice of each tool.

The third section is devoted to the development and analysis of the software product itself, i.e. the server part of the application.

Keywords: API, web forum, .NET, Server Side, Client-Server



ЗМІСТ

ВСТУП	5
ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	7
1.1 Аналіз предметної області	7
1.2 Постановка задачі	10
1.3 Огляд існуючих аналогів.....	10
Висновок до розділу 1	15
ОПИС ВИКОРИСТАНИХ ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ ДЛЯ НАПИСАННЯ СЕРВЕРНОЇ ЧАСТИНИ	16
2.1 Платформа .NET та мова програмування C#	16
2.2 Технологія ASP.NET Core.....	18
2.3 Visual Studio.....	21
2.4 Microsoft SQL Server.....	26
2.5 Entity Framework Core.....	29
2.6 Swagger OpenApi	31
2.7 ASP.NET Core Identity та JWT-токени.....	33
Висновок до розділу 2	37
РОЗРОБКА ТА АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	38
3.1 Архітектура додатку	38
3.2 Реалізація рівню доступу до даних	40
3.3 Реалізація аутентифікації/авторизації користувачів	45
3.4 Реалізація рівню доступу до API та демонстрація роботи додатку.....	50
Висновок до розділу 3	53
ВИСНОВКИ.....	54
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	55

ВСТУП

В історії людства новації в галузі технологій неодноразово надавали революційний вплив на суспільний, політичний та економічний розвиток. В період останніх десятиліть завдяки стрімкому розвитку інформаційних технологій світова спільнота вступила в епоху формування нового інформаційного простору, що створюється на базі комп'ютеризації та мережових комунікацій. Це об'єктивне явище сучасної дійсності супроводжується зростанням обсягів соціально значущої інформації, що використовується у системах управління організаційними системами з метою раціоналізації їхньої діяльності. Накопичувана у процесі розвитку суспільства інформація стає джерелом економії часу й праці. Наразі важко знайти сферу людської життєдіяльності, в якій не застосовуються інформаційні технології.

Застосування інформаційних технологій у науковій сфері та у сфері освіти складно переоцінити. Зараз важко уявити школу чи університет, в яких не було б комп'ютерного класу. Все більше людей освоюють інформаційні технології та використовують їх у повсякденному житті. У наші дні існує безліч електронних бібліотек, сайтів-помічників, веб-форумів, скористатися якими можна будь-де за допомогою смартфона, що значно полегшує процес пізнання нового. Зараз, майже на будь-яке запитання можна знайти відповідь в інтернеті, від кулінарних рецептів приготування борщу, ремонту автомобілів, побутової техніки, до вирішення складних питань у галузі фізики, хімії, математики, тощо. Майже в кожній сфері діяльності люди стикаються з проблемами які потрібно вирішувати, інформаційні технології у цьому добре допомагають, на кожную тематику зараз є безліч блогів, форумів, відео-блогів, веб-спільнот, де люди діляться своїми знаннями та допомагають одне одному вирішувати питання.

У рамках даної кваліфікаційної роботи ціль полягає в наступному:

1. Аналіз, поглиблене вивчення та застосування на практиці архітектурних підходів, програмних засобів, інструментів для створення серверних частин додатків.
2. Огляд існуючих додатків за схожою тематикою, виявлення їх переваг та недоліків.
3. Створення програмного продукту, а саме серверної частини додатку веб-форуму для ІТ-спеціалістів.

Задачами дослідження є:

- Огляд існуючих додатків, опис їх основних переваг та недоліків.
- Вивчення та застосування різних підходів та інструментів для створення відповідного програмного забезпечення.
- Розробка серверної частини додатку веб-форуму для ІТ-спеціалістів.

Об'єкт дослідження: серверна частина додатку веб-форуму для ІТ-спеціалістів.

Предмет дослідження: аналіз та процес розробки серверних додатків.

Структура роботи: кваліфікаційна робота складається із вступу, трьох розділів, висновку та списку використаних джерел.

Робота містить 57 сторінок, 1 таблицю, 41 рисунок та список використаної літератури із 31 джерела.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Розвиток сучасного суспільства на даний момент відбувається під впливом глобалізації інформаційних процесів, які людство відчуває все сильніше з кожним днем. Ці процеси проявляються у вигляді формування нового інформаційного простору, що тісно пов'язаний з появою глобальної мережі Інтернет. Формуються нова культура та нова наука. Нове інформаційне суспільство принципово відрізняється від попередніх типів, головні цінності в ньому не матеріальні, а духовні – знання, пізнання та інформація. Інформація та технологія її використання у життєдіяльності суспільства становлять основу сучасності. Інформатизація суспільства охоплює весь світ, все більшою мірою набуваючи глобального характеру. Нижче приведений графік кількості користувачів глобальної мережі Інтернет[1]:

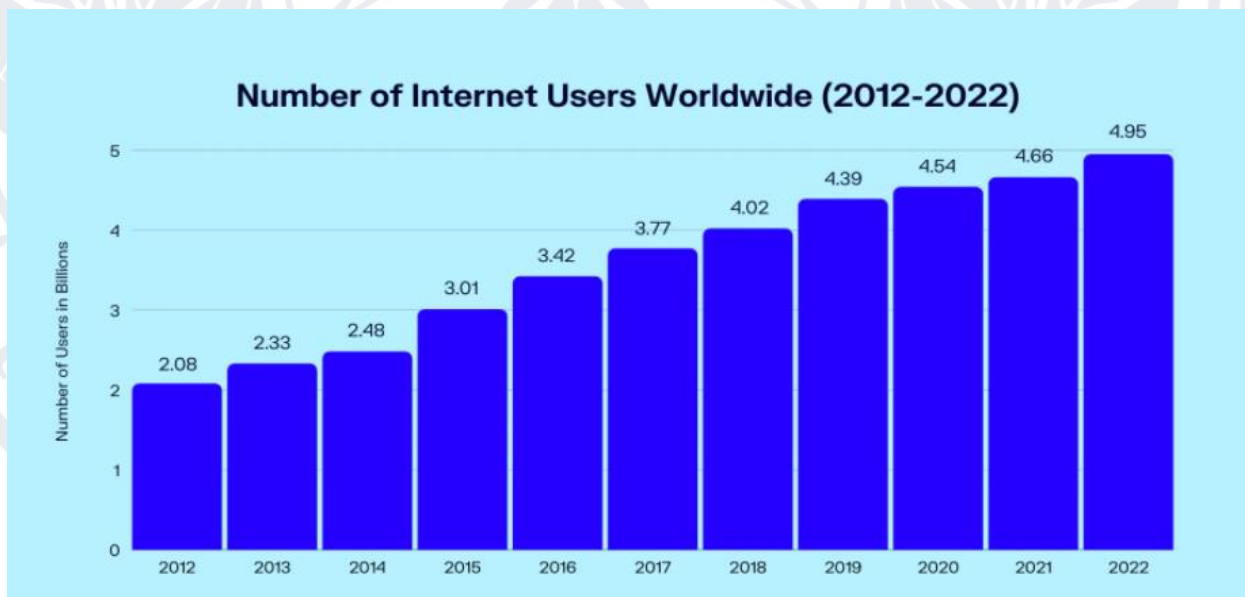


Рисунок 1.1 – графік кількості користувачів в мережі Інтернет

Аналізуючи графік видно, що з 2012 року кількість людей які користуються мережею інтернет зросла більш ніж у двічі і становить майже 5

мільярдів активних користувачів. Ці явища зумовлено високими темпами розвитку науки і техніки, розвитком засобів масової комунікації, змішання різних культур. Тенденція до глобалізації та віртуалізації є особливо характерною для сучасного суспільства.

Соціолог Говард Рейнгольд порівняв кіберпростір з чашками Петрі, в яких, як колонії мікроорганізмів, виростають і множаться віртуальні спільноти, що характеризуються при цьому просторовою уявою і відчуттям місця. Дуже високе значення в Інтернеті займає соціалізація, що включає такі елементи[2]:

- предметна область – коло інтересів користувача;
- соціальні ролі – ролі, що приміряються користувачем (пізнавальна, комунікативна та ін.);
- соціальні функції – ціль перебування користувача в конкретній інтернет-спільноті;
- соціальні конфлікти – природне явище будь-якого типу суспільства (віртуальне не виняток), що виникає у процесі тривалої комунікації, викликане розбіжністю думок, а також перерозподілом соціальних ролей в суспільстві.

За даними дослідження аналітичної компанії GlobalWebIndex[3], яка надає інформацію про аудиторію видавцям, медіа-агентствам та маркетологам по всьому світу, пошук інформації – основна причина користування інтернетом. Так кажуть майже дві третини користувачів інтернету у світі. Дані у вигляді графіку наведено нижче:

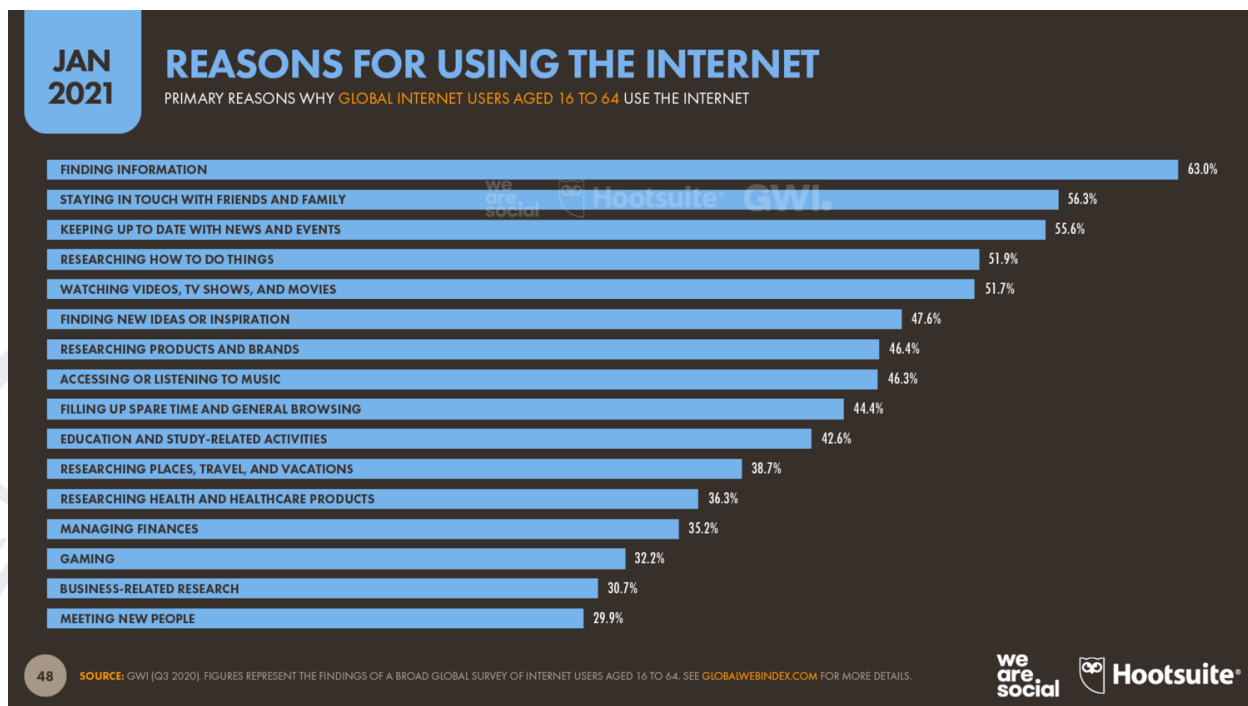


Рисунок 1.2 – причини використання інтернету

На графіку видно, що для 63% відсотків користувачів основне призначення мережі Інтернет – це пошук інформації. Для майже 52% - пошук відповідей на запитання у конкретних сферах діяльності. Для 42,6% користувачів інтернет допомагає із навчанням. Звідси можна зробити висновки, що для більшості активних користувачів – інтернет це те місце, де можна знайти відповіді на свої питання.

У глобальній мережі Інтернет відбувається активне формування локальних груп – інтернет-спільнот, членами яких можуть бути користувачі з усього світу. Веб-форуми є однією з найпопулярніших і найефективніших у плані представлення інформації в Інтернеті формою організації веб-спільнот. Основними об'єктами веб-форуму є учасники та інформаційне наповнення, яке вони створюють: дискусії та повідомлення. Учасником веб-спільноти вважається людина, яка відвідує сайт веб-спільноти, читає або публікує інформацію у вигляді дискусій та повідомлень. Інформаційне наповнення веб-форуму – це ієрархія підфоруму та повідомлень, дерево з вузлами двох типів: підфоруми та дискусії.

1.2 Постановка задачі

Необхідно реалізувати серверну частину, а саме REST-арі, додатку веб-форуму для ІТ-спеціалістів. Додаток повинен виконувати наступні функції:

1. Реєстрація та логін.
2. Створення особистого кабінету користувача.
3. Отримання даних про інших користувачів додатку.
4. Створення системи питань і відповідей із наступними можливостями:
 - а. Задання теми питання
 - б. Опис проблеми
 - с. Прикріплення додаткових матеріалів(програмного коду, скріншотів, тощо)
 - д. Додавання спеціальних міток.
 - е. Можливість інших користувачів системи надавати свою відповідь під опублікованим запитанням.
5. Перегляд найпопулярніших запитань.
6. Пошук запитань за ключовими словами.

1.3 Огляд існуючих аналогів

Спеціалізовані веб-форуми не є чимось інноваційним, перші веб-сайти такого плану почали з'являтися ще в кінці 1990-х років та стрімко розвиваються по сьогоднішній день.

Розглянемо найбільш популярні схожі за призначенням веб-спільноти для ІТ-спеціалістів.

1. Reddit

Reddit[4] - сайт, що поєднує риси соціальної мережі та форуму, де зареєстровані користувачі можуть розміщувати будь-яку вподобану інформацію(статті, відео, посилання на зовнішні ресурси) на платформі та обговорювати її або ж створювати власні пости. Reddit підтримує систему голосування за повідомлення, що сподобалися - найбільш популярні з них опиняються на великій сторінці сайту. Один з найпопулярніших сайтів у світі - 21-е місце за відвідуваністю за даними Alexa Internet[5] та SimilarWeb.

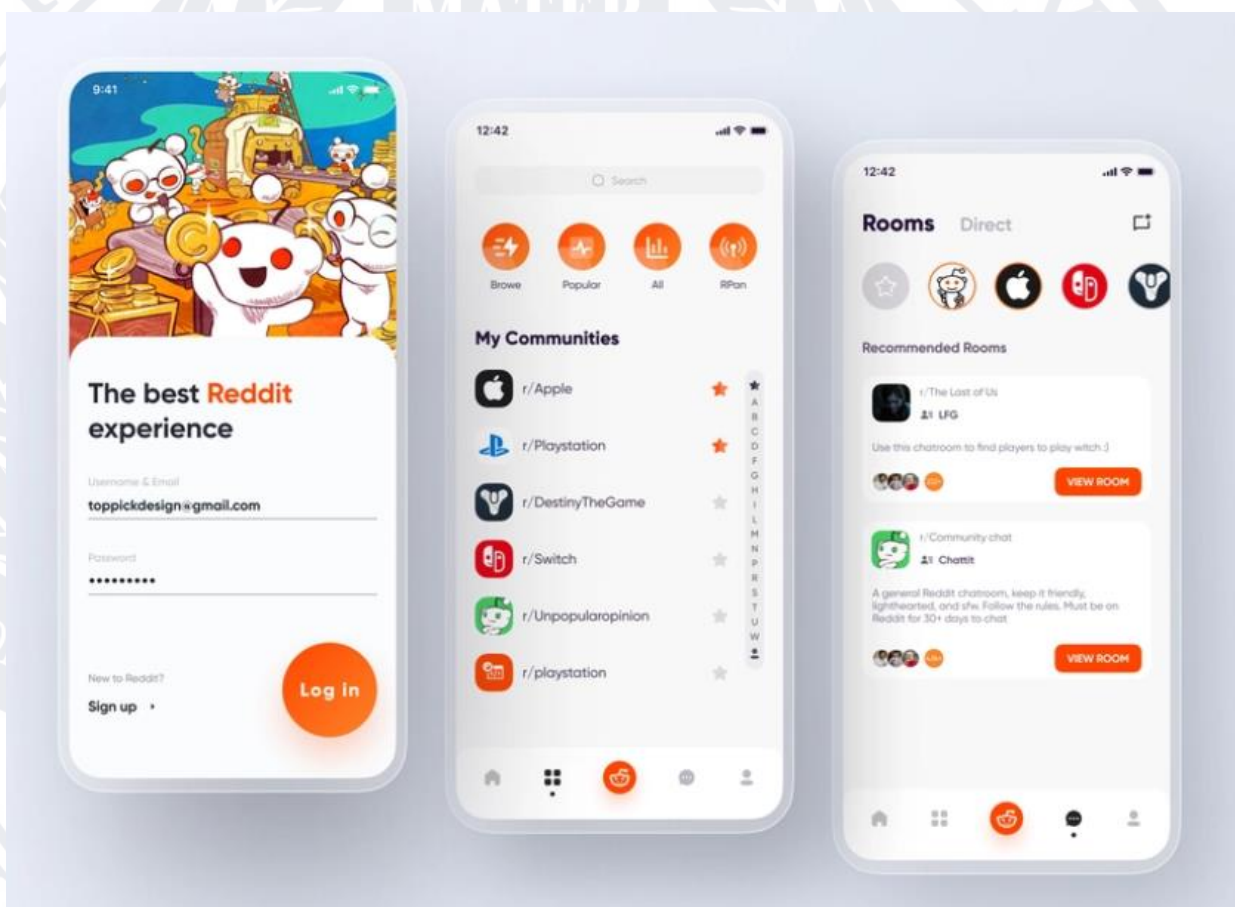


Рисунок 1.3 – сервіс Reddit

Весь сайт ділиться на «сабредіти» – групи за інтересами чи просто спільноти, об'єднані ідеєю. Спочатку повинен був існувати один головний «реддіт», проте ідея не прижилася. «Сабредіт» /r/popular включає найголовніші записи всіх груп, виключаючи деякі спірні «сабредіти». Користувач вільний сам вибирати, що його цікавить - у цьому сенсі можна провести аналогію з YouTube

та його каналами. Або з LJ та стрічкою новин груп. Загалом, все дуже просто, хоч і незвично.

Завдяки величезній популярності portalу, багато користувачів використовують його для привернення уваги до того чи іншого питання, або частіше сайту. Багато сайтів стикалися з так званим «Slashdot effect», коли одне посилання на Reddit (іноді – навіть просто в коментарях) призводить до перевантаження серверів сайту через наплив відвідувачів.

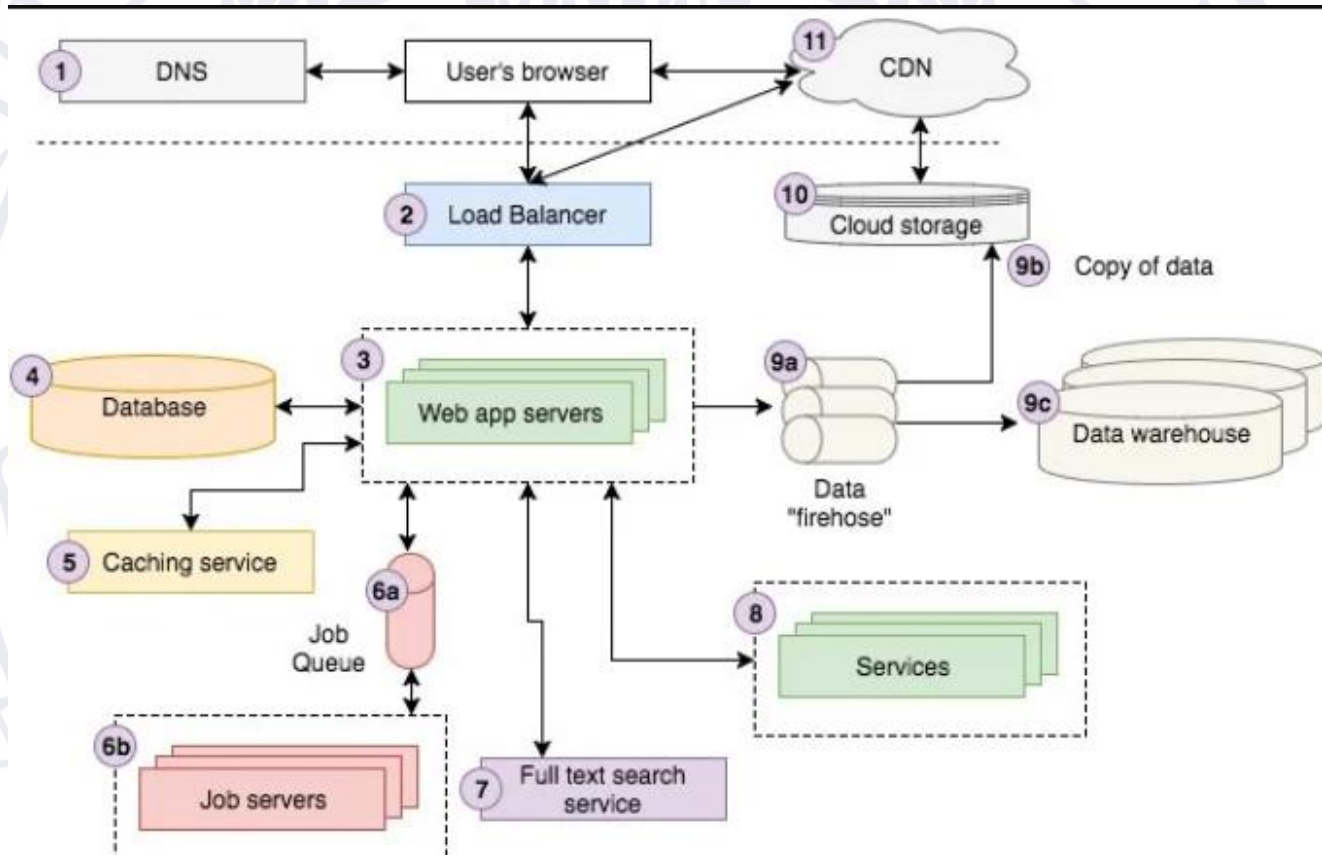


Рисунок 1.4 – архітектура сервісу Reddit

Переваги: зручний інтерфейс; мультиплатформенність; локалізація; гарантії анонімності; велика аудиторія.

Недоліки: є платна підписка; велике навантаження сервісу; Reddit – форум загального призначення, а не спеціального призначення; не користується попитом в Україні.

2. StackOverflow

Stack Overflow[6] — система питань та відповідей про програмування, розроблена Джоелем Спольські та Джеффом Етвудом у 2008 році. Є частиною Stack Exchange Network. Як і в інших системах подібного роду, Stack Overflow надає можливість оцінювати питання та відповіді, що порушує або знижує репутацію зареєстрованих користувачів (варіант ігрофікації). На липень 2014 року в базі даних веб-сайту зберігалось майже 7,7 мільйона питань. У вісімку найбільш популярних тем, згідно з тегами, на Stack Overflow входять Java, C#, JavaScript, PHP, Android, jQuery, Python і HTML. За даними на травень 2017 року, сайт займає 53 місце за відвідуваністю у всьому світі за рейтингом Alexa Internet[7], а кількість унікальних відвідувачів становила у грудні 2010 року 18 мільйонів осіб. У 2016 році опубліковані результати наукового дослідження, що показало, що розробники програм для Android, які використовують Stack Overflow, створюють значно функціональніший код (але менш безпечний), ніж розробники, які використовують офіційну документацію.

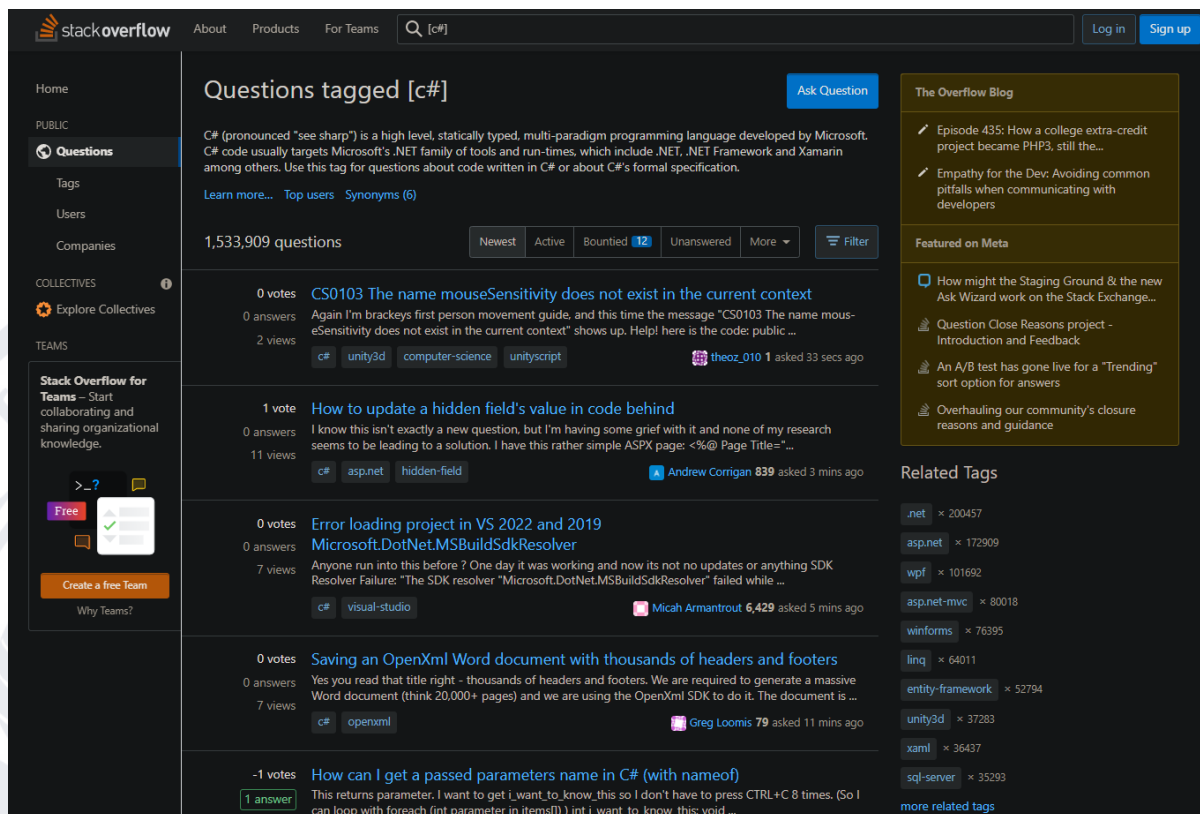


Рисунок 1.4 – сервіс StackOverflow

Переваги: зручний та інтуїтивно зрозумілий інтерфейс; велика база питань та відповідей; велика аудиторія.

Недоліки: високе навантаження на сервіс; система рейтингу, в якій нові користувачі мають мало прав, а отже їхні пости мають нижчий пріоритет в пошуковій системі ніж інші; погано реалізований додаток для мобільних пристроїв.

3. Habr Q&A

Habr[8] (колишній Хабрахабр) — веб-сайт у форматі системи тематичних колективних блогів (звані хабами), створений для публікації новин, аналітичних та технічних статей, створення віртуальних обговорень цікавих тем поглядами інших людей на інформаційні технології, бізнесом, інтернетом, тощо. Контент сайту формується користувачами-добровольцями, які пишуть у колективні та

персональні блоги, публікують подкасти, перекладають іноземні статті, проводять опитування (голосування) та спілкуються з іншими користувачами.

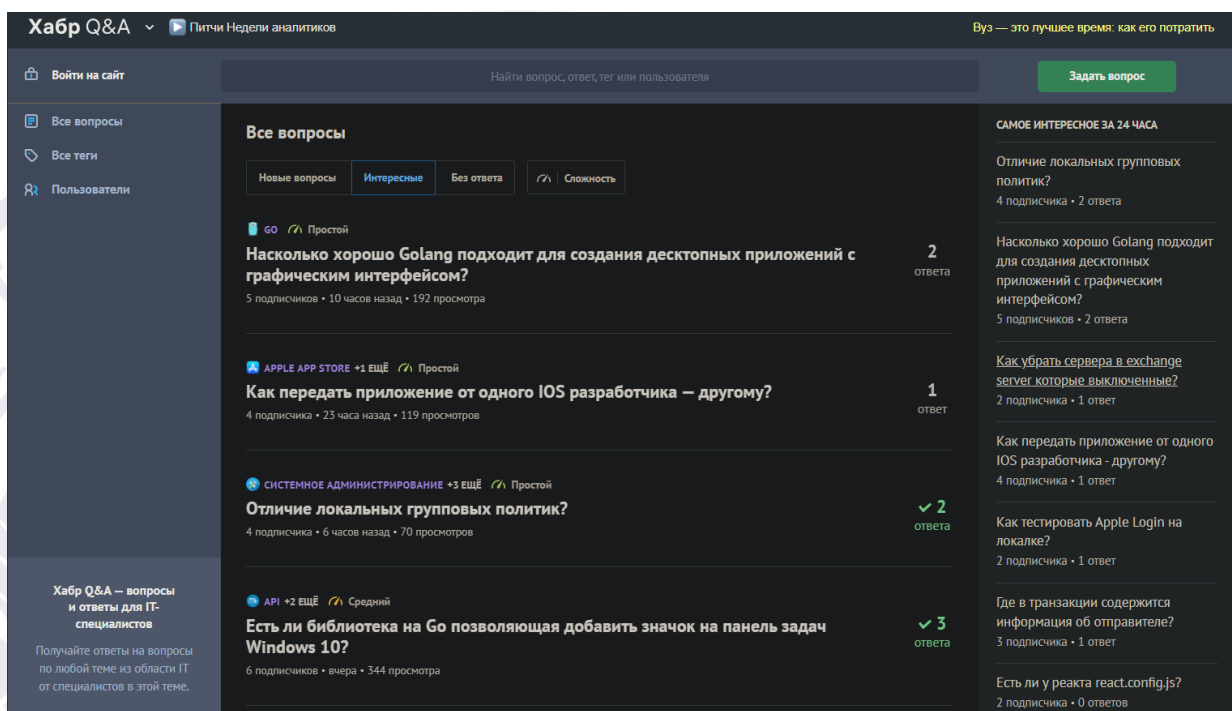


Рисунок 1.5 – сервіс Habr Q&A

Розділ «Питання та відповіді» запущений восени 2010 року, був влаштований за аналогією з «Google Питання та Відповіді» для пошуку відповідей на питання з ІТ-тематики.

Переваги: зручна система стоврення питань та відповідей

Недоліки: мобільна версія майже не підтримується розробниками; застарілий інтерфейс; невелика аудиторія; частково недоступний на території України.

Висновок до розділу 1

В даному розділі було розглянуто актуальність веб-форумів, їхню потребу на сьогоднішній день. Також, були описані необхідні функції програми, її альтернативи на даний момент, їх переваги та недоліки.

РОЗДІЛ 2

ОПИС ВИКОРИСТАНИХ ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ ДЛЯ НАПИСАННЯ СЕРВЕРНОЇ ЧАСТИНИ

Вибір правильного стеку технологій – фактор успіху реалізації проекту. Даний розділ присвячений огляду ключових інструментів та технологій для вирішення поставлених задач, їх порівняння, переваги та недоліки. Будуть описані та порівняні різні підходи до створення додатків такого типу.

2.1 Платформа .NET та мова програмування C#

.NET[9] — це безкоштовна платформа для створення програмних продуктів з відкритим кодом, що дозволяє створювати різні типи додатків. Середовище .NET дозволяє розробляти додатки одразу на кількох мовах і забезпечує всіма необхідними інструментами для створення настільних додатків, ігор, додатків у сфері машинного навчання.

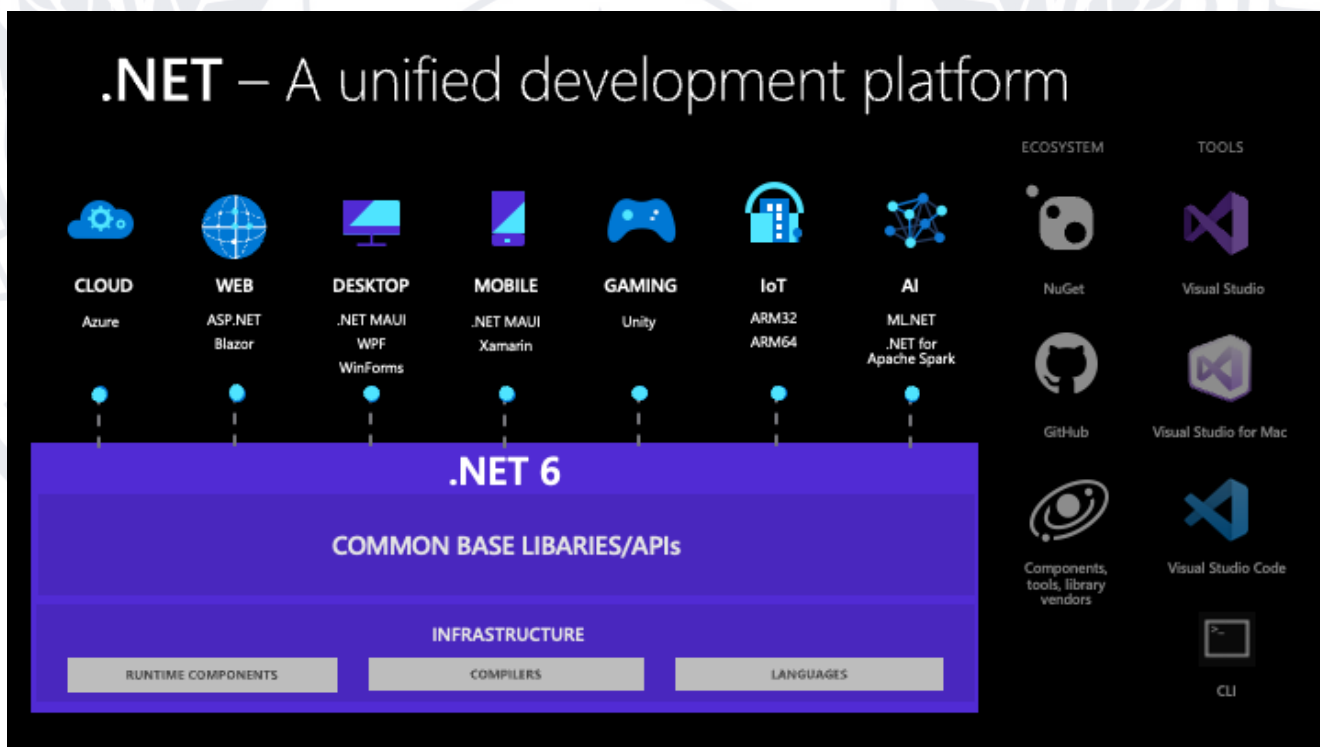


Рисунок 2.1 – можливості платформи .NET

Для створення додатків на цій платформі можна використовувати одну із трьох підтримуваних мов програмування:

- C# — це проста, сучасна, та дуже потужна об'єктно-орієнтована та безпечна для типів мова програмування.
- F# — це мова програмування, яка дозволяє легко писати стислий, надійний і продуктивний код у функціональному стилі.
- Visual Basic — це проста та доступна мова для створення безпечних для типів об'єктно-орієнтованих програм. Її головний недолік — важкий синтаксис.

На даний момент, C# та F# є основними мовами програмування, Visual Basic практично не розвивається. Для створення веб-додатків краще підходять мови із об'єктно-орієнтованим підходом, тому розглянемо C#.

C#[10] — це потужна об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Мова розроблена Андерсом Гейлсбергом, Пітером Гольде, та Скотом Вілтамутом в компанії Microsoft Research (належить Microsoft). Синтаксис C# дуже схожий до C++ і Java. Основними перевагами мови є статична типізація, підтримка поліморфізму, підтримка делегатів, атрибутів, подій, добре реалізований механізм інкапсуляції за допомогою властивостей та методів `get/set`, обробки виключних ситуацій. Розробники C# вирішили взяти все найкраще від таких популярних мов як Java, C++, SmallTalk, виключити деякі їхні моделі та ділянки, що зарекомендували себе як проблематичні. Наприклад, мова C# не підтримує механізм множинного спадкування, що підтримується в C++. Станом на 2022 рік поточна стабільна версія мови C# 10.0, яка була випущена в 2021 році як частина платформи .NET 6.0.

За даними дослідження аналітичної групи PYPL[11], мова програмування C# займає 4 місце у рейтингу популярності мов програмування та демонструє незначний зріст.

Worldwide, Apr 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.95 %	-2.2 %
2		Java	18.09 %	+0.6 %
3		JavaScript	9.14 %	+0.5 %
4		C#	7.39 %	+0.5 %
5		C/C++	7.06 %	+0.4 %
6		PHP	5.48 %	-0.9 %
7		R	4.41 %	+0.5 %
8	↑↑	TypeScript	2.27 %	+0.5 %
9	↓	Objective-C	2.23 %	-0.3 %
10	↓	Swift	2.06 %	+0.2 %

Рисунок 2.2 – рейтинг популярності мов програмування

2.2 Технологія ASP.NET Core

ASP.NET Core[12] (Active Server Pages Core для .NET) — популярний фреймворк, що входять до платформи .NET і призначений для розробки веб-додатків, що включає в себе безліч механізмів та інструментів для розробки веб-сервісів, програмної інфраструктури, як серверної так і клієнтської частини додатків. ASP.NET Core є логічним продовженням старої технології Microsoft ASP. Фреймворк побудований із набору відносно незалежних компонентів, середовище вже забезпечує розробника такими механізмами як DI-контейнером, системою маршрутизації, парсингу моделей, логування, тощо. Можна або

використати вбудовану реалізацію цих компонентів, або розширити їх за допомогою механізму наслідування, або створити і застосовувати свої компоненти зі своїм функціоналом.

Моделі розробки веб-додатків за допомогою ASP.NET Core[13]:

- Базовий ASP.NET Core, який підтримує всі основні механізми та компоненти, необхідні для роботи сучасного веб-додатку: маршрутизація, конфігурація, логування, можливість роботи з різними системами баз даних і т.д..
- ASP.NET Core MVC представляє у загальному вигляді побудову програми навколо трьох основних компонентів - Model (моделі), View (представлення) та Controller (контролери). Згідно такого підходу моделі відповідають за роботу з даними, їх відображення; контролери відповідають за логіку обробки запитів та систему маршрутизації; представлення визначають візуальну складову за допомогою мов, Blazor, React, WebAssembly та інших.
- Razor Pages представляє модель розробки, у якій запити обробляють спеціальні сутності - сторінки Razor Pages. Кожну окрему таку сутність можна порівнювати з окремою веб-сторінкою.
- ASP.NET Core Web API - представляє реалізацію патерну REST та сервісів RESTful, у якому кожному типу http-запиту (GET, POST, PUT, DELETE) призначений окремий ресурс. Такі ресурси визначаються як методи контролера Web API.

- Blazor представляє модель розробки, яка дозволяє створювати інтерактивні програми як на стороні сервера, так і на стороні клієнта за допомогою спеціального синтаксису та механізмів шаблонізації та дозволяє задіяти на рівні браузера низькорівневий код WebAssembly.

ASP.NET Core Architecture

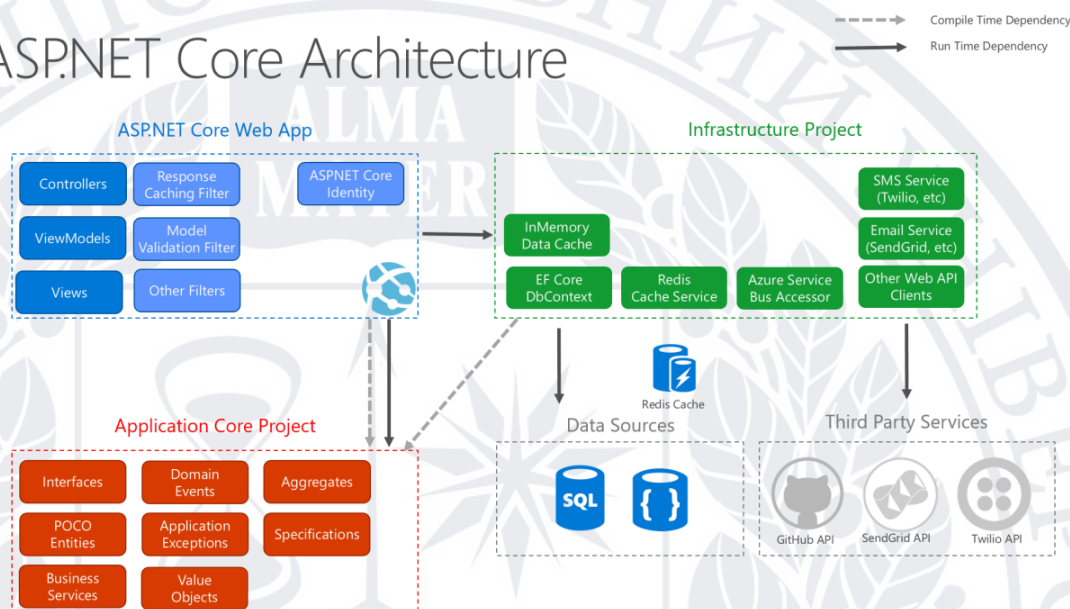


Рисунок 2.3 – архітектура ASP.NET Core додатку

На сьогоднішній день однією із найпопулярніших моделей створення веб-додатків за допомогою ASP.NET Core є ASP.NET Core Web API, її основні переваги:

- Реалізація патерну REST.
- Забезпечує достатню гнучкість у створенні веб-API за рахунок узгодження вмісту та забезпечує підтримку маршрутизації ASP.NET.
- Архітектура веб-API дуже легка, що робить її ідеальною альтернативою для розробників, що хочуть створювати програми для пристроїв з обмеженою пропускнуою здатністю.
- Використання клієнт-серверного підходу
- Підтримує шаблони URL-адрес і методи HTTP.
- Підтримує прив'язку моделі та її валідацію.

- Кросплатформеність: можливість розробки та розгортання додатків на Windows, Mac, Linux
- Вбудована підтримка впровадження залежностей

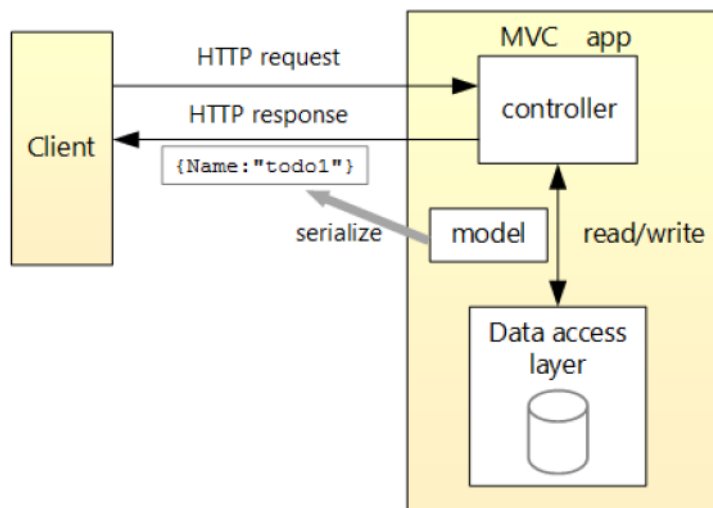


Рисунок 2.4 – архітектура ASP.NET Core Web Api додатку

2.3 Visual Studio

Microsoft Visual Studio[14] — інтегроване середовище розробки програмного забезпечення від компанії Microsoft, також включає ряд інших продуктів. Ці продукти дозволяють розробляти як консольні додатки, так і ігри та додатки з графічним інтерфейсом, у тому числі з підтримкою технологій UWP, Windows Forms, а також веб-сайти, веб-додатки, веб-служби.

Visual Studio представляє собою редактор вихідного коду з підтримкою технології IntelliSense, що значно полегшує процес розробки, форматування коду і можливість рефакторингу коду. Вбудований відладчик забезпечує розробника всіма необхідними інструментами для відладки коду. Останні вбудовувані інструменти включають в себе форми редактора для просування створення графічного інтерфейсу додатків, веб-редактор, дизайнер класів і дизайнерські схеми баз даних. Visual Studio дозволяє створювати та підключати сторонні доповнення (плагіни) для розширення функціональності практично на кожному рівні, включаючи додаткову підтримку системи контролю версії вихідного коду

(як, наприклад, Git, GitLab, GitHub), додавання нових наборів інструментів (наприклад, для редагування та візуального проектування коди предметно-орієнтованих мов програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server). Visual Studio є кросплатформеною, дозволяє створювати додатки на багатьох популярних операційних системах: Windows, Mac, Linux. На сьогодні, актуальною версією IDE є Visual Studio 2022

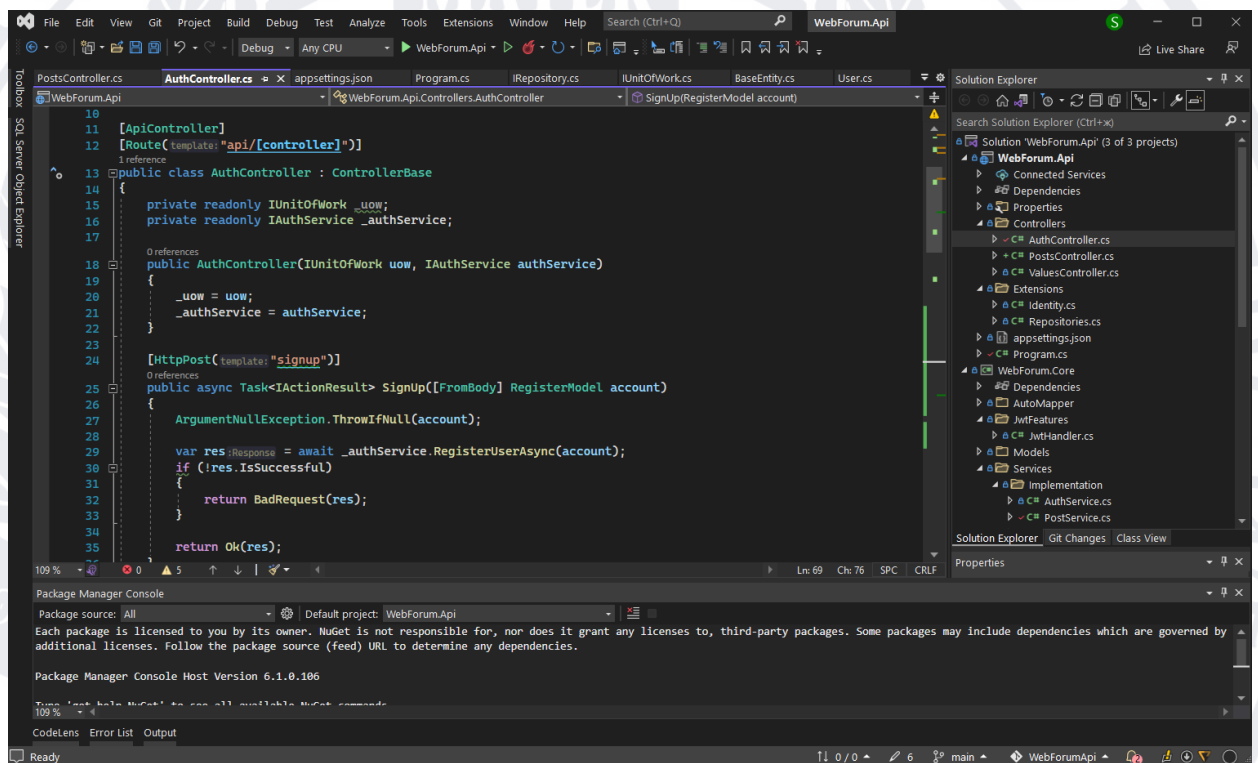


Рисунок 2.5 – середовище Visual Studio

Visual Studio 2022 розповсюджується із пакетним менеджером Visual Studio Installer, який допомагає розширити можливості стандартної IDE, встановити додаткові пакети, розширення для інших мов програмування, технологій, тощо. Наприклад, для того, щоб почати розробляти веб-додатки різних рівнів, потрібно всього лиш встановити відповідний компонент з пакетного менеджера.

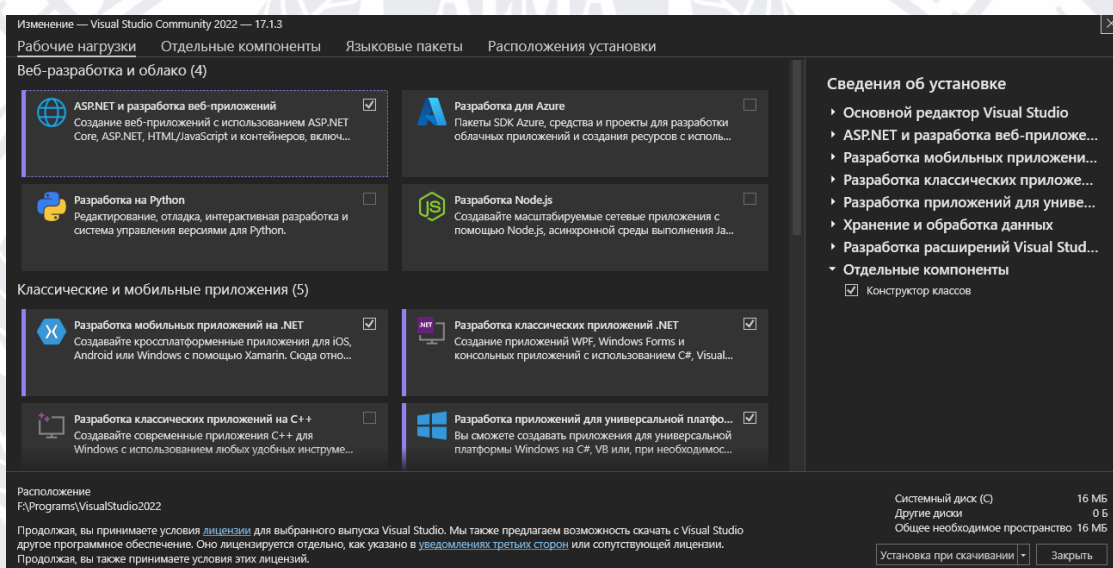


Рисунок 2.6 – Visual Studio Installer

IDE Visual Studio також забезпечує розробника всім необхідним для роботи із базами даних. Вбудований компонент SQL Server Object Explorer дозволяє легко створювати нові, підключатись до існуючих баз даних і забезпечує всіма необхідними інструментами для їх моніторингу та менеджменту.

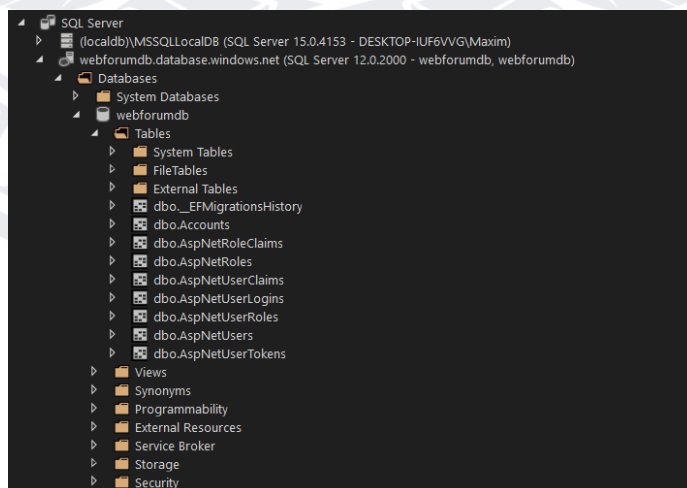


Рисунок 2.6 – підключення до існуючої БД

Visual Studio Debugger[15] – потужний інструмент, що дозволяє тестувати та налагоджувати код. Налагоджувач є важливим інструментом для пошуку та виправлення помилок у програмах. Ця технологія дозволяє покроково виконувати код для виявлення в ньому помилок, забезпечує моніторинг та аналізом ресурсів в конкретний момент виконання додатку та використання пам'яті ПК. Важливо використовувати всі доступні інструменти, щоб швидко усувати помилки. Іноді правильний «інструмент» може бути кращою практикою кодування.

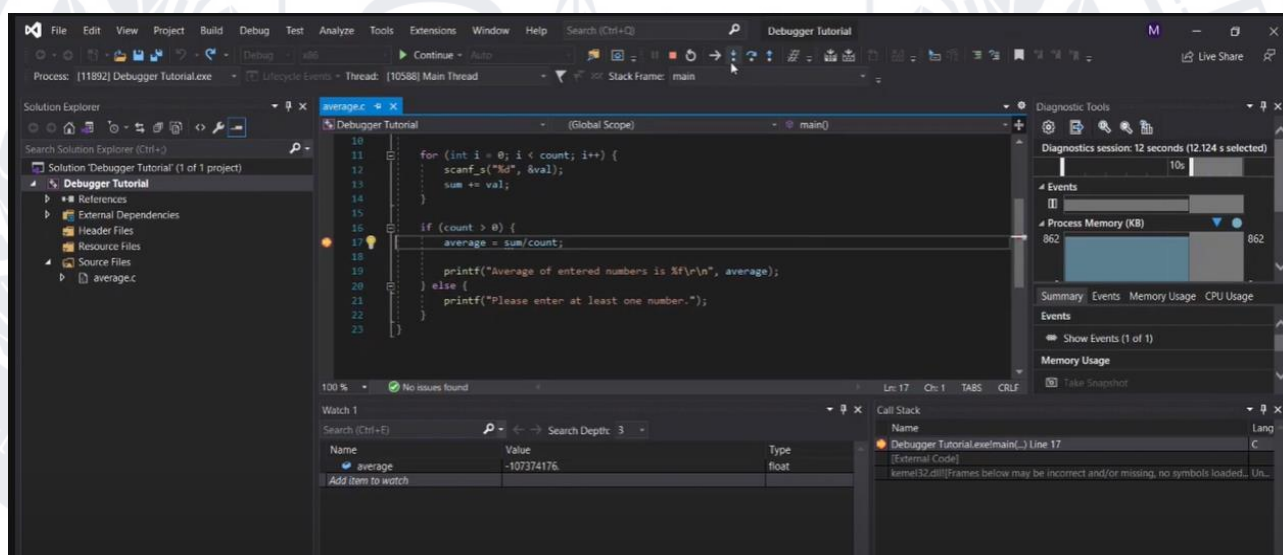


Рисунок 2.7 – запущений додаток в режимі налагодження

Git – дуже зручна та сучасна система контролю версії, яка зараз використовується практично повсюдно. Вона потрібна для того, щоб команди могли кооперуватися у створенні програмного продукту та поєднувати коди, написані різними людьми в одних репозиторіях. Також Git може виступати як хмарне середовище для зберігання коду. Середовище Visual Studio має зручний графічний інтерфейс для взаємодії із різними системами контролю версій та покриває весь функціонал, який потрібний розробнику.

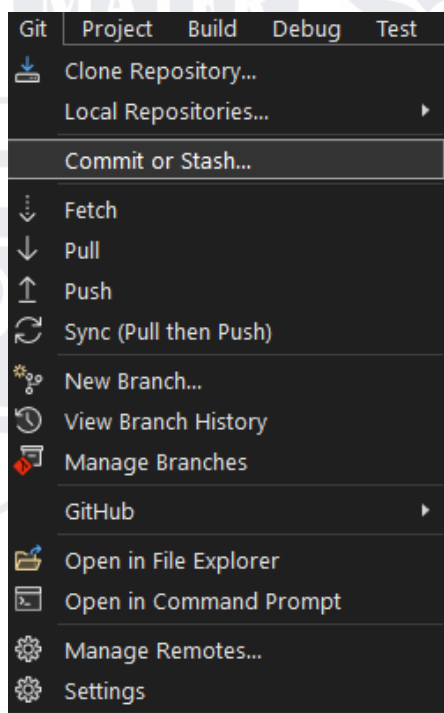


Рисунок 2.8 – робота з Git в середовищі Visual Studio

Для розробки в команді існує компонент Team Explorer[16], який дозволяє легко за потребою перемикатися між системами контролю версій, переглядати локальні коміти, створювати нові гілки та вирішувати конфлікти за допомогою графічного середовища у вигляді порівняння конфлікуючих комітів.

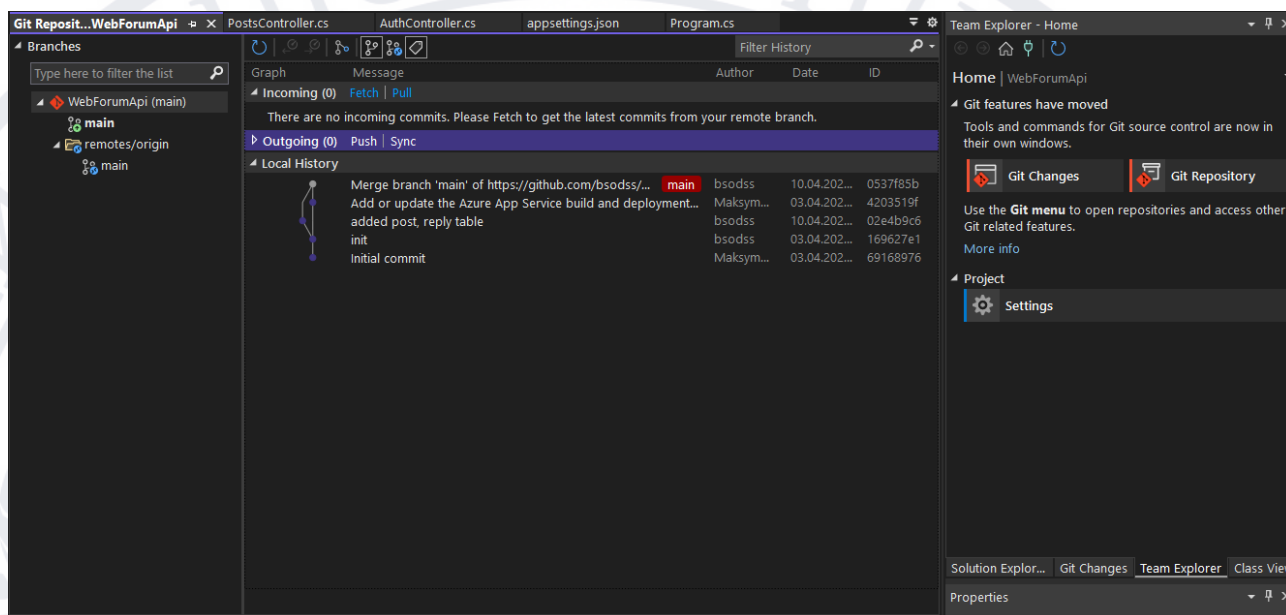


Рисунок 2.9 – Team Explorer

2.4 Microsoft SQL Server

Microsoft SQL Server[17] – популярна система управління реляційними базами даних (СКБД), розроблена корпорацією Microsoft. Для мови запитів використовується розширена версія SQL і реалізацією стандарту ANSI/ISO — Transact-SQL, який був розроблений спільно із Sybase.. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства; конкурує з іншими СКБД у цьому сегменті ринку.

Як і інші технології СУБД SQL Server в першу чергу будується навколо строкової структури таблиці, яка з'єднує пов'язані елементи даних різних таблиць один з одним, уникаючи необхідності надлишково зберігати інформацію в кількох місцях. Реляційна модель також забезпечує посилову цілісність та інші обмеження цілісності для підтримки точності. Ці перевірки є частиною більш широкого дотримання принципів атомарності, узгодженості, ізоляції та довговічності, які в сукупності відомі як властивості ACID і призначені для забезпечення надійної обробки транзакцій.

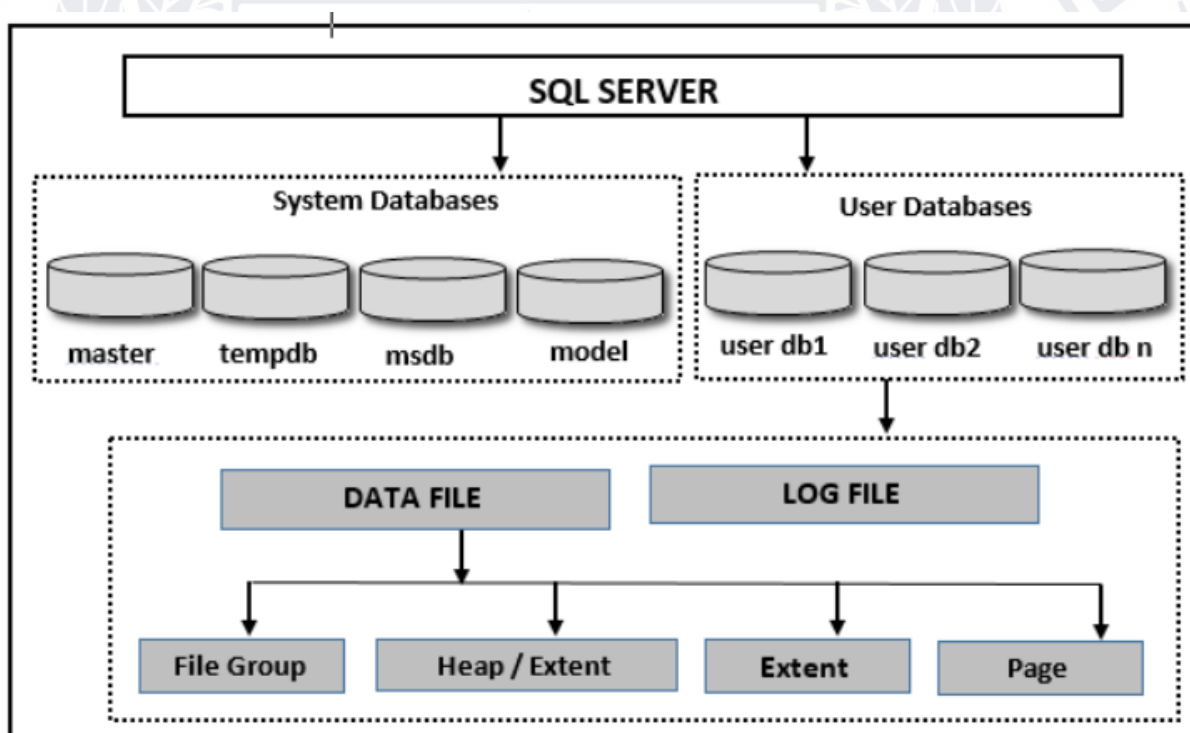


Рисунок 2.10 – схема роботи SQL Server

SQL Server Database Engine є основним компонентом Microsoft SQL Server, який контролює зберігання, обробку та відповідає за безпеку даних. Він включає в себе реляційний механізм, який обробляє команди та запити, і механізм зберігання, який керує файлами бази даних, сторінками, таблицями, індексами, буферами даних і транзакціями.

Процес роботи SQL Database Engine розділений на такі етапи:

1. Написаний SQL-код відправляється.

2. Відбувається перевірка синтаксису та правильність мовних конструкцій.
3. Компілятор перевіряє чи інсує вже план виконання, якщо ні то створює новий.
4. План виконання поміщається в кеш.
5. SQL Engine починає виконувати план та отримує дані.
6. Отримані дані переміщується із пам'яті в сховище.
7. Дані повертаються користувачеві.

Схематично це виглядає так:

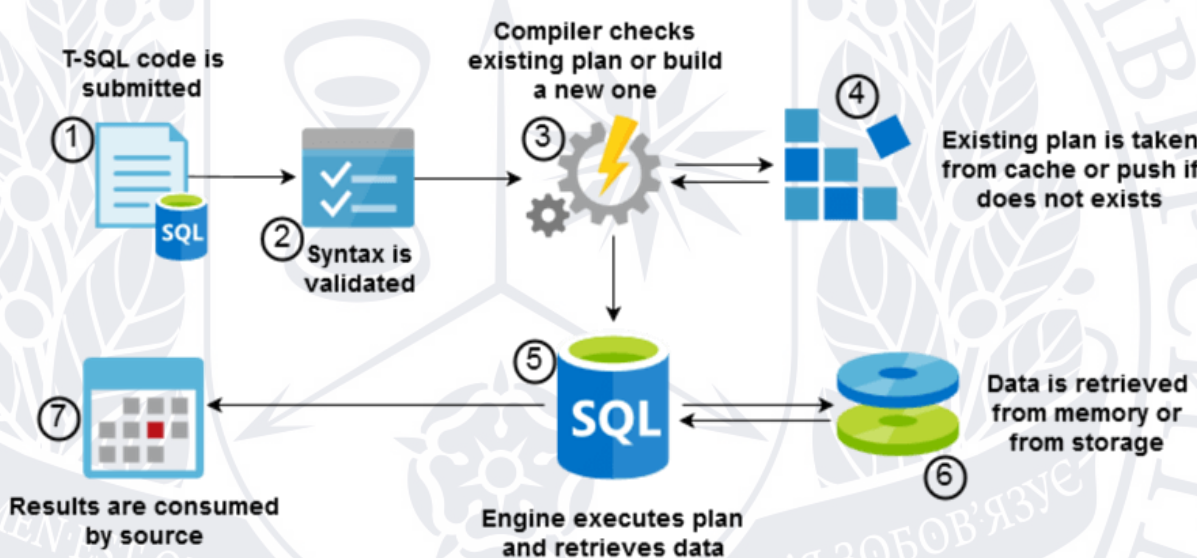


Рисунок 2.11 – схема роботи SQL Server Engine Database

Розширені функції безпеки включають три технології: Always Encrypted, яка дозволяє користувачеві оновлювати зашифровані дані без необхідності розшифровувати їх на рівні першого рядка, що забезпечує захист даних доступ для контролю на рівні рядків у таблицях бази даних; і динамічне маскування даних, яке автоматично приховує елементи конфіденційних даних від користувачів без повного доступу. Також важливою перевагою SQL Server є прозоре шифрування даних, що дозволяє шифрувати файли даних у базах даних, і тонкий аудит, який збирає детальну інформацію про використання бази даних для звітування про відповідність нормативним вимогам. Microsoft також надає

підтримку протоколу безпеки транспортного рівня для захисту зв'язку між клієнтами SQL Server і серверами баз даних[18].

2.5 Entity Framework Core

Entity Framework Core (EF Core)[19] є об'єктно-орієнтованою, легковажною і розширюваною технологією від компанії Microsoft для доступу до даних. EF Core є O/RM-інструментом (object-relational mapping – відображення даних на реальні об'єкти). Цей потужний інструмент дає можливість працювати з базами даних, використовуючи більш високий рівень абстракції: EF Core дозволяє абстрагуватися від самої бази даних та її таблиць та працювати з даними незалежно від типу сховища. Якщо на фізичному рівні розробники оперують таблицями, первинними та зовнішніми ключами, індексами. EF Core пропонує розробникам новий та дуже потужний підхід до взаємодії із базами даних – концептуальний, який дозволяє працювати з таблицями БД у об'єктно-орієнтованому вигляді. Entity Framework Core дозволяє працювати з різними системами баз даних, для цього потрібно лише підключити відповідний провайдер.

EF Core надає універсальний механізм для роботи з даними – FluentApi. Коли виникає потреба змінити цільову СУБД, то основні зміни в проекті будуть мати місце насамперед щодо конфігурації та підключення до відповідних провайдерів. Код, який безпосередньо працює з даними, отримує їх, та додає у БД і т.д., залишиться попереднім. Існує два основних підходи до створення БД в середовищі EF Core:

1. Code First. Це підхід на основі коду. EF Core API створює базу даних і таблиці за допомогою міграції на основі стуностей(класів) які описані в коді розробником. Такий підхід використовують при проектуванні систем у дизайні, керованому доменом (DDD).

2. Database First. Це підхід на основі бази даних, при якому EF Core API створює домен і класи контексту на основі існуючої бази даних, для цього потрібно лише підключити відповідний провайдер БД та виконати кілька команд EF Core. Такий спосіб обмежено підтримується в EF Core, так як інструмент не підтримує візуальний дизайнер. Для використання Database First краще користуватися технологіями СКБД і вже потім підключатися засобами EF Core.

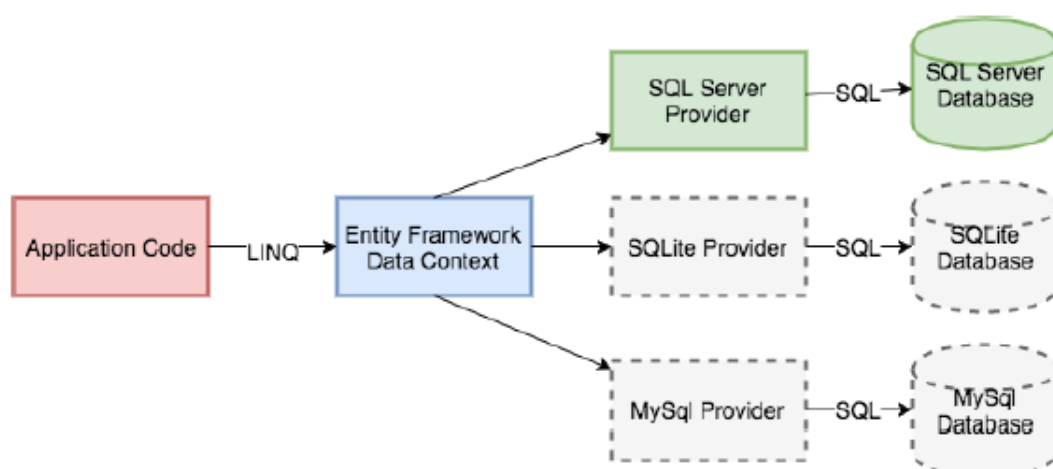


Рисунок 2.12 – схема принципу роботи EF Core

Отже, EF Core є O/RM-фреймворком, що дозволяє працювати із базами даних на концептуальному рівні, який:

- Дозволяє розробникам .NET працювати з базою даних в об'єктно-орієнтованому вигляді, використовуючи всю міць ООП.
- Для підключення до бази даних потрібно всього лиш передати стрічку підключення, що усуває потребу в написанні вручну більшості коду доступу до даних.
- EF Core дозволяє легко змінити СКБД, використовуючи провайдери.

2.6 Swagger OpenApi

Swagger[20] — це потужний, але простий у використанні набір інструментів для розробників API для команд і окремих людей, що дає змогу розробляти весь життєвий цикл API, від проектування та документації до тестування та розгортання. Він складається з поєднання безкоштовних і комерційно доступних інструментів з відкритим кодом, які дозволяють кожному, від технічних інженерів до вуличних розумних менеджерів продуктів, створювати дивовижні API, які всім подобаються. Swagger створено компанією SmartBear Software, лідером інструментів якості програмного забезпечення для команд. Його основними цілями є:

- Зведення до мінімуму обсяг роботи, необхідної для підключення відокремлених послуг.
- Скорочення кількості часу, необхідного для точного документування послуги.

Swagger а складається з 4 основних компонентів:

1. Swagger Core – ядро фреймворку, що дозволяє генерувати документацію на основі існуючого коду, а також за допомогою XML-коментарів.
2. Swagger Codegen – дозволить генерувати клієнтів для існуючої документації.
3. Swagger UI – користувацький інтерфейс(графічний), який представляє документацію до написаного коду. Забезпечує можливість перегляду, які типи запитів є, опису моделей та їх типів даних, а також безпосередньо відправляти запити на сервер та отримувати відповідь.
4. Swagger Editor - Дозволяє писати документацію в форматі YAML або JSON

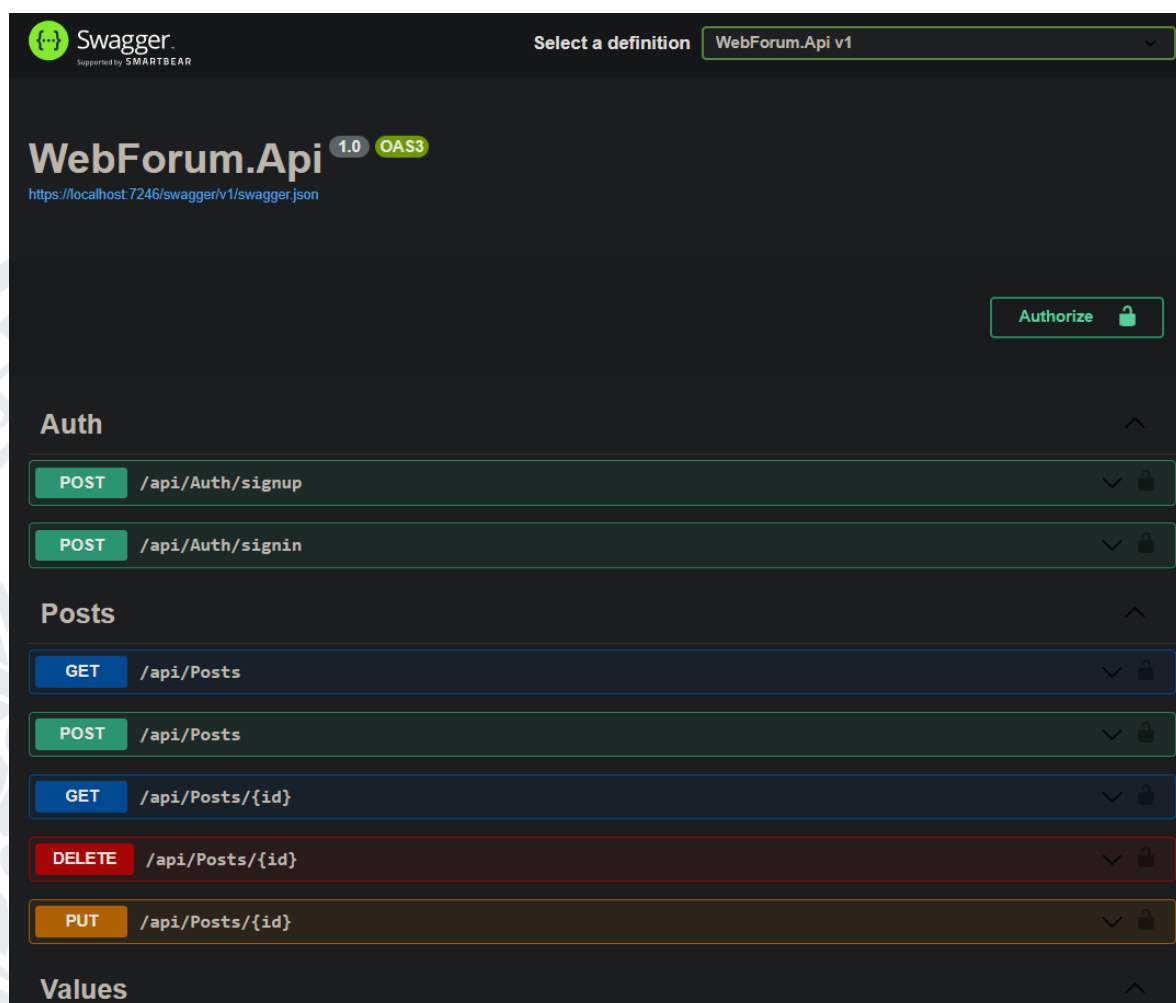


Рисунок 2.12 – документація згенерована Swagger

Середовище Swagger забезпечує розробника двома підходами написання документації:

- Документація пишеться на основі коду.
 - Даний підхід є найпростішим. Достатньо додати кілька залежностей у проект, додати конфігурацію і вже будемо мати потрібну документацію.
 - Код проекту стає не дуже читабельним від перевантаження анотаціями і описом в них.
 - Вся документація буде вписана в код (всі контролери та моделі переходять в .NET Swagger Code)

- Підхід не радять використовувати, якщо є можливість, але його дуже просто інтегрувати.
- Документація пишеться окремо від коду.
 - Даний підхід вимагає знань синтаксису Swagger Specification.
 - Документація пишеться або в файлі YAML/JSON, або в редакторі Swagger Editor.

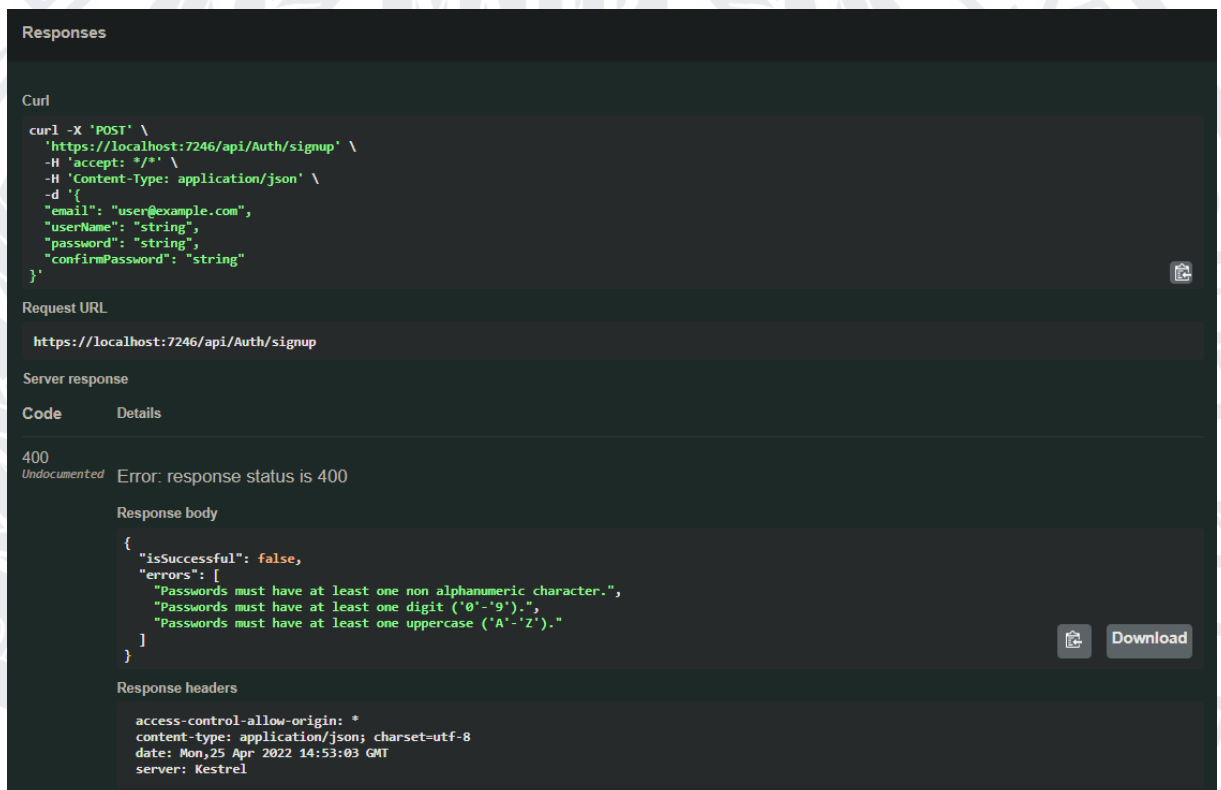


Рисунок 2.13 – приклад виконання запиту в середовищі Swagger

2.7 ASP.NET Core Identity та JWT-токени

Механізм здійснення аутентифікації та авторизації користувачів на сьогоднішній день є чи не одним із найважливіших критеріїв успішного функціонування сучасних додатків. Хоча аутентифікація та авторизація часто використовуються як взаємозамінні, вони є окремими процесами, які використовуються для захисту організації від кібератак. Оскільки, за даними досліджень все більше персональних даних користувачів попадають до рук

зловмисників, саме аутентифікація та авторизація є першою лінією захисту, яка дозволяє запобігати цьому. Надійні методи аутентифікації та авторизації мають бути важливою частиною загальної стратегії безпеки кожної організації. Отже, яка різниця між аутентифікацією та авторизацією?

Простіше кажучи, аутентифікація — це процес перевірки, ким є користувач, тоді як авторизація — це процес перевірки, до яких конкретних програм, файлів і даних користувач має доступ. Процес схожий на ситуацію з авіакомпанією, яка повинна визначити, які люди можуть потрапити на борт. Перший крок – це підтвердити особу пасажир, щоб переконатися, що він є тим, за кого себе видає. Після визначення особи пасажир другим кроком є перевірка будь-яких спеціальних послуг, до яких пасажир має доступ, чи то політ першим класом, чи відвідування VIP-зали. У цифровому світі аутентифікація та авторизація досягають цих же цілей[21]. У таблиці 2.1 наведена порівняльна характеристика цих двох понять.

Таблиця 2.1 – порівняльна характеристика понять аутентифікації та авторизації

	Аутентифікація	Авторизація
Для чого призначена?	Перевіряє облікові дані	Надає або відхиляє дозволи
Як працює?	Через паролі, біометричні дані або програми	За допомогою налаштувань, які підтримують служби безпеки
Доступність для користувачів	Так	Ні
Є можливість змінювати користувачем	Частково	Немає

Спосіб передачі даних	Через ID-токени	Через токени доступу
-----------------------	-----------------	----------------------

В середовищі ASP.NET Core є всі необхідні інструменти для створення механізмів аутентифікації та авторизації. ASP.NET Core Identity являє собою вбудовану в ASP.NET Core систему аутентифікації та авторизації. Дана система дозволяє користувачам створювати вхідні записи, аутентифікуватися, керувати вхідними записами або використовувати для входу на сайт облікові записи зовнішніх ресурсів, таких як Google, Meta, Microsoft, Twitter та інші. Основні механізми, якими забезпечує ASP.NET Core Identity[22]:

- Підтвердження облікового запису: ASP.NET Core Identity дозволяє підтвердити обліковий запис, підтвердивши електронну адресу користувача. Ця функція використовується на більшості веб-сайтів для підтвердження своєї електронної пошти перед доступом до їхніх облікових записів і служб. Це також запобігає створенню фіктивних облікових записів.
- Блокування облікового запису: якщо користувач введе слово та два коди факторів неправильно, то після певної кількості недійсних спроб, яку може налаштувати розробник, його або її обліковий запис буде заблоковано на певний період часу.
- Двофакторна аутентифікація: двофакторна аутентифікація додає ще один рівень безпеки для веб-програми.
- Role Provider: технологія, що дозволяє обмежувати доступ до частин програми за ролями. Можна створювати ролі, наприклад "Адміністратор", і додавати користувачів до ролей.
- Простота підключення даних профілю про користувача. Розробник має контроль над схемою інформації про користувача та профілю.

Наприклад, можна легко дозволити системі зберігати дати народження, введені користувачами під час реєстрації облікового запису.

JSON Web Token (JWT)[23] — це відкритий стандарт (RFC 7519[24]), що надає компактний і автономний спосіб безпечної передачі інформації між сторонами як об'єкт JSON. Ці дані можна верифікувати, оскільки JSON-токен має цифровий підпис. Для підпису можна використовувати секретний алгоритм HMAC або пари відкритих і закритих ключів за допомогою RSA або ECDSA. Використовуючи механізми відкритих/закритих ключів, підпис також засвідчує, що підписала його лише сторона, яка видає ключ, тобто володіє закритим ключем.

Веб-токени JSON складаються з трьох частин, розділених крапками (.):

- Header
- Payload
- Signature



Рисунок 2.14 – приклад JWT-токену

Для того, щоб почати використовувати JWT-токени в середовищі ASP.NET Core Identity потрібно вставити пакет JwtBearer із пакетного менеджера NuGet.

Висновок до розділу 2

У даному розділі розглянуті основні інструменти та технології для швидкої та комфортної розробки веб-додатків, наведена їх порівняльна характеристика. Розглянуті основні бібліотеки та підходи для побудови багатофункціонального додатку.

РОЗДІЛ 3

РОЗРОБКА ТА АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

3.1 Архітектура додатку

Multilayered Architecture(Багатошарова архітектура)

Багатошарова архітектура[25] програмного забезпечення є одним із найпопулярніших архітектурних шаблонів сьогодні. Це пом'якшує зростаючу складність сучасних додатків. Багаторівнева архітектура програмного забезпечення, яку іноді називають багатошаровою архітектурою, або n-рівневою архітектурою, складається з різних рівнів, кожен з яких відповідає певній службі або інтеграції. Оскільки кожен шар окремий, вносити зміни до кожного шару легше, ніж братися за весь додаток в цілому.

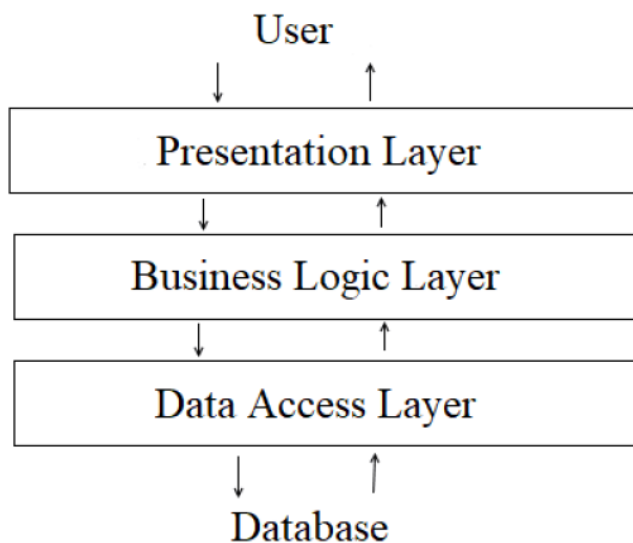


Рисунок 3.1 – схема багаторівневої архітектури

Порівняна сегрегація дозволяє відповідно керувати та підтримувати кожен рівень. Теоретично це повинно значно спростити спосіб управління програмною інфраструктурою. Багаторівневий підхід особливо підходить для розробки веб-додатків та хмарних додатків, що полегшує оновлення будь-яких застарілих систем – коли архітектура розбита на кілька шарів, зміни, які потрібно внести,

будуть простішими та менш масштабними, ніж вони могли б бути в іншому випадку. Багаторівнева архітектура також дозволяє, при потребі, розділити програмні компоненти на окремих серверах. Якщо розглянута система вимагає швидшого мережевого зв'язку, високої надійності та продуктивності, то n-tier має можливість забезпечити це, оскільки цей архітектурний шаблон призначений для зменшення накладних витрат, спричинених мережевим трафіком.

Архітектура додатку розділена на чотири логічні рівні згідно багаторівневої архітектури:

1. Domain Layer – рівень доступу до даних, де зберігаються моделі, описують використовувані сутності, а також містить класи для роботи з різними технологіями доступу до даних, тобто клас контексту даних Entity Framework Core, або ж провайдер Dapper. Тут також зберігаються репозиторії, через які рівень бізнес-логіки взаємодіє з базою даних.
2. Core Layer – бізнес-рівень, що містить набір компонентів, які відповідають за обробки отриманих від рівня вище даних, реалізує всю необхідну логіку додатків, все вирахування, взаємодіє з базою даних і передає рівню вище результати обробки. Бізнес логіка представлена у вигляді сервісів, які відповідають за якусь конкретну роботу.
3. Core Infrastructure Layer – рівень спільних модулів, який включає в себе інтерфейси(контракти) основних репозиторіїв та сервісів, спільні моделі, налаштування. Допомогає зробити код слабозв'язаним та з легкістю підключати інші компоненти або взаємозамінити їх. Містить інтерфейси, тобто контракти.
4. Web Api Layer – фактичний інтерфейс, через який клієнти можуть працювати з API, реалізовано через ASP.NET Core. Конфігурації маршрутів визначаються атрибутами. Цей рівень побудовано за архітектурним принципом REST.

Нижче наведена схема архітектури додатку з її основними компонентами:

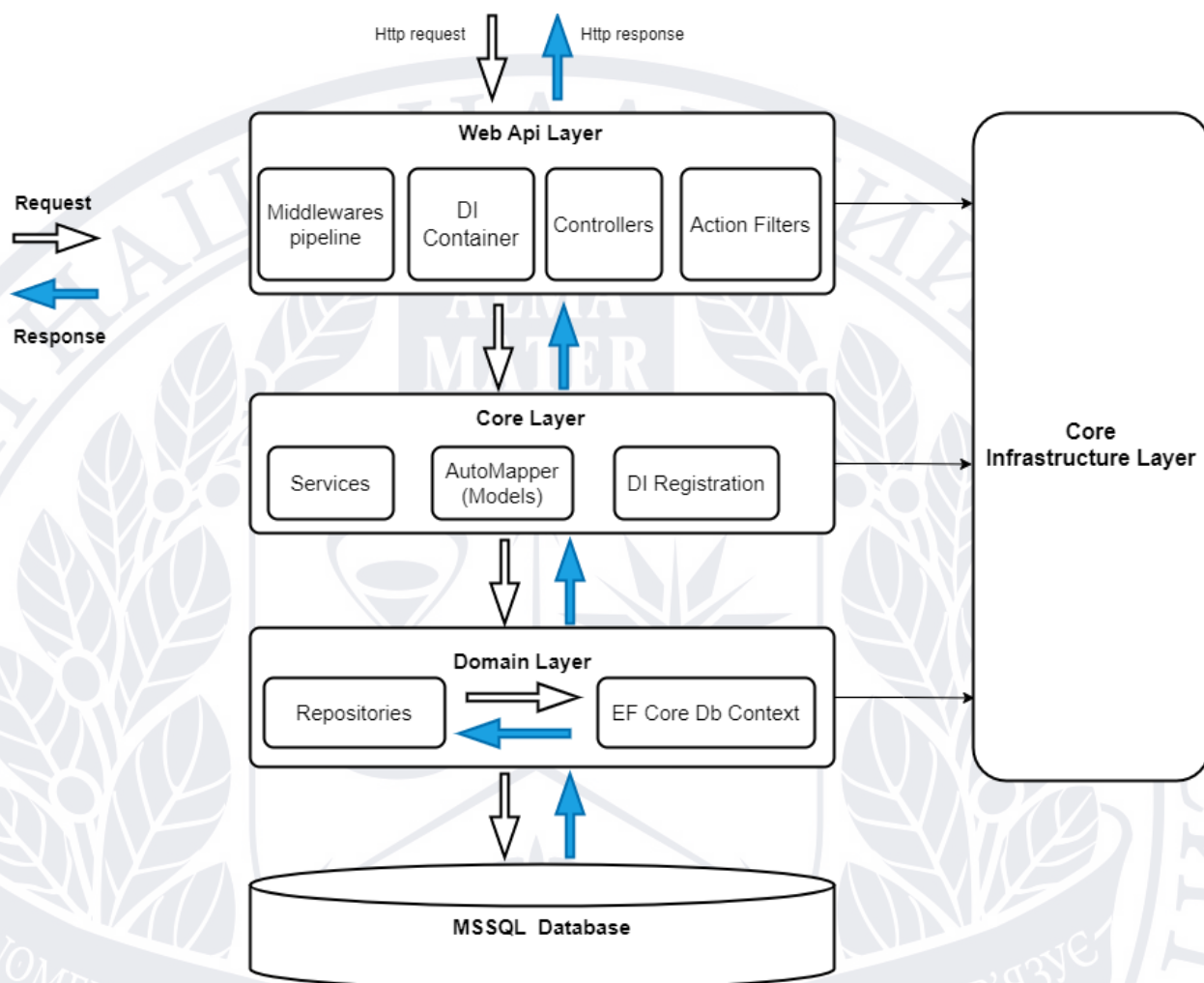


Рисунок 3.2 – архітектура додатку

3.2 Реалізація рівню доступу до даних

Для реалізації найнижчого рівня (Domain Layer) було вирішено використати патерни Repository та UnitOfWork. Репозиторій[26] — це ні що інше, як клас, визначений для сутності, з усіма можливими операціями для цієї

конкретної сутності. Наприклад, репозиторій для користувача сутності матиме основні операції CRUD(створення, читання, оновлення, видалення) та будь-які інші можливі операції, пов'язані з ним. Патерн дозволяє абстрагуватися від конкретної бази даних або компоненту взаємодії з базою(ORM), при використанні та при потребі дозволяє легко перейти з однієї системи на інші.

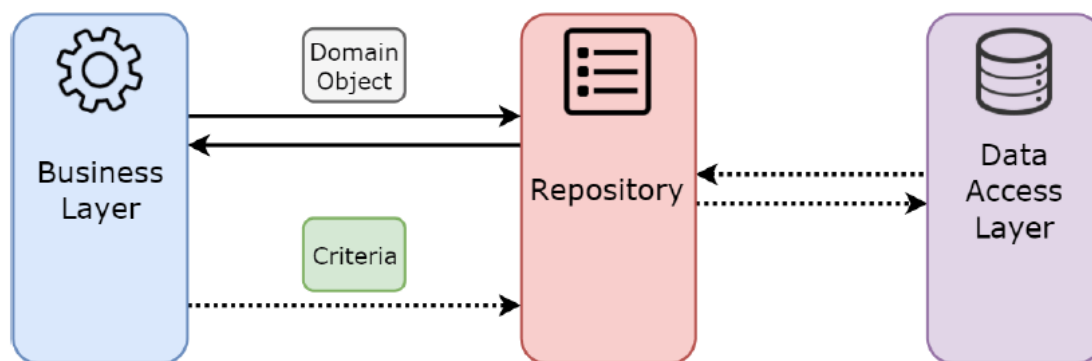


Рисунок 3.2 – механізм роботи патерну Репозиторій

Отже, репозиторій інкапсулює об'єкти, представлені в сховищах даних і операції, які до них застосовуються надаючи більш об'єктно-орієнтоване уявлення над даними.

Шаблон Unit of Work[27] дозволяє зібрати всі репозиторії в одному місці і дає впевненість, що всі репозиторії будуть використовуватися один і той же контекст даних. За його допомогою можна здійснювати доступ до репозиторіїв з єдиної точки. Основною перевагою такого підходу є використання основної властивості транзакцій баз даних – атомарності. Такий підхід дає гарантію, що при роботі з кількома сутностями(таблицями) при збереженні даних в БД потраплять або всі зміни одразу або ні одна. Також використання шаблону Repository та UoW дозволяє створити правильну структуру для розгортання програми та впровадження DI практик, які в тому числі допомагають у тестуванні проекту.

Для взаємодії з базою даних на концептуальному рівні використана технологія Entity Framework Core та провайдер БД Microsoft SQL Server. Основні сутності та схема бази даних спроектовані за допомогою підходу Code First[28], який дозволяє описати класи(сутності), встановити між ними відповідні та зв'язки та автоматично згенерувати таблиці бази даних, взаємодія з ними відбувається за допомогою об'єктно-орієнтованого підходу.

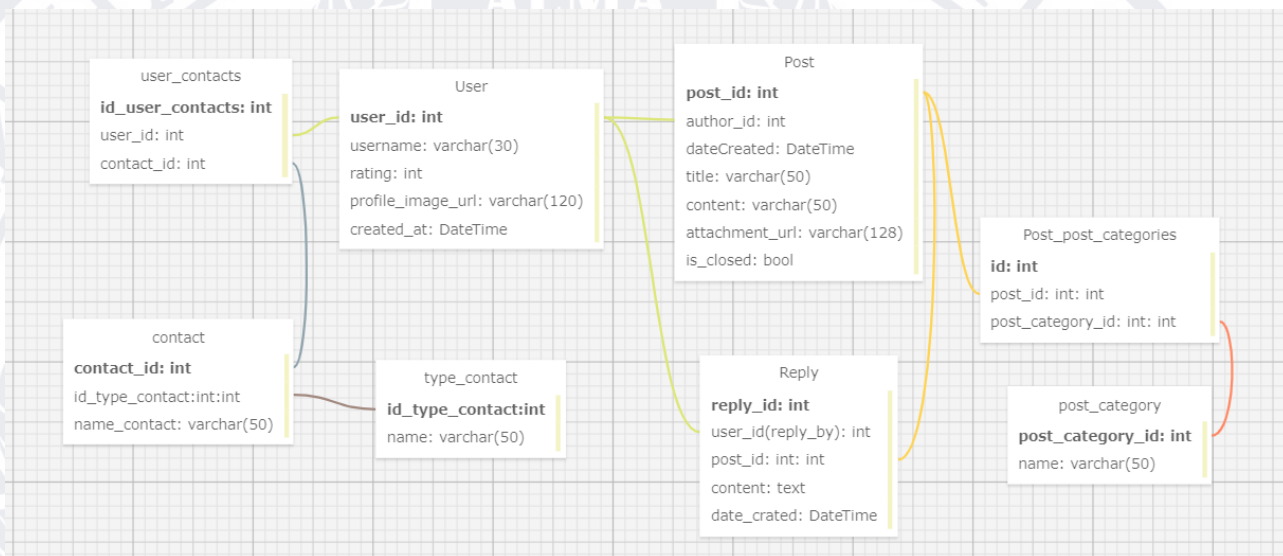


Рисунок 3.3 – схема бази даних додатку

Схема бази даних містить такі таблиці:

Таблиця User

User

user_id: int
 username: varchar(30)
 rating: int
 profile_image_url: varchar(120)
 created_at: DateTime

Рисунок 3.4 – таблиця User

Таблиця призначена для зберігання даних про користувачів, містить інформацію.

Таблиці contact, type_contact

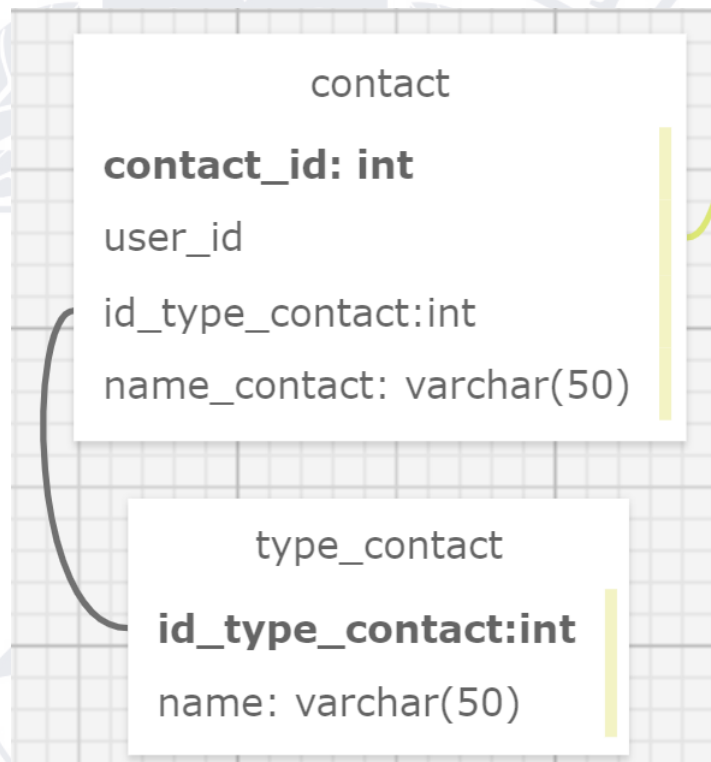
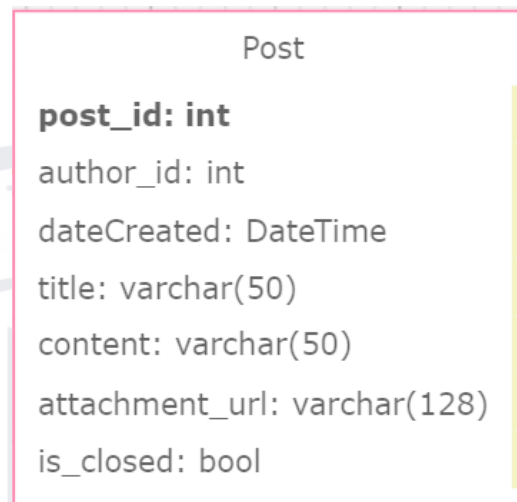


Рисунок 3.5 – таблиці contact, type_contact

Дані таблиці призначені для зберігання контактів користувача, є можливість додавати різні типи контактів, наприклад E-mail: usermail@mail.com, GitHub: [githublink.com](https://github.com), тощо. В одного користувача може бути багато контактів.

Таблиця Post



The diagram shows a box representing the 'Post' table. It contains a list of fields: 'post_id' (int, primary key), 'author_id' (int), 'dateCreated' (DateTime), 'title' (varchar(50)), 'content' (varchar(50)), 'attachment_url' (varchar(128)), and 'is_closed' (bool). A yellow vertical bar is on the right side of the box.

Post	
post_id:	int
author_id:	int
dateCreated:	DateTime
title:	varchar(50)
content:	varchar(50)
attachment_url:	varchar(128)
is_closed:	bool

Рисунок 3.6 – таблиця Post

Таблиця призначена для створення постів із питаннями, містить інформацію про автора, дату створення, заголовок, опис, поле яке вказує чи дане питання вирішено, прикріплення(у вигляді посилання на ресурс).

Таблиці post_category, post_post_categories

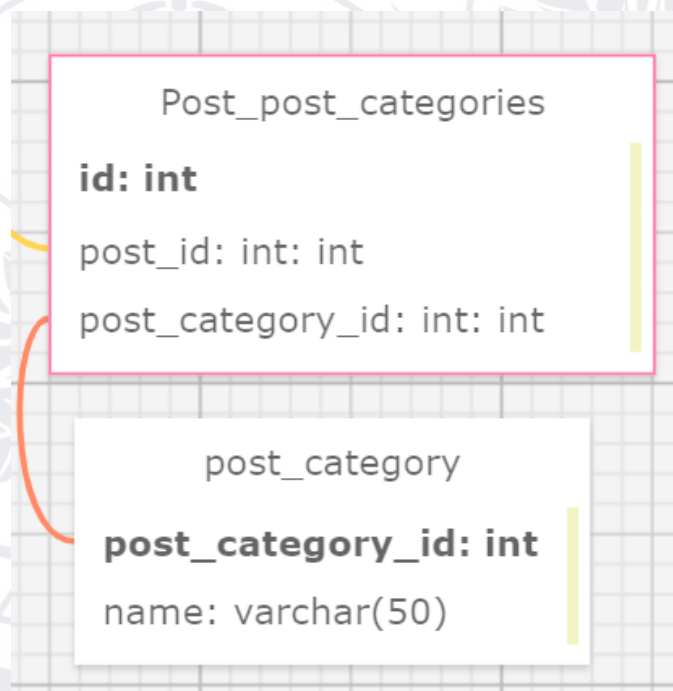


Рисунок 3.7 – таблиці post_category, post_post_categories

Дані таблиця містять дані про категорії до яких може належати пост. Один пост може мати багато категорій, одна категорія може відповідати багатьом постам. Для реалізації типу зв'язку «багато до багатьох» використовується допоміжна таблиця `post_post_categories`.

Таблиця Reply



Reply	
reply_id: int	
user_id(reply_by): int	
post_id: int: int	
content: text	
date_crated: DateTime	

Рисунок 3.8 – таблиця Reply

Таблиця призначена для можливості давати відповіді на пости із запитаннями, містить дані про користувача, що дає відповідь, пост, до якого надається відповідь, сам текст відповіді та дата створення.

3.3 Реалізація аутентифікації/авторизації користувачів

Для реалізації механізмів аутентифікації/авторизації використовується інструмент ASP.NET Core Identity та JWT-токени. Стандартна база даних була розширена за допомогою засобів ASP.NET Core Identity[29]:

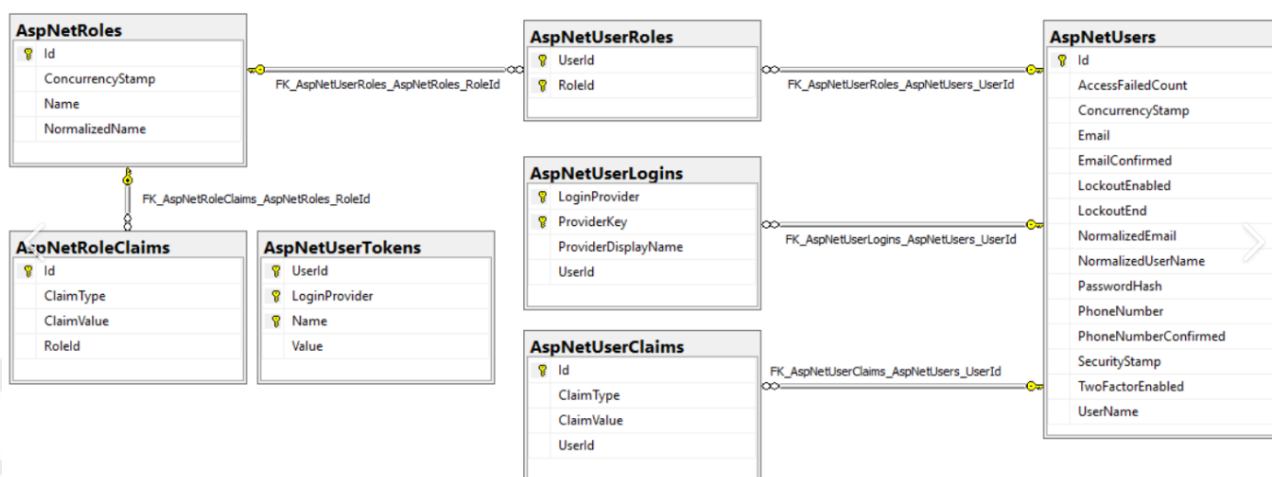


Рисунок 3.9 – схема бази даних для аутентифікації/авторизації користувачів

У файлі appsettings.json описані рядки підключення до баз даних та налаштування для створення та видачі сервером JWT-токенів.

```

{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=WebForumDb;Trusted_Connection=True;MultipleActiveResultSets=true",
    "WebForumProd": "Server=tcp:webforumdb.database.windows.net,1433;Initial Catalog=webforumdb;Persist Security Info=False;User ID=webforumdb"
  },
  "AllowedHosts": "*",
  "JWTSettings": {
    "securityKey": "WebForumSuperSecretKey12",
    "validIssuer": "WebForumApi",
    "validAudience": "WebForumApi",
    "expiryInMinutes": 180
  }
}

```

Рисунок 3.10 – налаштування в файлі appsettings.json

Такий підхід дає змогу тримати всі необхідні налаштування в одному місці та при потребі легко змінювати їх. Наприклад, секція «JWTSettings» містить рядки:

- securityKey – прихований ключ за допомогою якого сервер шифрує токени та при вхідних запитах перевіряє їх валідність
- validIssuer – вказує на того, хто видає токени
- validAudience – вказує на того хто приймає токени
- expiryInMinutes – задає час життя токenu у хвиликах

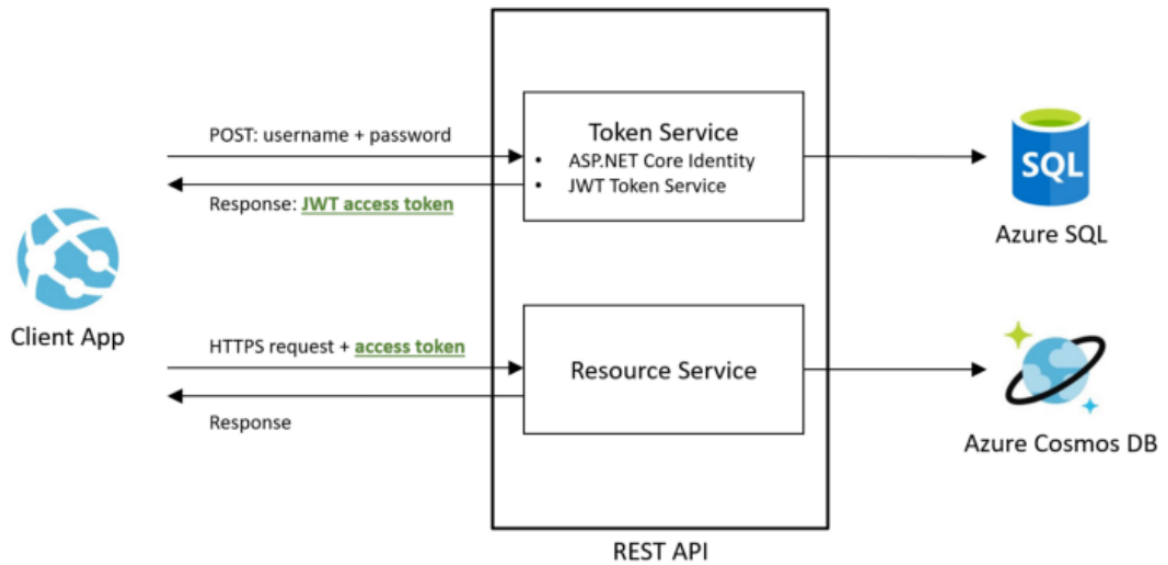


Рисунок 3.11 – процес авторизації/аутентифікації за допомогою JWT

Принцип роботи сервісу аутентифікації/авторизації[30]:

- Програма або клієнт запитує авторизацію до сервера авторизації. Це виконується за допомогою одного з різних потоків авторизації. Наприклад, типова веб-програма, сумісна з OpenID Connect, буде проходити через кінцеву точку /oauth/authorize за допомогою потоку коду авторизації.
- Коли авторизація проходить успішно, сервер авторизації повертає токен доступу до програми.

- Програма використовує маркер доступу для доступу до захищеного ресурсу (наприклад, API).

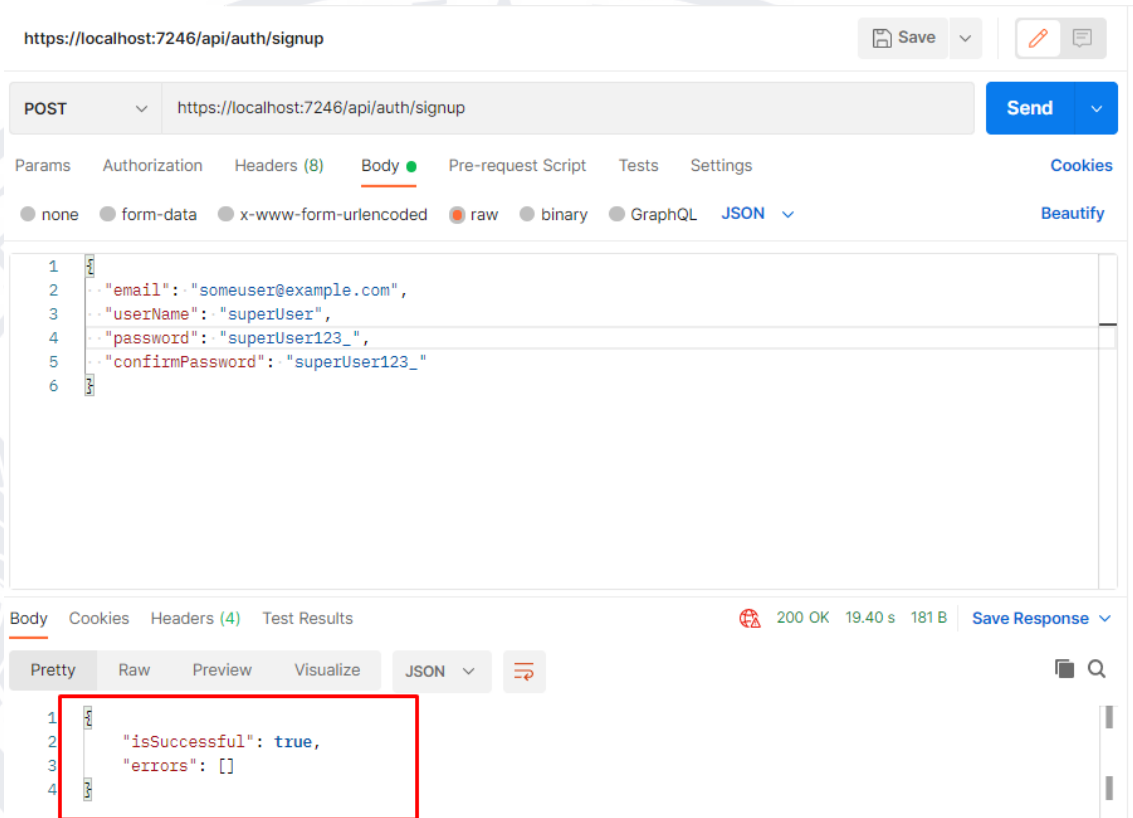


Рисунок 3.12 – приклад запиту на реєстрацію нового користувача

Для такого запиту створюється та відправляється POST-запит із клієнтського додатку на сервер, де в тілі запиту передаються дані користувача, які задля безпеки шифруються. При коректних даних сервер повертає статус 200(OK), який означає, що запит успішно виконано. В тілі відповіді можна помістити додаткову корисну інформації або дані. В БД дані зареєстрованого користувача виглядають так:

	Id	UserName	NormalizedU...	Email	NormalizedE...	EmailConfirm...	PasswordHash	SecurityStamp	ConcurrencyS...
	4a-08da3127193c	superUser	SUPERUSER	someuser@ex...	SOMEUSER@E...	False	AQAAAAEAACcQAAAAEB9PbqqTYMfpbftQuN...	3E72MYPO6GI...	1eec2a7a-b54...

Рисунок 3.13 – запис в БД

Слід зауважити, що такі важливі та уразливі дані як пароль користувача шифруються алгоритмом SHA-256[31], а не вносяться напряму.

логіку валідації та обробки даних, звернення до бази даних, формування відповідей та забезпечують користувача легким інтерфейсом для користування зручним функціоналом цих інструментів. Використання атрибуту `Authorize` дозволяє обмежити доступ до ресурсів на основі ролей. Це декларативний атрибут, який можна застосувати до контролера або методу дії. При вказанні цього атрибуту без аргументів він лише перевірить чи автентифікований користувач, для авторизації же слід вказувати додатковий атрибут `Role`.

3.4 Реалізація рівню доступу до API та демонстрація роботи додатку

API – програмний інтерфейс додатку, що представляє опис способів, якими одна програма може взаємодіяти з іншою, абстрагує базову реалізацію та надає тільки необхідні дані та методи розробнику. API відповідає на запитання "як можна звернутися до системи?" і включає:

- Операцію, яку ми можемо виконати;
- Дані, що надходять на вхід;
- Дані на виході (дані або повідомлення про помилку).

Для можливості доступу до API-додатку із клієнтських додатків(які можуть бути представлені у вигляді веб-сайтів, мобільних додатків, сервісів, тощо), використовуються так звані «ендпоінти» – кінцеві точки, які по суті представляють собою URL-адресу, що дає доступ до певних ресурсів системи, або дій. Наприклад – «`https://localhost:7246/api/posts`» у вигляді GET-запиту дає змогу отримати список всіх постів із серверу.

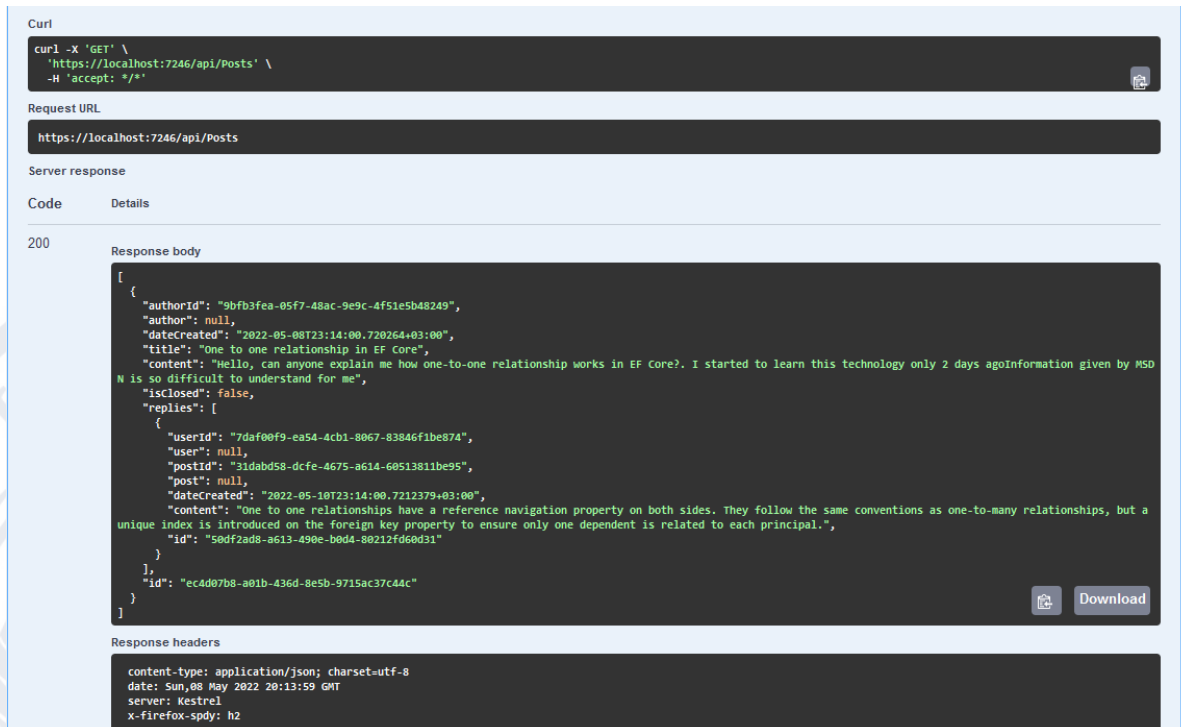


Рисунок 3.15 – приклад виконання запиту

Для коректної взаємодії клієнтської та серверної частин встановлюється формат узгодження даних, в даному випадку найпопулярніший – JSON. Статус коди встановлюються у відповідність HTTP-запитам.

```
3 public class Post : BaseEntity
4 {
5     1 reference
    public Guid? AuthorId { get; set; }
    0 references
    public Account Author { get; set; } = null!;
    1 reference
    public DateTime DateCreated { get; set; }
    1 reference
    public string Title { get; set; } = null!;
    1 reference
    public string Content { get; set; } = null!;
    1 reference
    public bool IsClosed { get; set; }
    3 references
    public ICollection<Reply> Replies { get; set; } = null!;
12 }
```

Рисунок 3.16 – таблиця Post в базі даних

Обробка запитів побудована по принципу конвеєру, який представлений у вигляді компонентів проміжного рівня – middleware. Компонент middleware приймає вхідний HTTP-запит, обробляє його та може передати далі наступному елементу в конвеєрі або ж виконати обробку та завершити роботу. Такий механізм можна застосовувати для обробки помилок, аутентифікації/авторизації, шифрування даних, переадресації тощо. Кінцевою точкою як показано на рисунку 3.17 є «ендпоінти» - кінцеві точки, звернення до маршрутів окремим HTTP-методами, що представлені URL-адресами. Виконують конкретну задачу, приймають параметри та повертають дані клієнту.

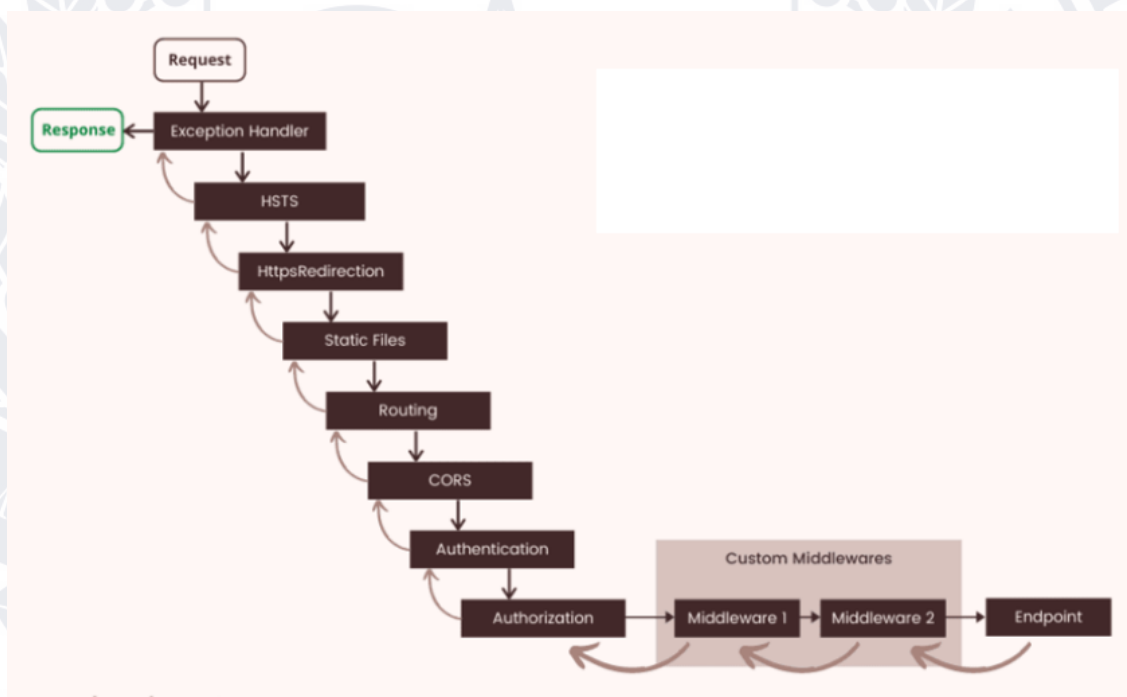
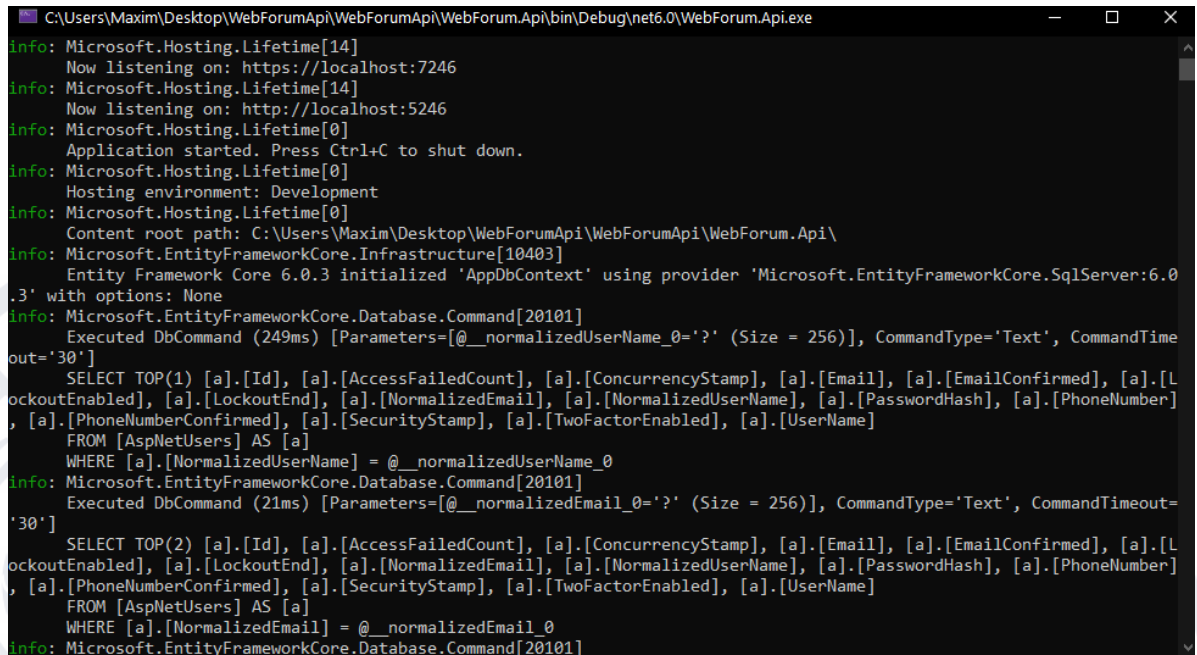


Рисунок 3.17 – конвеєр Middleware в додатку

Самий додаток запускається на базі веб-серверу Kestrel, який підтримує останні версії протоколу HTTP, HTTPS.



```

C:\Users\Maxim\Desktop\WebForumApi\WebForumApi\WebForum.Api\bin\Debug\net6.0\WebForum.Api.exe
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7246
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5246
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Maxim\Desktop\WebForumApi\WebForumApi\WebForum.Api\
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 6.0.3 initialized 'AppDbContext' using provider 'Microsoft.EntityFrameworkCore.SqlServer:6.0
      .3' with options: None
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (249ms) [Parameters=[@_normalizedUserName_0='?' (Size = 256)], CommandType='Text', CommandTimeo
      ut='30']
      SELECT TOP(1) [a].[Id], [a].[AccessFailedCount], [a].[ConcurrencyStamp], [a].[Email], [a].[EmailConfirmed], [a].[L
      ockoutEnabled], [a].[LockoutEnd], [a].[NormalizedEmail], [a].[NormalizedUserName], [a].[PasswordHash], [a].[PhoneNumber]
      , [a].[PhoneNumberConfirmed], [a].[SecurityStamp], [a].[TwoFactorEnabled], [a].[UserName]
      FROM [AspNetUsers] AS [a]
      WHERE [a].[NormalizedUserName] = @_normalizedUserName_0
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (21ms) [Parameters=[@_normalizedEmail_0='?' (Size = 256)], CommandType='Text', CommandTimeout=
      '30']
      SELECT TOP(2) [a].[Id], [a].[AccessFailedCount], [a].[ConcurrencyStamp], [a].[Email], [a].[EmailConfirmed], [a].[L
      ockoutEnabled], [a].[LockoutEnd], [a].[NormalizedEmail], [a].[NormalizedUserName], [a].[PasswordHash], [a].[PhoneNumber]
      , [a].[PhoneNumberConfirmed], [a].[SecurityStamp], [a].[TwoFactorEnabled], [a].[UserName]
      FROM [AspNetUsers] AS [a]
      WHERE [a].[NormalizedEmail] = @_normalizedEmail_0
info: Microsoft.EntityFrameworkCore.Database.Command[20101]

```

Рисунок 3.18 – запущений веб-сервер

Перевагою Kestrel є також підтримка асинхронності, процесу виконання запитів до зовнішніх ресурсів без блокування потоків, що дає змогу зменшити навантаження на сервер. Механізми логування дозволяють вести журнал подій, зберігати його та витягувати інформації при виникненні виключних ситуацій.

Для тестування кінцевих точок та надання зручного інтерфейсу взаємодії з API клієнтським додаткам використовується середовище Swagger, яке автоматично генерує та логічно розділяє всі описані маршрути веб-додатку, а також надає можливість формувати HTTP-запити та отримувати HTTP-відповіді у легкій формі. Особливо спрощує процес тестування вбудований механізм підтримки аутентифікації/авторизації на базі JWT-токенів.

Висновок до розділу 3

В даному розділі були описані архітектурні підходи для створення додатку, описаний сам процес створення та продемонстрована робота серверної частини додатку веб-форуму для ІТ-спеціалістів.

ВИСНОВКИ

В рамках даної кваліфікаційної роботи були виконані всі поставлені задачі. Проведено аналіз предметної області, який свідчить про збільшення попиту на технології, зростання користувачів глобальної мережі Інтернет. Огляд існуючих аналогів в даній сфері показав, що навіть найсучасніші та найбільші додатки не позбавлені недоліків. Активно формуються все більше веб-спільнот – місць, де люди можуть допомагати одне одному та ділитися інформацією, що свідчить про актуальність даної роботи.

Проведено аналіз, поглиблене вивчення та застосовано на практиці архітектурні підходи, програмні засоби та інструменти для створення серверних частин додатків.

В результаті було розроблено додаток веб-форуму для ІТ-спеціалістів, який відповідає всім поставленим вимогам. Весь функціонал працює відповідно до опису. Набуті навички проектування архітектури серверних додатків, що виражається у застосуванні багатошарової архітектури та принципів REST. Впроваджено та обґрунтовано найсучасніші процеси та концепції розробки програмного забезпечення, застосування баз даних та вивчення методології тестування додатків. Продемонстровано функції, які може виконувати додаток.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Number of Internet Users WordWide(2012-2022) [Електронний ресурс]. Режим доступу до ресурсу: <https://www.oberlo.com/statistics/how-many-people-use-internet>
2. Rheingold H. The virtual community. – Reading, Mass.: Addison-Wesley, 1993. 123 с.
3. The Reasons to use Internet [Електронний ресурс]. Режим доступу до ресурсу: <https://www.gwi.com/reports/social>
4. What is Reddit? [Електронний ресурс]. Режим доступу до ресурсу: <https://root-nation.com/ru/posts/internet-posts/ru-what-is-reddit/>
5. Alexa Internet – Reddit [Електронний ресурс]. Режим доступу до ресурсу: <https://www.alexa.com/siteinfo/reddit.com>
6. What is StackOverflow? [Електронний ресурс]. Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Stack_Overflow#cite_note-12
7. Alexa Internet – StackOverflow [Електронний ресурс]. Режим доступу до ресурсу: <https://www.alexa.com/siteinfo/stackoverflow.com>
8. Nabrahabr [Електронний ресурс]. Режим доступу до ресурсу: <https://ru.wikipedia.org/wiki/%D0%A5%D0%B0%D0%B1%D1%80>
9. What is .NET Platform? [Електронний ресурс]. Режим доступу до ресурсу: <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet>
10. Мова програмування C# [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/C_Sharp
11. Popularity of Programming Language [Електронний ресурс]. Режим доступу до ресурсу: <https://pypl.github.io/PYPL.html>
12. Технологія ASP.NET Core [Електронний ресурс]. Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/ASP.NET>

13. Введение в ASP.NET Core. Что такое ASP.NET Core? [Электронный ресурс]. Режим доступа до ресурсу: <https://metanit.com/sharp/aspnet6/1.1.php>
14. IDE Microsoft Visual Studio [Электронный ресурс]. Режим доступа до ресурсу: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio
15. First look at the Visual Studio Debugger [Электронный ресурс]. Режим доступа до ресурсу: <https://docs.microsoft.com/en-us/visualstudio/debugger/debugger-feature-tour?view=vs-2022>
16. Connect to projects in Team Explorer [Электронный ресурс]. Режим доступа до ресурсу: <https://docs.microsoft.com/en-us/visualstudio/ide/connect-team-project?view=vs-2019>
17. Microsoft SQL Server [Электронный ресурс]. Режим доступа до ресурсу: https://ru.wikipedia.org/wiki/Microsoft_SQL_Server
18. What is SQL Server? [Электронный ресурс]. Режим доступа до ресурсу: <https://www.techtarget.com/searchdatamanagement/definition/SQL-Server>
19. Введение в Entity Framework Core. Что такое Entity Framework Core? [Электронный ресурс]. Режим доступа до ресурсу: <https://metanit.com/sharp/entityframeworkcore/1.1.php>
20. Swagger(OpenApi 3.0) [Электронный ресурс]. Режим доступа до ресурсу: <https://habr.com/ru/post/541592/>
21. What Is the Difference Between Authentication and Authorization? [Электронный ресурс]. Режим доступа до ресурсу: <https://www.sailpoint.com/identity-library/difference-between-authentication-and-authorization/#:~:text=Simply%20put%2C%20authentication%20is%20the,people%20can%20come%20on%20board.>
22. Introducing to ASP.NET Identity [Электронный ресурс]. Режим доступа до ресурсу: <https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity>

23. Introducing to JWT tokens [Електронний ресурс]. Режим доступу до ресурсу: <https://jwt.io/introduction>
24. Internet Engineering Task Force [Електронний ресурс]. Режим доступу до ресурсу: <https://tools.ietf.org/html/rfc7519>
25. What is multalayered architecture in software [Електронний ресурс]. Режим доступу до ресурсу: <https://hub.packtpub.com/what-is-multi-layered-software-architecture/>
26. Паттерн «Репозиторий» в ASP.NET [Електронний ресурс]. Режим доступу до ресурсу: <https://metanit.com/sharp/articles/mvc/11.php>
27. Unit Of Work [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unit_Of_Work
28. Approach Code-First [Електронний ресурс]. Режим доступу до ресурсу: <https://entityframeworkcore.com/approach-code-first>
29. ASP.NET Core Identity DB Scheme [Електронний ресурс]. Режим доступу до ресурсу: <https://deblokt.com/2020/01/24/04-part-2-identityserver4-asp-net-core-identity-net-core-3-1/>
30. JWT Introduction [Електронний ресурс]. Режим доступу до ресурсу: <https://jwt.io/introduction>
31. SHA 256 Class [Електронний ресурс]. Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.sha256?view=net-6.0>

Декларація щодо унікальності текстів роботи
та невикористання матеріалів інших авторів без посилань

Сніжинський Максим Вадимович

Прізвище, ім'я, по батькові

Факультет інформаційних і прикладних технологій

Факультет

122 Комп'ютерні науки

Шифр і назва спеціальності

Сучасні інформаційні технології та програмування

Освітня програма

ДЕКЛАРАЦІЯ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (бакалаврська) робота на тему: «Розробка серверної частини додатку веб-форуму для ІТ-спеціалістів» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувались іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомляю, що у разі порушення цього порядку моя кваліфікаційна (бакалаврська) робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

дата

підпис

