Міністерство освіти і науки України
Одеський національний технологічний університет
Кафедра комп'ютерної інженерії

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему *Дослідження процесу клієнт-серверної взаємодії при виконанні математичних обчислень*
(назва кваліфікаційної роботи згідно наказу ОНТУ)

Здобувача *Кондратов О.С.*
(прізвище, ініціали)

6 курсу *561 а* групи

Керівники: *к.т.н., доц. Сахарова С.В.*
(посада, прізвище та ініціали)
ст. викл. Жирнова Т.М.
(посада, прізвище та ініціали)

Консультанти: _____
(посада, прізвище та ініціали)

_____
посада, прізвище та ініціали)

Кваліфікаційна робота допускається до захисту

Рішення кафедри від _____ 2023 р., протокол № _____

Завідувач кафедри комп. інженерії _____ Сергій АРТЕМЕНКО
(назва кафедри) (підпис) (Ім'я ПРІЗВИЩЕ)

Одеса – 2023 рік

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерної інженерії, програмування та кіберзахисту*
Кафедра *комп'ютерної інженерії*
Ступінь вищої освіти *магістр*
Спеціальність *123 «Комп'ютерна інженерія»*
Освітня програма *Комп'ютерні системи та мережі*

ЗАТВЕРДЖУЮ
Зав. кафедри *комп'ютерної інженерії*
Сергій АРТЕМЕНКО
« *30* » 11 2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Кондратова Олександра Сергійовича

1. Тема роботи *Дослідження процесу клієнт-серверної взаємодії при виконанні математичних обчислень*

Затверджена наказом університету від « *30* » *11* 2022 р., наказ № *884-03*

2 Термін здачі здобувачем закінченої роботи *25 листопада 2023 р.*

3. Вихідні дані роботи

1. Методи математичних обчислень

2. Технології розробки клієнт-серверних додатків.

4. Перелік питань, які потрібно розробити

1. Вступ. 2. Аналіз принципів створення клієнт-серверних систем. 3. Розробка схеми процесу клієнт-серверної взаємодії при виконанні математичних операцій.

4. Розробка інтерфейсу користувача системи виконання математичних операцій

Реалізація клієнт-серверної системи для виконання математичних операцій 5. Економічні розрахунки. 6. Охорона праці. 7. Загальні висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Інтерактивна презентація (мета, об'єкт, предмет дослідження, постановка задачі;

Схема взаємодії модулів програми, Варіанти реалізації додатку за допомогою Xamarin, Інтерфейс програми, Інтерфейс мобільного додатку, Набір юніт-тестів, що використані

Фрагменті програмного коду, Висновки.)

6. Консультанти по роботі, із зазначенням розділів роботи, що стосуються їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Економіка			
Охорона праці			
Нормоконтроль			

7. Дата видачі завдання *10.10.2023*

Керівники _____

_____

Завдання прийняв до виконання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Дослідженні об'єкту.*	*05.09.2023*	
2.	*Дослідження технології.*	*05.10.2023*	
3.	*Постановка завдання. Визначення вихідних даних.*	*07.11.2023*	
4.	*Розробка схеми процесу клієнт-серверної взаємодії при виконанні математичних операцій.*	*07.11.2023*	
5.	*Реалізація клієнт-серверної системи для виконання математичних операцій.*	*27.11.2023*	
6.	*Підготовка техніко-економічної частини*	*30.11.2023*	
7.	*Підготовка розділу охорони праці*	*30.11.2023*	
8.	*Оформлення пояснювальної записки*	*30.11.2023*	
9.	*Оформлення графічної частини та лістингу*	*30.11.2023*	
10.			

Здобувач-дипломник _____ *Олександр КОНДРАТОВ*

Керівник роботи _____ *Світлана САХАРОВА*

Несу відповідальність за ідентичність електронного та друкованого варіантів кваліфікаційної роботи, даю згоду на обробку персональних даних та не заперечую проти розміщення кваліфікаційної роботи на офіційних web-ресурсах ОНТУ.

Підтверджую, що в кваліфікаційній роботі відсутні порушення норм академічної доброчесності.

Здобувач-дипломник *Олександр КОНДРАТОВ* _____

АНОТАЦІЯ

Дипломна робота присвячена дослідження процесу клієнт-серверної взаємодії при виконанні математичних обчислень.

Основні завдання, вирішені в роботі: аналіз клієнт-серверної архітектури, аналіз особливостей взаємодії клієнта та серверу; розробка схеми процесу клієнт-серверної взаємодії при виконанні математичних операцій; розробка інтерфейсу користувача системи виконання математичних операцій; описання програмної частини для розробника; реалізація клієнт-серверної системи для виконання математичних операцій.

У розділі Реалізація клієнт-серверної системи наведено Опис роботи програмного забезпечення; Опис інтерфейсу користувача; Опис системи для розробника; Технології реалізації автоматизованої системи.

Для реалізації програмного забезпечення та вирішення задачі даного дипломного проекту була обрана клієнт-серверна технологія ASP.NET WEB API та мова програмування C#. Середою розробки для обраної технології є Microsoft Visual Studio, що має безліч інструментів для створення програмного забезпечення. Інтегроване середовище розробки Visual Studio використовується для редагування, налагодження та складання коду, а також для публікації програм.

Також виконано економічний розрахунок та розглянуто питання охорони праці.

Ключові слова: система клієнт-сервер

.

ABSTRACT

The thesis is devoted to the study of the process of client-server interaction when performing mathematical calculations.

The main tasks solved in the work: analysis of the client-server architecture, analysis of the features of the interaction between the client and the server; development of a scheme of the process of client-server interaction when performing mathematical operations; development of the user interface of the system for performing mathematical operations; description of the software part for the developer; implementation of a client-server system for performing mathematical operations.

The section Implementation of the client-server system provides a description of the operation of the software; Description of the user interface; Description of the system for the developer; Technologies for implementing an automated system.

ASP.NET WEB API client-server technology and C# programming language were chosen to implement the software and solve the problem of this diploma project. The development environment for the chosen technology is Microsoft Visual Studio, which has many tools for creating software. The Visual Studio integrated development environment is used to edit, debug, and compile code, as well as to publish applications.

An economic calculation was also performed and the issue of labor protection was considered.

Keywords: client-server system

.

ЗМІСТ

					КРМ.КІ.0.884-03.2.16		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розробив		Олександр КОНДРАТОВ			Дослідження процесу клієнт-серверної взаємодії при виконанні математичних обчислень	Літ.	Арк. / Акрушів
Перевірив		Світлана Сахарова					
Рецензент						гр. 561, ОНТУ	
Нормоконтроль							
Затвердив		Сергій АРТЕМЕНКО					

ВСТУП

Область застосування інформаційних систем постійно розширюється,а самі вони стають дедалі складнішими. Деякі системи виростають і ускладнюються настільки, що набувають глобального характеру,і від їхнього правильного та надійного функціонування починає залежати діяльність десятків або навіть сотень тисяч людей. Через свою"глобальності" (потрібно забезпечити доступ до системи з територіально рознесених між собою точок), а також через ряд інших причин такі системи часто мають дуже складну архітектуру, що передбачає їх функціонування у вигляді набору компонентів, кожен з яких виконується на окремому вузлі. Оскільки кількість таких систем постійно зростає, вимоги до них досить серйозні.

Бурхливе зростання індустрії персональних комп'ютерів, що почалося, спочатку мало що змінило в ідеології побудови програмних систем.

Як і раніше, здебільшого програми мали справу з локальними ресурсами. Щоправда, частина цих ресурсів була вже "псевдолокальною", наприклад, файли на мережному диску. Проте файл оброблявся безпосередньо самим вузлом, при цьому файл спочатку передавався по мережі (вже на цьому етапі розвитку виникли складності - проблеми блокування ресурсів та попередження глухих кутів, проблеми підтримки логічної цілісності для внесених змін тощо).

У якийсь момент стало очевидно, що традиційні підходи не працюють.

При збільшенні обсягу даних, що переробляються, а також у міру зростання їх вартості стало очевидно, що довіряти їх обробку клієнтським машинам не можна. Будь-яка помилка на них (а чим більше клієнтів, тим більше ймовірність помилки) призводить або до втрати даних, або до їх блокувань у процесі роботи, а отже, до зниження загальної продуктивності системи.

Наступним ключовим кроком стало поширення ідеології клієнт-серверної обробки. Це були "дворолеві" системи: клієнт ніс відповідальність за

відображення інтерфейсу користувача і виконання коду програми, а роль сервера зазвичай доручалася СУБД.

Застосування наприклад з файлом перехід до клієнт-серверної архітектури може бути проілюстрований наступним чином: замість того, щоб читати файл повністю і обробляти його, машина-клієнт передає машині-сервер запит, в якому вказує, яким чином файл повинен бути оброблений. Сервер запит клієнта обробляє та повертає йому результат.

Повсюдний перехід на технологію клієнт-сервер допоміг вирішити багато старих проблем, але при цьому створив багато нових. Однією з основних труднощів було й залишається визначення межі між функціоналом клієнта та сервера. Часто рішення про перенесення частини завдань на сервер згубно впливає на загальну продуктивність системи, і навпаки, перенесення частини навантаження на клієнта може призвести до втрати централізації.

У міру зростання популярності систем "клієнт-сервер" набирала сили і технологія об'єктно-орієнтованого програмування, яка пропонувала перейти до системної архітектури з трьома шарами: шар уявлення відводиться інтерфейсу користувача, шар предметної області призначений для опису основних функцій програми, необхідні досягнення поставленої ним мети, а третій шар представляє джерело даних.

З появою Web всім раптово захотілося мати системи "клієнт-сервер", де в ролі клієнта виступав би Web-браузер. З'явилися інструментальні засоби конструювання Web-сторінок були меншою мірою пов'язані з SQL і тому більш підходили для реалізації третього рівня.

В даний час можна вважати, що бум технологій, пов'язаних з клієнт-серверною архітектурою, все ще продовжується - більшість інформаційних систем, що працюють в даний час, виконано в цій технології. Однак актуальними є напрями, пов'язані з розвитком цієї ідеї - так звані тришарові та багатошарові, а також децентралізовані програми.

Однією з найважливіших переваг клієнт-серверних інформаційних систем у порівнянні з їхніми аналогами, створеними на основі мережевих версій

настільних СУБД є зниження мережного трафіку під час виконання запитів. Наприклад, при необхідності вибору п'яти записів з таблиці, що містить мільйон, клієнтський додаток посилає серверу запит, який сервером компілюється, оптимізується і виконується, після чого результат запиту (ті п'ять записів, а зовсім не вся таблиця) передається назад на робочу станцію (якщо, звичайно, клієнтська програма коректно формулює запити до сервера). При цьому нерідко в першому наближенні можна не замислюватися, а чи є в загальному індекс, здатний полегшити пошук потрібних записів, - якщо він є він буде використаний сервером, якщо ні - запит все одно буде виконаний, хоча, швидше за все, за більшу кількість часу.

Другою перевагою архітектури "клієнт/сервер" є можливість зберігання бізнес-правил на сервері, що дозволяє уникнути дублювання коду в різних додатках, що використовують загальну базу даних, вироблено лише у межах цих правил.

Крім того, для опису серверних бізнес-правил, у найбільш типових ситуаціях (як у прикладі із замовниками та замовленнями) існують дуже зручні інструменти - так звані CASE-засоби (CASE означає Computer-Aided System Engineering), що дозволяють описати подібні правила та створювати реалізуючі їх об'єкти бази даних (індекси, тригери), буквально малюючи мишею зв'язку між таблицями, без будь-якого програмування. У цьому випадку клієнтська програма буде позбавлена значної частини коду, пов'язаного з реалізацією бізнес-правил безпосередньо в додатку. Зазначимо також, що частина коду, пов'язаного з обробкою даних, також може бути реалізована у вигляді збережених процедур сервера, що дозволяє ще більш "полегшити" клієнтський додаток, це означає, що вимоги до робочих станцій можуть бути не настільки високі. Це зрештою здешевлює вартість інформаційної системи навіть за використання дорогої серверної СУБД і потужного серверу баз даних.

Крім перерахованих можливостей сучасні серверні СУБД мають численні засоби управління користувальницькими привілеями і правами доступу до різних об'єктів бази даних. Як правило, у базі даних зберігаються відомості про

її користувачів, їх паролі та привілеї, а кожен об'єкт бази даних, такий, як, наприклад, таблиця, належить якомусь користувачеві. Власник об'єкта може надати іншим користувачам право в той чи інший спосіб використовувати об'єкт (наприклад, дозволити читати з нього дані будь-якому іншому користувачеві).

Об'єктом дослідження є клієнт-серверні системи.

Предметом – методи розробки клієнт-серверних систем.

Метою роботи є розробка клієнт-серверної системи для виконання математичних операцій.

До основних **завдань**, вирішення яких веде до досягнення поставленої мети, належать:

1. Аналіз клієнт-серверної архітектури, аналіз особливостей взаємодії клієнта та серверу.

2. Розробка схеми процесу клієнт-серверної взаємодії при виконанні математичних операцій.

3.Розробка інтерфейсу користувача системи виконання математичних операцій.

4. Описання програмної частини для розробника.

5. Реалізація клієнт-серверної системи для виконання математичних операцій.

РОЗДІЛ 1
АНАЛІЗ ПРИНЦИПІВ СТВОРЕННЯ КЛІЄНТ-СЕРВЕРНИХ СИСЕМ

1.1 Визначення поняття Клієнт — сервер та особливості їх взаємодії

«Клієнт — сервер» (англ. client–server) — обчислювальна чи мережева архітектура, у якій завдання чи мережне навантаження розподілені між постачальниками інфокомунікаційних послуг, званими серверами, та замовниками послуг, званими клієнтами (користувачами). Фактично клієнт та сервер – це програмне забезпечення. Зазвичай програми розташовані на різних обчислювальних машинах і взаємодіють між собою через обчислювальну мережу за допомогою мережевих протоколів, але можуть бути розташовані також і на одній машині. Програми-сервери очікують від клієнтських програм запити та надають їм свої ресурси у вигляді даних (наприклад, передача файлів за допомогою HTTP, FTP, BitTorrent, потокове мультимедіа або робота з базами даних) або у вигляді сервісних функцій (наприклад, робота з електронною поштою, спілкування за допомогою систем миттєвого обміну повідомленнями або перегляду веб-сторінок у всесвітній павутині). Оскільки одна програма-сервер може виконувати запити від багатьох програм-клієнтів, її розміщують на спеціально виділеній обчислювальній машині, налаштованій особливим чином, як правило, спільно з іншими програмами-серверами, тому продуктивність цієї машини повинна бути високою. Через особливу роль такої машини в мережі, специфіки її обладнання та програмного забезпечення її також називають сервером, а машини, що виконують клієнтські програми, відповідно, клієнтами.

Характеристика клієнт-сервер визначає відносини взаємодіючих програм у додатку. Серверний компонент надає функцію чи послугу одному або декільком клієнтам, які ініціюють запити на такі послуги. Сервери класифікуються за наданими ними послугами. Наприклад, веб-сервер обслуговує веб-сторінки, а файловий сервер обслуговує комп'ютерні файли.

Загальний ресурс може бути будь-яким із програмного забезпечення та електронних компонентів комп'ютера - сервера, від програм і даних у процесорах і пристроїв, що запам'ятовують. Спільне використання ресурсів сервера є послугою.

Чи є комп'ютер клієнтом, сервером або й тим, і іншим, визначається характером програми, якому потрібні сервісні функції. Наприклад, на одному комп'ютері можуть одночасно працювати веб-сервери та програмне забезпечення файлового сервера, щоб обслуговувати різні дані для клієнтів, які надсилають різні типи запитів. Клієнтське програмне забезпечення також може взаємодіяти з серверним програмним забезпеченням на тому самому комп'ютері. Зв'язок між серверами, наприклад, для синхронізації даних, іноді називається міжсерверним.

Взаємодія клієнта та сервера

Взагалі кажучи, служба - це абстракція комп'ютерних ресурсів, і клієнту не потрібно турбуватися про те, як сервер працює під час виконання запиту та доставки відповіді. Клієнту потрібно тільки зрозуміти відповідь, засновану на відомому протоколі додатка, тобто зміст та форматування даних для запитуваної послуги.

Клієнти та сервери обмінюються повідомленнями у шаблоні запит-відповідь. Клієнт надсилає запит, а сервер повертає відповідь. Цей обмін повідомленнями є прикладом взаємодії між процесами. Для взаємодії комп'ютери повинні мати спільну мову, і вони повинні виконувати правила, щоб і клієнт, і сервер знали, чого очікувати. Мова та правила спілкування визначені у протоколі зв'язку. Усі протоколи клієнт-серверної моделі працюють лише на рівні додатків. Протокол прикладного рівня визначає основні шаблони діалогу. Щоб ще більше формалізувати обмін даними сервер може реалізувати інтерфейс прикладного програмування (API). API — це рівень абстракції доступу до сервісу. Обмежуючи зв'язок певним форматом контенту, він полегшує синтаксичний аналіз. Абстрагуючи доступ, він полегшує міжплатформний обмін даними.

Сервер може отримувати запити від багатьох клієнтів за короткий період часу. Комп'ютер може виконувати лише обмежену кількість завдань у будь-який момент і покладається на систему планування для визначення пріоритетів запитів, що входять від клієнтів для їх задоволення. Щоб запобігти зловживанням та максимізувати доступність серверне програмне забезпечення може обмежувати доступність для клієнтів. Атаки типу "відмова в обслуговуванні" використовують обов'язки сервера обробляти запити, такі атаки діють шляхом навантаження сервера надмірною частотою запитів. Шифрування слід застосовувати, якщо між клієнтом та сервером має передаватися конфіденційна інформація.

Переваги

Відсутність дублювання коду програми-сервера програмами-клієнтами.

Оскільки всі обчислення виконуються на сервері, то вимоги до комп'ютерів, де встановлено клієнт, знижуються.

Всі дані зберігаються на сервері, який, як правило, захищений набагато краще за більшість клієнтів. На сервері простіше організувати контроль повноважень, щоб дозволяти доступ до даних лише клієнтам із відповідними правами доступу.

Недоліки

Непрацездатність сервера може зробити непрацездатною всю обчислювальну мережу. Непрацездатним сервером слід вважати сервер, продуктивності якого не вистачає на обслуговування всіх клієнтів, а також сервер, що знаходиться на ремонті, профілактиці тощо.

Підтримка роботи даної системи потребує окремого спеціаліста – системного адміністратора.

Висока вартість обладнання.

1.2 Поняття клієнт-серверної архітектури

З клієнт-серверною архітектурою в процесі доступу до інфокомунікаційних послуг користувачі стикаються, коли, наприклад купують квиток в кіно онлайн, бронюють путівку на море або записуються до лікаря.

На клієнт-серверній архітектурі збудовано всі сайти та Інтернет-сервіси. Також її використовують десктоп-програми, які передають дані через пережі передачі даних Інтернет. Тому ІТ-фахівцеві потрібно розуміти, що це таке і як працює.

Архітектура клієнт-сервер визначає загальні принципи організації взаємодії в мережі, де є сервери, вузли-постачальники деяких специфічних функцій (сервісів) і клієнти (споживачі цих функцій).

Практичні реалізації такої архітектури називаються клієнт-серверними технологіями.

Дволанкова архітектура - розподіл трьох базових компонентів між двома вузлами (клієнтом та сервером). Дволанкова архітектура використовується в клієнт-серверних системах, де сервер відповідає на запити клієнтів безпосередньо і в повному обсязі.

Застосунками та сайтами одночасно можуть користуватися сотні і навіть мільйони людей. Всі вони звертаються до одного комп'ютера, який має вміти обробляти запити та надсилати відповіді. Такий підхід називається клієнт-серверною архітектурою. Вона описує, як відбувається робота з користувачами, де зберігаються дані та як забезпечується їх захист.

У клієнт-серверній архітектурі використовуються три компоненти:

Клієнт – програма, яку ми використовуємо в Інтернеті. Найчастіше це браузер, але може бути інша окрема програма

Сервер — це комп'ютер, на якому зберігається сайт або програма. Коли ми заходимо на сайт магазину, звертаємося до сервера, на якому знаходиться сайт.

База даних — це програма, в якій зберігаються всі дані програми. Для магазину це буде база клієнтів, товарів та замовлень

Архітектура клієнт-сервер визначає принципи спілкування між комп'ютерами, а правила і взаємодії визначені в протоколі.

Мережевий протокол – це набір правил, за якими відбувається взаємодія між комп'ютерами в мережі.

Мережеві протоколи:

TCP/IP – набір (стек) протоколів передачі даних. TCP/IP – це позначення всієї мережі, яка працює на основі двох протоколів – TCP і IP.

TCP (Transfer Control Protocol) – протокол, який служить для встановлення надійного з'єднання між двома пристроями, передачі інформації і підтвердження її отримання.

IP (Internet Protocol) – інтернет протокол, який відповідає за правильність доставки повідомлень за певною адресою. При цьому дані розбиваються на пакети, які можуть доставлятися по-різному.

MAC (Media Access Control) – протокол, за допомогою якого відбувається ідентифікація мережевих пристроїв. Всі пристрої, підключені до інтернету, мають свою унікальну MAC адресу.

ICMP (Internet control message protocol) – протокол, який відповідає за обмін інформацією, але не використовується для передачі даних.

UDP (User datagram protocol) – протокол, який керує передачею інформації, але інформація не проходить перевірку при отриманні. Даний протокол працює швидше, ніж TCP.

HTTP (Hyper Text Transfer Protocol) – протокол передачі гіпертексту, на основі якого працюють всі сайти. Він запитує необхідні дані у віддаленій системи (веб-сторінки, файли).

FTP (File Transfer Protocol) – протокол передачі файлів зі спеціального файлового сервера на комп'ютер користувача.

SSH (Secure Shell) – протокол, який служить для забезпечення віддаленого керування системою по захищеному каналу.

POP3 (Post Office Protocol) – стандартний протокол поштового з'єднання, який відповідає за доставку пошти.

SMTP (Simple Mail Transfer Protocol) – протокол, який визначає правила для передачі пошти. Відповідає за повернення або підтвердження про доставку, оповіщення про помилку.

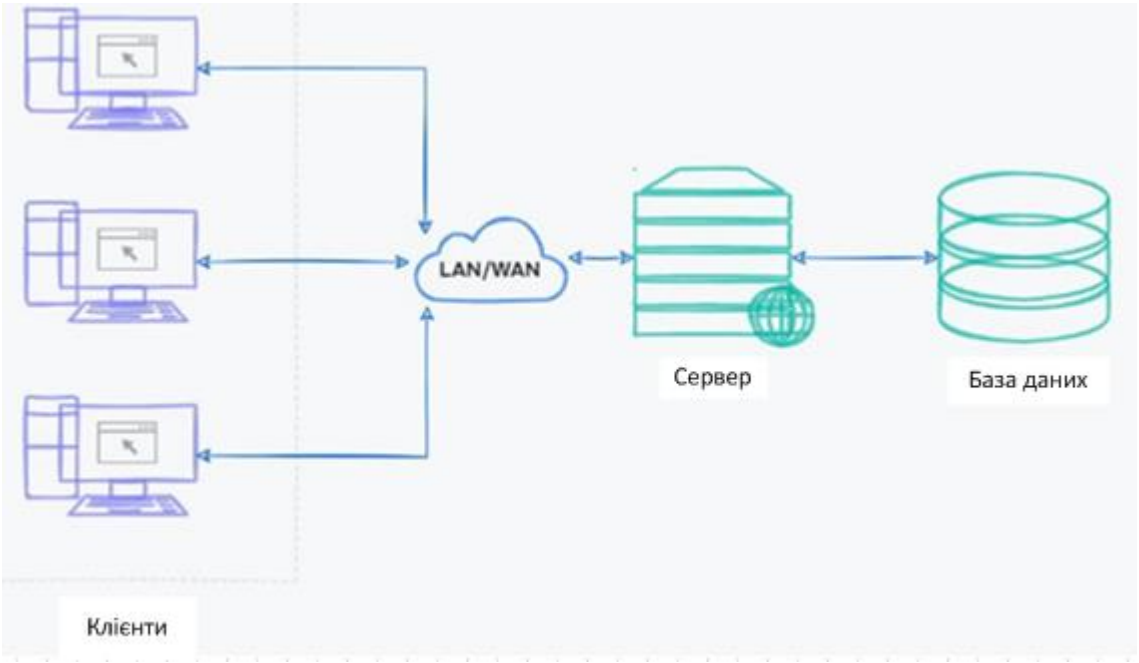


Рис. 1.1 – Система клієнт-сервер

Клієнт – це завжди програма. Її призначення – дати користувачеві зручний спосіб взаємодії із сервером.

З точки зору сервера, вибрати товар - це надіслати запит спеціальною мовою запитів, наприклад SQL. Але для простого користувача це складно. Тому клієнт пропонує зручний спосіб взаємодії, щоб не писати код своїми руками.

Існують концепції побудови системи клієнт-сервер:

Слабкий клієнт – потужний сервер. У такій моделі вся обробка інформації перенесена на сервер, а у клієнта права доступу суворо обмежені. Сервер відправляє відповідь, яка не вимагає додаткової обробки. Клієнт

взаємодіє з користувачем: складає та відправляє запит, приймає результат і виводить інформацію на екран.

Сильний клієнт – концепція, в якій частина обробки інформації надається клієнтові. У такому випадку сервер виступає сховищем даних, а вся робота по обробці та подання інформації переноситься на комп'ютер клієнта.

Особливості сервера

Сервер — комп'ютер, такий же, як у нас, лише набагато потужніший. Основне завдання сервера – безперебійна робота та можливість обробляти мільйони запитів від користувачів.

Сервер дозволяє не дублювати програми. Без нього для замовлення продуктів довелося б завантажити весь сайт до себе на комп'ютер, вибрати товари, записати їх та відправити на комп'ютер магазину. Оскільки сайт знаходиться на сервері, тисячі людей можуть звертатися до одного сервера і отримувати від нього потрібну інформацію.

Якщо сервер виконує функції програми та бази даних, то така архітектура називається дворівневою. Такий підхід використовують для невеликих програм, де немає великої кількості клієнтів. Хоч такий спосіб і простіший, але його надійність невелика. Якщо сервер зламають, то зловмисники отримають усі дані.

Для вирішення проблеми безпеки в клієнт-серверній архітектурі використовують базу даних. Вона зберігається окремо від сервера. Сервер у разі виконує роль логічної машини, яка обробляє дані, але з зберігає їх.

Особливості бази даних

У клієнт-серверній архітектурі сервер - це не тільки комп'ютер, на якому знаходиться програма або сайт. Ще це база, де зберігаються всі дані програми. У клієнтів немає прямого доступу до бази даних, оскільки це порушило їх приватність. Наприклад, зокрема особисту інформацію інших користувачів у соціальних мережах.

Клієнти запитують інформацію на сервері. Якщо сервер вважає, що клієнт має права на отримання інформації, то він її надає. Завдяки цьому ми не

можемо користуватися обліковими записами своїх друзів у соціальних мережах або отримувати інформацію про банківські перекази незнайомих нам людей.

Рис. 1.2 – Трирівнева клієнт-серверна архітектура

За такої схеми роботи архітектура називається трирівневою, оскільки складається з трьох компонентів.

Як і в будь-якого рішення, клієнт-сервер має плюси і мінуси. Розберемо часті випадки.

Плюси та мінуси клієнт-серверної архітектури

Найчастіше мінуси клієнт-серверної архітектури пов'язані з працездатністю сервера чи бази даних. Розробники можуть вирішувати більшість мінусів, але не все так просто. Рішення одних мінусів призводить до інших, найчастіше зростання вартості розробки та підтримки.

Розглянемо плюси та мінуси клієнт-серверної архітектури та почнемо з гарних сторін.

Плюси

Відсутність дублювання. Весь сайт або програма зберігається на одному комп'ютері-сервері. Це дозволяє використовувати його з різних пристроїв, будь то комп'ютер або мобільний телефон

Мінімальні вимоги до користувача. Від нього потрібна лише наявність програми-клієнта. Для роботи із сайтами достатньо мати браузер

Безпека. Дані зберігаються в базі даних, і користувачі не можуть їх переглядати. Це забезпечує безпеку для персональних даних

Продуктивність. Сервери зазвичай продуктивніші, ніж комп'ютери користувачів. Це дозволяє обробляти тисячі запитів від сотні різних користувачів одночасно.

Мінуси

Перевантаження сервера. Популярні портали можуть отримувати багато запитів одночасно. Наприклад, при десятці мільйонів запитів на секунду сервер може не витримати та вимкнутись. Цим користуються хакери під час використання DDoS-атаки.

Вихід з ладу сервера чи бази даних. Це зробить сервіс недоступним для всіх користувачів

Висока вартість обладнання. Сервер схожий на простий комп'ютер, але його комплектуючі мають бути розраховані на безперебійну роботу 24/7. Така надійність забезпечується компонентами зі спеціальними функціями, через що вартість обладнання підвищується.

Витрати підтримки. Зазвичай недостатньо просто отримати сервер і забути про його існування. Має бути спеціаліст, який обслуговуватиме сервер і швидко реагуватиме у разі поломок

Щоб позбавитися більшості перерахованих мінусів, розробники використовують кластери серверів.

Кластери серверів та балансувальники

Щоб уникнути проблем із клієнт-серверною архітектурою, розробники застосовують кластери серверів - коли програма знаходиться не на одному, а відразу на декількох серверах. Інформація на таких серверах дублюється:

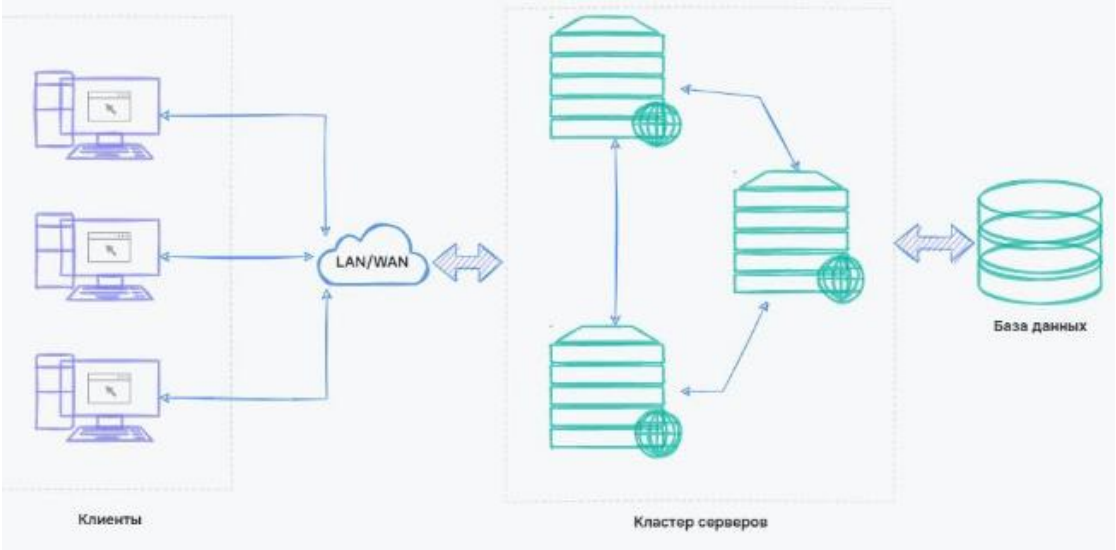


Рис. 1.3 – Застосування кластерів серверів

Якщо один сервер вийшов з ладу, його підміняє інший. Це дає час, щоб усунути несправності сервера так, що користувачі не помітять.

Ще кластери використовуються, щоб розподіляти навантаження. Ми вже говорили, що сервер може відключитись від десятка мільйонів одночасних запитів. Кластери можна налаштувати так, щоб за десяти мільйонів запитів вони пішли не на один сервер, а на різні.

Щоб визначити, на який сервер надіслати запит, використовується балансувальник — сервіс, який пропускає всі запити через себе та стежить, щоб сервери не перевантажувалися.

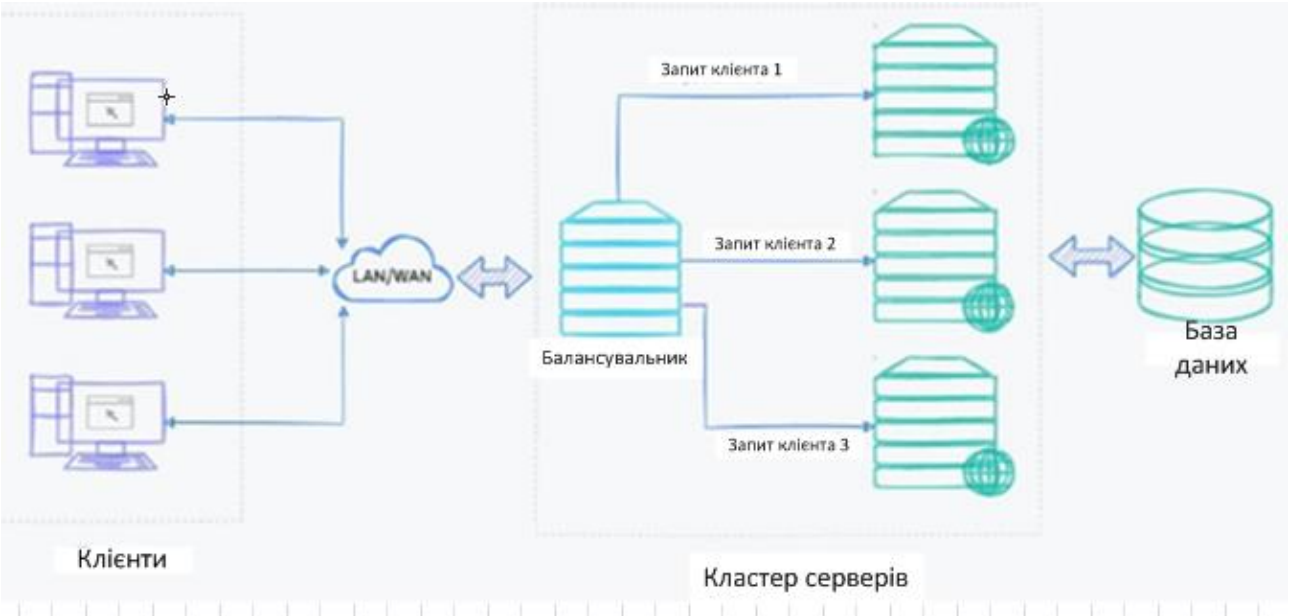


Рис. 1.4 – Використання балансувальника

Балансувальники та кластери можуть використовуватися не тільки для серверів з програмою, але і для баз даних. Так можна отримати незалежні кластери всієї програми, які зможуть підміняти один одного та допомагати в обробці запитів від користувачів.

Такі рішення підвищують надійність програми, але сильно зростає вартість обладнання та підтримки. Тому невеликі компанії та приватні сайти не використовують їх. І тут відповідальність за працездатність беруть хостинги — компанії, які надають послугу оренди серверів додатків.

Висновки

Сайти та застосунки в Інтернеті працюють за клієнт-серверною архітектурою

Клієнт — програма, яка забезпечує зв'язок із сервером і доступна користувачам без технічних знань.

Сервер — це комп'ютер, на якому зберігається сайт або програма. Сервери спілкуються з клієнтами та базами даних

Для забезпечення надійності використовуються кластери серверів та баз даних

У кластерах для керування потоком запитів використовуються балансувальники - сервіси, що розподіляють навантаження.

1.3 Постановка завдання

Метою роботи є розробка клієнт-серверної системи для виконання математичних операцій.

До основних завдань, вирішення яких веде до досягнення поставленої мети, належать:

1. Аналіз клієнт-серверної архітектури, аналіз особливостей взаємодії клієнта та серверу.

2. Розробка схеми процесу клієнт-серверної взаємодії при виконанні математичних операцій.

3.Розробка інтерфейсу користувача системи виконання математичних операцій.

4. Описання програмної частини для розробника.

5. Реалізація клієнт-серверної системи для виконання математичних операцій.

РОЗДІЛ 2
АНАЛІЗ ТА ВИБІР ЗАСОБІВ РОЗРОБКИ ПРОГРАМ КЛІЄНТ-СЕРВЕР

2.1 Аналіз засобів розробки програм клієнт-сервер

2.1.1 Середовища розробки додатків для серверів баз даних

Групу інструментальних засобів для створення інформаційних програм з архітектурою клієнт-сервер можна розділити на наступні підгрупи: середовища розробки додатків для серверів баз даних, незалежні від СУБД, інструменти для створення додатків клієнт-сервер; засоби підтримки розподілених інформаційних програм.

Середовища розробки додатків для серверів БД є системами програмування четвертого покоління 4GL або інструментальні засоби швидкої розробки додатків RAD (Rapid Application Development). Особливостями цієї підгрупи є: реалізація віддаленого доступу до СУБД за дволанковою схемою клієнт-сервер; зв'язок клієнтських додатків із серверами БД за допомогою непроцедурної мови структурованих запитів SQL (крім серверів Btrieve); забезпечення цілісності БД, включаючи цілісність транзакцій; підтримка процедур, що зберігаються на серверах БД; реалізація клієнтських та серверних тригерів-процедур; генерація елементів діалогового інтерфейсу та звітів.

Як приклад можна назвати інструменти Informix/4GL, Oracle*Forms та ін. Нині нові середовища розробки SQL-серверів БД (Informix NewEra та Oracle Power Objects) розвиваються у бік незалежних від СУБД інструментів. Незалежні інструментальні засоби, орієнтовані на багато платформ СУБД, представлені у вигляді засобів швидкої розробки додатків RAD. Для таких засобів створення програм клієнт-сервер характерні: можливість розподілу програми на клієнтах та/або серверах; створення програм для різних серверів БД; підтримка специфікації ODBC доступу до різних серверів БД, включаючи СУБД для ПК; зв'язок із моніторами транзакцій для організації триланкової архітектури додатків клієнт-сервер; об'єктно-орієнтоване програмування додатків; візуальний характер генерації програми; ведення репозитарію об'єктів

та їх властивостей, що полегшує інтеграцію із засобами автоматизації проектування програм CASE; управління проектами та версіями додатків; інтеграція програми з електронною поштою та засобами офісної автоматизації.

Відомими прикладами незалежних інструментальних засобів розробки є ErWin, SQLWindows, PowerBuilder, JAM і Uniface.

2.1.2 Засоби підтримки розподілених інформаційних програм

Засоби підтримки розподілених програм належать до категорії проміжного програмного забезпечення middleware для організації серверів програм. Сюди входять різноманітні бібліотеки та набори інструментальних засобів: інтерфейси доступу до баз даних ODBC та IDAPI; шлюзи для систем керування базами даних; протоколи та команди моніторів обробки транзакцій; поштові інтерфейси MAPI, VIM, MHS, X.400 та EDI; засоби обміну повідомленнями MOM; протоколи зв'язування та включення об'єктів OLE та динамічного обміну даними DDE; протоколи віддаленого виклику процедур RPC та іменованих конвеєрів Named Pipes, засоби комунікаційного вводу-виводу BSD Sockets та WinSock.

Інструментальні набори для розробки додатків клієнт-сервер необхідно вибирати, виходячи з таких критеріїв (див. таблицю 2.1): наявність об'єктно-орієнтованої інфраструктури, можливості розподілу додатків між клієнтом та сервером, чи забезпечена підтримка моніторів транзакцій, доступність CASE-репозитарію, можливість перенесення додатків та контроль версій. При цьому слід з'ясувати, наскільки досвід розробників підприємства відповідає вимогам продукту, чи важлива переносимість додатків на інші апаратні платформи та бази даних, який ступінь інтеграції додатків влаштує замовника і чи потрібно буде надалі підключати до додатка додаткових користувачів, функції та дані.

Крім того, розвиток сучасних програмних засобів призводить до розширення їх функціональних можливостей, у результаті програмні забезпечення різних типів конкурують один з одним. Так, продукт Borland C++ Builder перетворює компілятор Borland Visual C++ на повноцінне середовище

розробки додатків в архітектурі клієнт-сервер. Пропонований продукт доповнює C++ візуальними "дизайнерами", інтуїтивними "майстрами" та засобами доступу до об'єктно-орієнтованих даних, зберігаючи знайоме оточення Visual C++.

Таблиця 2.1

Інструментальні набори для розробки програм клієнт-сервер

Продукт/компанія	Об'єктно-орієнтована інфраструктура	Розподіл програм між клієнтом та сервером	Підтримка моніторів транзакцій	CASE-репозитарій	Перенесення додатків та контроль версій
JAM компанії JYACC	ні	так	так	ні	ні
New Era компанії Informix	так	ні	ні	так	так
Developer 2000 компанії Oracle	ні	так	так	так	так
Power Builder	так	ні	так	так	так
Delphi компанії Borland	так	ні	так	так	так
MS-Access компанії Microsoft	ні	ні	ні	ні	ні
Oracle Power Object компанії Oracle	так	ні	ні	ні	так

Потужний засіб Oracle Forms з набору Developer/2000 призначений для створення додатків баз даних у середовищі клієнт/сервер, які можуть бути перенесені на платформи з різними графічними та символьними інтерфейсами користувача. Oracle Forms є частиною Developer/2000, який підтримує розробку програм під час всього життєвого циклу. Програми, створені за допомогою Developer/2000, повністю масштабуються та застосовні на будь-якому рівні: від систем підтримки прийняття рішень для невеликих робочих груп до проектів з великим обсягом транзакцій, які підтримують сотні користувачів. Програми,

створені за допомогою Developer/2000, оптимізовані з метою використання всіх переваг сервера Oracle7, тому вони мають бути основними засобами розробки додатків у середовищі Oracle7.

Інструментальне середовище NewEra для СУБД Informix має всі властивості для ефективної розробки додатків у цьому середовищі. Додаткові переваги – можливість інтеграції з програмами на С та багатоплатформність – роблять її придатною не тільки при розробці додатків для СУБД Informix, але й для інших систем. Слід зауважити, що питання інтероперабельності Informix-Oracle вирішено незадовільно.

Uniface підтримує інтерфейс практично з усіма відомими програмно-апаратними платформами, протоколами, СУБД та моніторами транзакцій. Цей засіб необхідно використовувати при розробці та супроводі типових комплексів додатків з високою тиражуванням. Платою за універсалізм є висока ціна товару.

Аналіз та апробація можливостей MS Access показав, що цей інструментальний засіб добре зарекомендував себе як у розробці файл-серверних додатків, так і для розробки клієнтської частини додатків в архітектурі клієнт/сервер, наявність підтримки мови SQL та інтерфейсу ODBC відкриває доступ до SQL-серверів БД . Є засіб для міграції програм MS Access у середу MS SQL Server. До переваг Access слід віднести і знижені вимоги до ресурсів. На жаль, останні версії пакета орієнтовані лише на офісну автоматизацію та не містять runtime-компонент для створення закінченої інформаційної програми.

Засіб JAM має недостатню розрядність і може бути використаний тільки в додатках, що не потребують високої точності, наприклад, для створення аналітичних систем. Але його відрізняє багатоплатформність та підтримка моніторів транзакцій.

Пакет Oracle Power Object призначений для розробників, які вперше приступають до розробки програм клієнт-сервер і переходять від таких систем,

як FoxPro або Clipper, і найбільш придатний для створення прототипів великих систем.

Система Delphi надзвичайно зручна для розробки програм локальних баз даних, які при необхідності можуть бути конвертовані в додатки типу клієнт-сервер. Delphi слід використовувати для створення масштабованих додатків для робочих груп, для розробки засобів доступу до різних БД, для створення аналітичних систем, для створення одиночних та групових програм, критичних за часом виконання.

Всі три засоби - JAM, Oracle Power Object та Delphi - придатні для створення швидких прототипів, і їх використання в такій якості може мати певні переваги.

2.2 Технології реалізації клієнт-серверної системи, що обрано для реалізації проекту

Для реалізації програмного забезпечення та вирішення задачі даного дипломного проекту була обрана клієнт-серверна технологія ASP.NET WEB API та мова програмування C#. Середою розробки для обраної технології є Microsoft Visual Studio, що має безліч інструментів для створення програмного забезпечення. Інтегроване середовище розробки Visual Studio використовується для редагування, налагодження та складання коду, а також для публікації програм.

Десктопна версія клієнт-серверної системи для операційної системи Windows реалізована за допомогою WPF-платформи, яка дозволяє створювати фронтальну частину програмного забезпечення за допомогою різних інструментів та патернів, задовольняючи будь-які потреби розробників та клієнтів, які можуть замовити програмний продукт для власного або комерційного використання.

C# (C-Sharp) - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона використовується для створення різноманітних програм, включаючи веб-додатки, настільні застосунки, ігри та веб-служби. C#

є частиною платформи .NET і надає розширену підтримку для роботи з об'єктами, подіями, лямбда-виразами, LINQ (Language Integrated Query) та іншими сучасними програмними концепціями. Вона також володіє високим рівнем безпеки, включаючи автоматичне управління пам'яттю через систему сміття, та дозволяє розробникам ефективно працювати з платформами Windows і багатьма іншими.

ASP.NET Web API - це фреймворк для розробки веб-служб в середовищі .NET. Використовується для створення HTTP-серверів, які надають дані та послуги через RESTful інтерфейс. Web API полегшує роботу з HTTP, автоматично визначає формати виводу (наприклад, JSON), і надає зручний інтерфейс для роботи з ресурсами через HTTP-методи (GET, POST, PUT, DELETE). Основна ідея - спростити створення веб-служб, забезпечити їх гнучкість та легкість розширення в .NET-екосистемі.

Використання мови програмування C# для розробки клієнт-серверних систем має кілька переваг. По-перше C# є кросплатформеною мовою завдяки ініціативі Microsoft - .NET Core та її наступнику .NET 5 і .NET 6. Це дозволяє розгортати клієнт-серверні додатки на різних операційних системах, включаючи Windows, Linux та macOS. По-друге C# є високорівневою мовою програмування, яка спрощує розробку та зменшує кількість необхідного коду. Об'єктно-орієнтована структура мови дозволяє використовувати концепції такі як спадкування та поліморфізм для зручності програмістів. Також серед переваг є те, що C# розроблена компанією Microsoft, і це важливо для тих, хто використовує інші технології в їхніх проектах, такі як ASP.NET, .NET Framework, або Azure. C# має відмінну інтеграцію з цими технологіями. Для створення веб-додатків або API на стороні сервера можна використовувати ASP.NET. Це потужний інструментарій для створення сучасних та ефективних веб-застосунків. C# має вбудовану систему безпеки, яка допомагає запобігти типовим помилкам безпеки, таким як переповнення буфера. Також, вона дозволяє використовувати атрибути безпеки для контролю доступу до ресурсів. А також не менш значимим є те, що C# має вбудовану

підтримку асинхронного та паралельного програмування, що дозволяє ефективно обробляти багатозадачні сценарії та підвищує продуктивність додатків. Враховуючи ці переваги, C# є привабливим вибором для клієнт-серверних систем, особливо для розробників, які працюють в екосистемі.

Архітектура REST

Архітектура REST (Representational State Transfer) є стилем архітектури програмного забезпечення для розподілених систем, таких як Інтернет. Цей стиль був вперше описаний в дисертації Роя Філдінга в 2000 році і є основою для багатьох веб-служб та API. Клієнт-серверна модель: цей принцип визначає розподіл системи на дві основні складові - сервер, який забезпечує ресурси, і клієнт, який їх використовує. Це спрощує систему, дозволяючи їй еволюціонувати незалежно від інших компонентів.Кожен запит від клієнта до сервера містить всю інформацію, необхідну для розуміння і обробки запиту. Сервер не зберігає жодної інформації про стан клієнта між запитами. Це полегшує масштабування та підтримку великої кількості клієнтів. Сервер або клієнт можуть зберігати відповіді на запити(ресурси) для подальшого використання. Це зменшує обсяг мережевого трафіку і покращує швидкість відповідей.

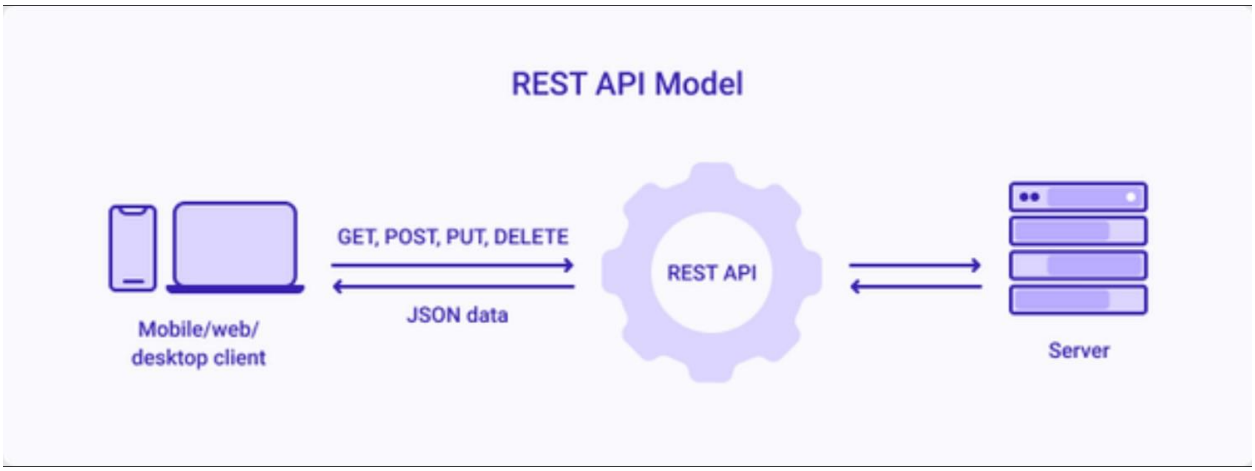


Рис. 2.1 - Архитектура REST API Model

Єдина точка доступу (Uniform Interface): Цей принцип стверджує, що інтерфейс до кожного ресурсу повинен бути єдиною і чіткою точкою доступу.

Це полегшує розуміння та використання системи. Архітектура може бути складеною з різних рівнів, при цьому кожен рівень може бачити лише рівні, що безпосередньо знаходяться вище або нижче нього. Це полегшує розподіл системи та поліпшує її масштабованість. Код на вимогу (Code on Demand): Цей принцип дозволяє розширювати функціональність клієнта, завантажуючи та виконуючи код від сервера, наприклад, у вигляді сценаріїв або аплетів. Усі ці принципи дозволяють створювати гнучкі та легкі системи, які можуть еволюціонувати з часом та взаємодіяти з різними клієнтами та серверами. Архітектура REST зазвичай використовується для розробки веб-служб та API, де важлива простота взаємодії та невелика залежність від внутрішньої структури сервера.

Кожен ресурс має свій унікальний ідентифікатор, який може бути використаний для доступу до нього. У веб-архітектурі, URL (Uniform Resource Locator) зазвичай використовується як ідентифікатор ресурсу. Представлення ресурсу (Resource Representation): Ресурс може бути представлений у різних форматах, таких як JSON, XML тощо. Клієнт та сервер взаємодіють, обмінюючись цими представленнями ресурсів.

Методи HTTP:

GET: Використовується для отримання ресурсу або списку ресурсів.

POST: Використовується для створення нового ресурсу.

PUT: Використовується для оновлення існуючого ресурсу або створення нового, якщо його не існує.

DELETE: Використовується для видалення ресурсу.

PATCH: Використовується для часткового оновлення ресурсу.

HTTP-статус-коди вказують результат виконання запиту. Наприклад, код 200 означає успішний запит, 404 - ресурс не знайдено, 201 - ресурс успішно створено тощо. Принцип, що ресурс повинен містити гіперпосилання (гіпермедіа), яке дозволяє клієнтові взаємодіяти з системою, не знаючи всіх деталей перед цим. Гіпермедіа вказує на доступні дії та наступні кроки для клієнта.

Формати обміну даними:

- JSON (JavaScript Object Notation): Легкий та зрозумілий формат обміну даними, часто використовується в REST-архітектурі.
- XML (eXtensible Markup Language): Інший формат для обміну структурованими даними.

Кожен запит містить всю необхідну інформацію для сервера для розуміння та обробки запиту. Відсутність стану на сервері між запитами дозволяє легше масштабування системи. Ці елементи та концепції роблять архітектуру REST ефективною для розробки веб-систем, що мають високу читабельність коду, легку масштабованість та здатність пристосовуватися до змін. Основна ідея полягає в тому, щоб створити простий та легко зрозумілий інтерфейс для взаємодії між клієнтом та сервером.

ASP.NET Web API - це фреймворк, розроблений Microsoft, призначений для створення HTTP-сервісів, які можна використовувати як застосунок веб-служб. Він базується на платформі ASP.NET і дозволяє розробникам легко створювати RESTful веб-служби. ASP.NET Web API надає простий інтерфейс для створення веб-служб. За замовчуванням він підтримує протокол HTTP, для якого не потрібно додатково налаштовувати багато різних аспектів веб-служби. Web API автоматично визначає формат виводу (наприклад, JSON або XML) на основі запиту клієнта. Також можна явно вказати формат, який потрібно використовувати. Механізм маршрутизації в Web API дозволяє визначити, які URL пов'язані з якими методами контролера. Це дозволяє створювати логічно

організовані та зрозумілі URL для вашого API. Модель пам'яті обміну даними (Data Exchange Model): Web API використовує модель пам'яті обміну даними, яка дозволяє автоматично відображати дані між кодом сервера і форматами обміну даними, такими як JSON або XML.

Підтримка аутентифікації та авторизації: Web API має можливості для налаштування механізмів аутентифікації та авторизації, щоб забезпечити безпеку ваших веб-служб. Функціональність Web API можна легко розширювати за допомогою розширень (extensions), що дозволяє додавати власні функції та підтримку. Web API розроблений так, щоб бути легко тестовим, що полегшує розробку і підтримку ваших служб. ASP.NET Web API є частиною фреймворку ASP.NET, і він може використовувати багато його функціональних можливостей, таких як маршрутизація, модель пам'яті обміну даними та інші. По суті Web API є веб-службою, до якої можуть звертатися інші програми. Ці програми можуть представляти будь-яку технологію і платформу - це можуть бути веб-додатки, мобільні або десктопні клієнти.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМИ КЛІЄНТ-СЕРВЕР ДЛЯ ВИКОНАННЯ МАТЕМАТИЧНИХ ОБЧИСЛЕНЬ

3.1 Опис реалізації клієнт-серверної системи

Для реалізації цієї дипломної роботи, задача якої було створення гнучкої клієнт-серверної системи обчислення даних, яка може використовуватися на різних платформах та пристроях була обрана середа розробки Microsoft Visual Studio та мова програмування C#. Для прикладу такої системи був обраний обчислювальний модуль, який реалізує функції калькулятора. Цей модуль вміє обчислювати складні математичні вирази із дотриманням послідовності. Програма створена таким чином, що отримуючи математичний вираз вона розкладує його на більш дрібні вирази та обирає послідовність їх вирішення, у залежності від математичних правил, отримуючи на виході очікуваний результат.

Даний програмний продукт був реалізований за допомогою розподілення задач, які були вирішені окремими модулями програми. Таким чином програма ділиться на наступні модулі: модуль розрахунку, консольний модуль, десктопний модуль, модуль мобільного додатку, серверний модуль, юніт-тести та база даних.

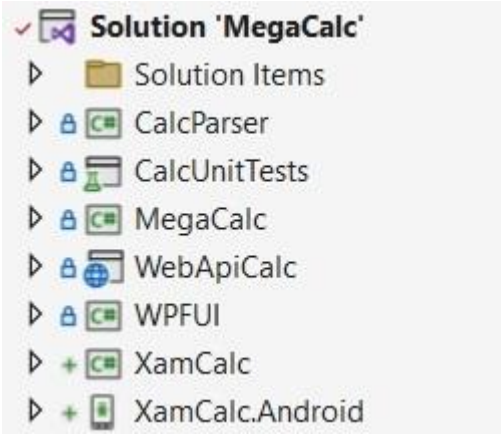


Рис. 3.1 - Набір модулів клієнт-серверної системи у середі Visual Studio

Завдяки такому підходу можна створювати програмний продукт для будь-якої платформи, при цьому, для всіх пристроїв методи обчислення програми будуть однаковими. Також такий метод дозволяє використовувати алгоритми обчислень з будь-якої точки світу, де є Інтернет. Також якщо буде необхідність змінити логіку обчислень або виправити можливу помилку обчислень, то це можна зробити в модулі обчислень на сервері і після цього користувачі всіх пристроїв можуть використовувати оновлений метод обчислень.

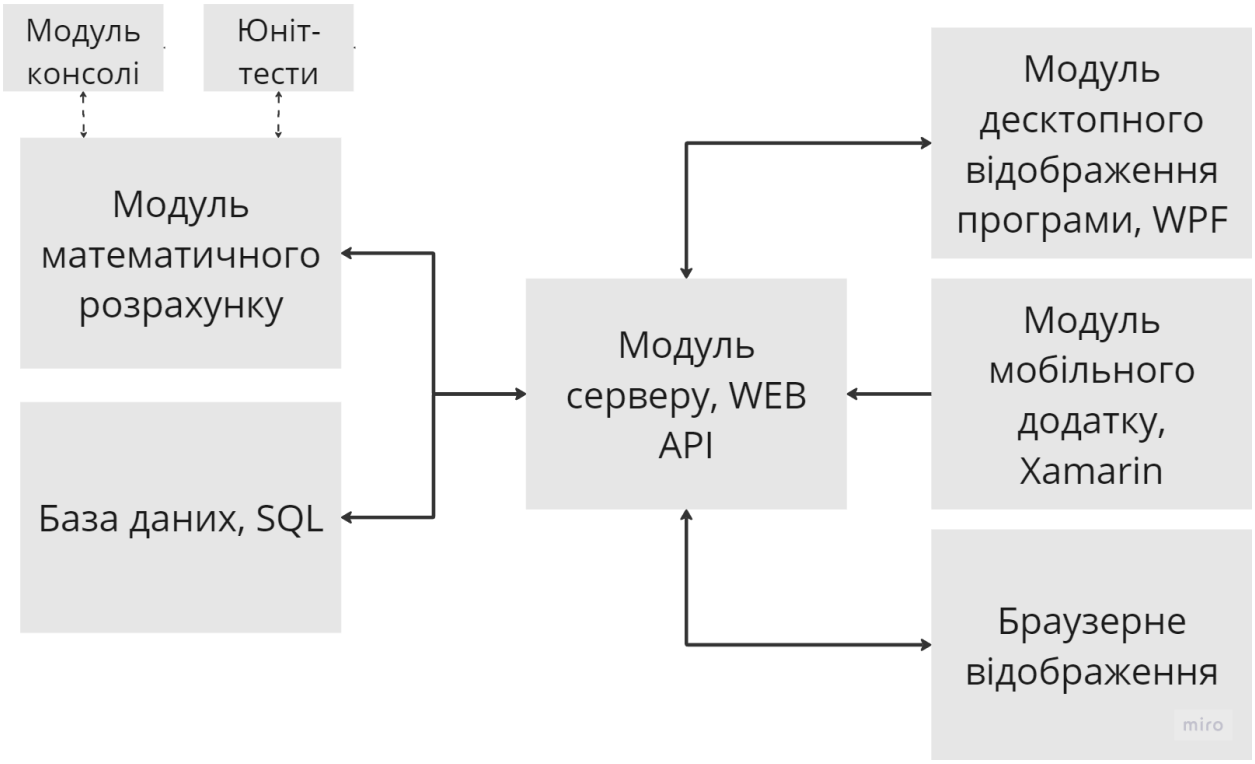


Рис. 3.2 - Схема взаємодії усіх модулей програми

На Рис. 3.2, розташованому вище, видно, що модуль серверу слугує основним модулем для передачі даних між іншими модулями програми. Модулі діляться на дві категорії: клієнтська частина для користувача, що частіше за все називається фронт-ендом та служить для взаємодії користувача з системою модулів, та бєк-ендова, серверна частина, яка і відповідає за обробку інформації, розрахунок та надання результату для клієнтів.

Також однією з переваг розробки такої системи саме за допомогою Visual Studio та мови програмування C# є те, що для розробки мобільних додатків використовується кросплатформенний фреймворк Xamarin. За допомогою цього фреймворку значно спрощується процес розробки для мобільних додатків, бо більшість коду співпадає для різних мобільних операційних систем.

3.2 Опис модулів розробленої системи

3.2.1 Модуль розрахунку

Основною функцією модуля розрахунку є вирішення математичних виразів та отримання результатів згідно до вимог розробки. У даному випадку, для прикладу, була розроблена та використана програма обчислення числового арифметичного виразу у вигляді калькулятора.

Даний модуль реалізован наступним чином: після вводу даних за допомогою одного з методів вводу (консоль, десктопний клієнт, мобільний додаток або інший метод) програма, за допомогою спеціально розробленого алгоритму для обчислення складних математичних послідовностей, робить певні операції, розбиваючи вираз на більш дрібні та тривіальні вирази, які вирішуються згідно математичних правил та послідовностей.

Фрагмент коду, що зображений на Рис. 3.3 демонструє послідовність вирішення виразу методом розбиття складного виразу на більш тривіальні та їх подальше рішення. Для цього спочатку програма перевіряє послідовність на наявність символів, які не пов’язані з математичним виразом. Далі програма сортує дрібні вирази за пріоритетом вирішення згідно до математичних правил. Потім бере кожен дрібний вираз та вирішує його, а результат записується у змінну solution. Якщо рішення не було знайдено за певних причин, то програма видасть відповідне повідомлення.

```
26 references
public double Solve(string str)
{
    str = str.Replace(" ", "");

    var equationSorted = equationParsers.OrderBy(e => e.Priority);

    foreach (var parser in equationSorted)
    {
        if (parser.IsSovable(str))
        {
            if (parser is not DoubleParser)
                Console.WriteLine($"\t{parser.GetType().Name,10}\tsolving {str}");

            var solution = parser.ParseAndSolve(str);

            if (parser is not DoubleParser)
                Console.WriteLine($"\t{parser.GetType().Name,10}\t\t{str}={solution}");

            return solution;
        }
    }

    throw new WrongEquationException("No suitable parsers were found.", str);
}
```

Рис. 3.3 Метод сортування, розбиття на дрібні вирази та вирішення цих виразів

Результат роботи цього фрагменту коду можна побачити на Рис., який демонструє послідовність вирішення виразу, та результат. Як можна бачити, програма розбиває послідовність, що була введена користувачем на кілька більш простих виразів, сортує їх за математичними правилами та вирішує кожен з них, отримуючи очікуваний результат.

```
(5 + 3) / 2
          Brackets      solving (5+3)/2
              Plus      solving 5+3
              Plus              5+3=8
               Div      solving 8/2
               Div              8/2=4
          Brackets              (5+3)/2=4
Result: 4
```

Рис. 3.4 - Результат виконання методу вирішення математичного виразу

Рис. 3.5 ничже, що відображає блок-схему роботи методу Solve дає візуальне уявлення взаємодії блоків цього методу.

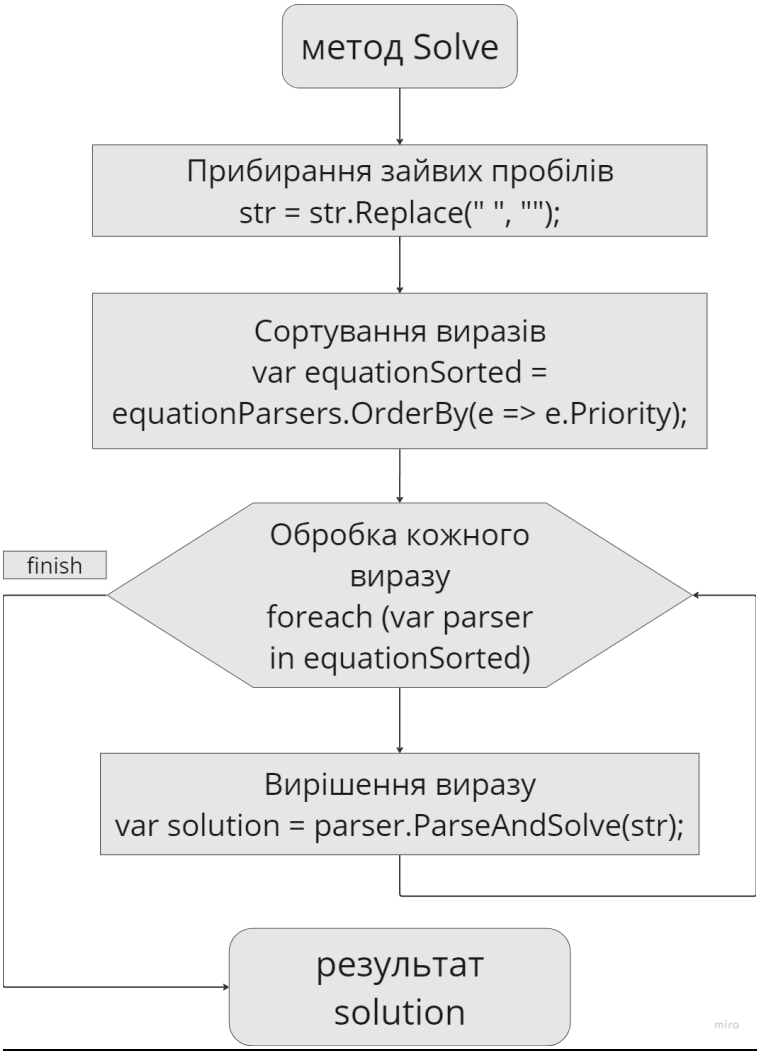


Рис. 3.5 - Блок-схема роботи методу Solve

3.2.2 Консольний модуль

Базовою функцією консолі є ввід та вивід інформації. Консольный модуль слугує для налаштування роботи при розробці логіки для математичних обчислень. Це дозволяє пришвидшити розробку програмного продукту не вдаючись до користувацької частини програми, зосередившись на якості обчислювальних операцій та результатах. Також такий підхід може пришвидшити розробку шляхом паралельної розробки різних частин програми різними спеціалістами. Таким чином спеціалісти одночасно можуть працювати кожен над своєю частиною програми.

Консоль в C# - це інтерфейс командного рядка, який використовується для введення та виведення текстової інформації під час виконання консольних програм. Консоль також підтримує різні інші функції, такі як обробка клавіш,

робота з аргументами командного рядка і багато іншого. Основні класи, пов'язані з консоллю в C#, містяться в просторі імен System. Функції консолі дозволяють реалізовувати різноманітні консольні застосунки, включаючи інтерактивні меню, ігри, інструменти командного рядка та багато іншого.

Існує три потоки вводу-виводу: потік вводу, потік виводу і потік виводу помилок. Ці потоки операційна система автоматично пов'язує з консоллю, коли запускається консольна программа. Програма зчитує вхідні дані стандартного вхідного потоку користувача; записує стандартні дані у стандартний вихідний потік; і записує дані помилок у потік виведення помилок.

```
0 references
internal class Program
{
    0 references
    private static void Main(string[] args)
    {
        BasicEquationSolver solver = new BasicEquationSolver();
        do
        {
            try
            {
                string? expr = Console.ReadLine();

                double result = solver.Solve(expr);

                Console.WriteLine(result);
            }
            catch (WrongEquationException ex)
            {
                Console.WriteLine($"{ex.Message} --- {ex.Equation}");

            }
            catch (BracketsExeption ex)
            {
                Console.WriteLine($"{ex.Message} --- {ex.Equation}");

            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
            }


        } while (true);

    }
}
```

Рис. 3.6 - Програмний код консольного модулю

У даному фрагменті коду програма створює новий об’єкт класу BasicEquiationSolver, викликається його метод Solve і передається вираз, який вноситься у консоль користувачем. Нижче, Рис.3.7 містить скрін екрану, на якому видно результат роботи консольного модулю, та послідовність вирішення виразу, який реалізує модуль розрахунку.

```
-1 * (5.3 + 2.7) / 2.5
          Brackets      solving -1*(5.3+2.7)/2.5
              Plus      solving 5.3+2.7
              Plus              5.3+2.7=8
              Mult      solving -1*8/2.5
               Div      solving 8/2.5
               Div              8/2.5=3.2
              Mult              -1*8/2.5=-3.2
          Brackets              -1*(5.3+2.7)/2.5=-3.2
Final result: -3.2
```

Рис. 3.7 - Результат, що відображає консоль після вирішення математичного виразу

3.2.3 Десктопний модуль

Для використання програми обчислення у операційній системі Windows та взаємодії користувача з програмою був побудований клієнтський десктопний модуль з використанням платформи WPF. За допомогою WPF можна для любих методів математичних обчислень створити зручний та інтуїтивно зрозумілий інтерфейс. Також при наявності вимог щодо додаткового функціоналу платформа WPF дозволяє створювати додаткові вікна з розрахунками для відображення інформації у таблицях. При розробці клієнтської десктопної частини програми можна використовувати сучасний дизайн та стилі, що відповідають вимогам, які замовник продукту пред'являє при замовленні розробки такого типу продукту.

DependencyProperty - це концепція, пов'язана з роботою зі створенням власних властивостей для елементів у віконній системі WPF (Windows Presentation Foundation) в платформі .NET. WPF використовує цю концепцію для створення гнучкого і широкого функціоналу, пов'язаного з прив'язкою даних, анімацією та іншими аспектами створення інтерфейсів. Основні характеристики DependencyProperty: прив'язка до елементів у XAML,

автоматична система визначення залежностей, оптимізація пам'яті та продуктивності, прив'язка даних, анімація та стилі, метадані.

DependencyProperty дозволяє визначити властивості для елементів у XAML (eXtensible Application Markup Language) - декларативній мові розмітки для WPF. Властивості, які потрібно прив'язати або анімувати, можуть бути визначені як DependencyProperty. DependencyProperty використовує систему залежностей для автоматичного визначення, які властивості взаємозалежні та які можуть бути оновлені автоматично, коли змінюється значення інших властивостей.

```
    />
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="1" Grid.Column="0" Grid.Row="1" >1</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="2" Grid.Column="1" Grid.Row="1" >2</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="3" Grid.Column="2" Grid.Row="1" >3</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="4" Grid.Column="0" Grid.Row="2" >4</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="5" Grid.Column="1" Grid.Row="2" >5</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="6" Grid.Column="2" Grid.Row="2" >6</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="7" Grid.Column="0" Grid.Row="3" >7</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="8" Grid.Column="1" Grid.Row="3" >8</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="9" Grid.Column="2" Grid.Row="3" >9</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="0" Grid.Column="1" Grid.Row="4" >0</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="+" Grid.Column="3" Grid.Row="1" >+</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="-" Grid.Column="3" Grid.Row="2" >-</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="*" Grid.Column="3" Grid.Row="3" >*</Button>
<Button Command="{Binding AddSymbolCommand}" Height="30" Width="60" CommandParameter="/" Grid.Column="3" Grid.Row="4" >/</Button>
<Button Command="{Binding ClearCommand}" Height="30" Width="60" Grid.Column="0" Grid.Row="4" >C</Button>
<Button Command="{Binding SquareRootCommand}"  Height="30" Width="60" Grid.Column="3" Grid.Row="5" >√</Button>
```

Рис. 3.8 - Створення кнопок калькулятора та прив’язка їх властивостей

На Рис.3.8 Видно код програми на мові розмітки XAML, що створює кнопки калькулятора, задаються їх параметри та призначаються певні команди за допомогою методу Dependency property.

DependencyProperty допомагає оптимізувати використання пам'яті та покращувати продуктивність шляхом використання реєстрації властивостей в системі залежностей. Це дозволяє властивостям інтелектуально взаємодіяти між собою. Також властивий для прив'язки даних, що дозволяє зв'язувати значення властивостей об'єкта з джерелом даних, таким як джерело даних в моделі виду (MVVM) або інші елементи у XAML. DependencyProperty дозволяє застосовувати анімацію та стилі до властивостей елементів, надаючи широкі можливості для визначення зовнішнього вигляду та поведінки інтерфейсу.

Ця технологія надає можливість використовувати метадані для DependencyProperty, щоб визначити додаткові характеристики, такі як валідація, значення за замовчуванням тощо. DependencyProperty стає потужним інструментом для створення гнучких та ефективних інтерфейсів у WPF, а його особливості полегшують створення багатофункціональних та інтерактивних додатків.

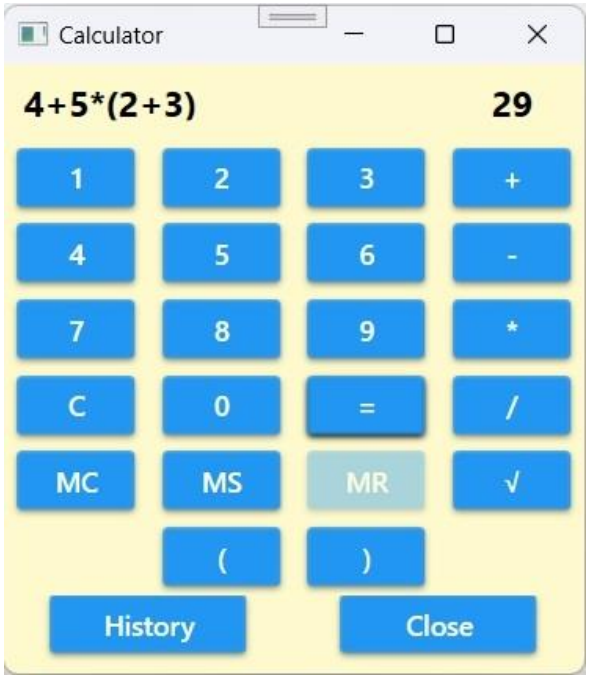


Рис. 3.9 - Інтерфейс клієнтського десктопного модуля

Рис. 3.9 містить зображення екрану користувача десктопного модуля, на якому видно поле вводу математичного виразу, результат, кнопки калькулятора та кнопку історії розрахунків. При натисканні на кнопку “=” програма розраховує вираз та зберігає всю інформацію у базу даних автоматично.

3.2.4 Модуль мобільного додатку

Для того, щоб реалізувати мобільний додаток за допомогою Visual Studio та мови програмування C# була використана платформа Xamarin, що є нативною платформою для цієї середи розробки при розробці мобільних додатків.

Xamarin - це платформа для розробки кросплатформенних мобільних додатків. Крос-платформенність Xamarin дозволяє розробляти мобільні додатки для різних платформ, таких як iOS та Android, використовуючи спільний код. За допомогою нього можна використовувати мову програмування C# та платформу .NET для побудови додатків, які працюють на різних операційних системах. Розробка додатків в Xamarin відбувається мовою програмування C#. Це забезпечує консистентність та повторне використання коду між різними платформами.

Xamarin дозволяє використовувати спільний код для бізнес-логіки, що спрощує розробку і обслуговування додатків. Логіка, яка не залежить від платформи (наприклад, обробка даних, логіка валідації), може бути використана безпосередньо на обох платформах. Ви можете використовувати екосистему .NET для розробки, включаючи багато фреймворків та бібліотек. Xamarin.Forms дозволяє створювати єдиний інтерфейс користувача, який автоматично адаптується під платформи. Xamarin надає доступ до нативних API і бібліотек для кожної платформи. За допомогою нього можна використовувати нативні функції та інструменти, коли це необхідно, зберігаючи при цьому спільний код для логіки застосунку. Розробка Xamarin-додатків часто ведеться в середовищі розробки Visual Studio, що полегшує управління проектами та реалізацію додатків.

Xamarin підтримує не лише розробку мобільних додатків, але і використання в інших областях, таких як Інтернет речей (IoT). Засоби моніторингу та тестування Xamarin дозволяють вивчати та тестувати додатки під час розробки.

Xamarin дозволяє розробникам створювати високоякісні кросплатформенні додатки, зберігаючи переваги мови C# та платформи .NET. Це робить Xamarin популярним вибором для розробників, які хочуть максимально використовувати переваги кросплатформної розробки.

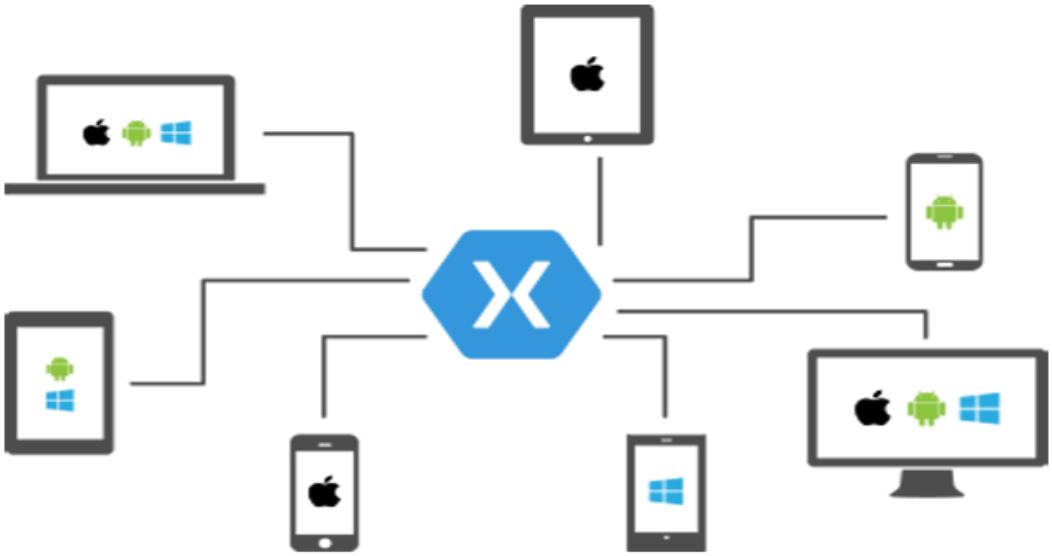


Рис. 3.10 - Можливі варіанти реалізації мобільних додатків за допомогою кросплатформної розробки Xamarin

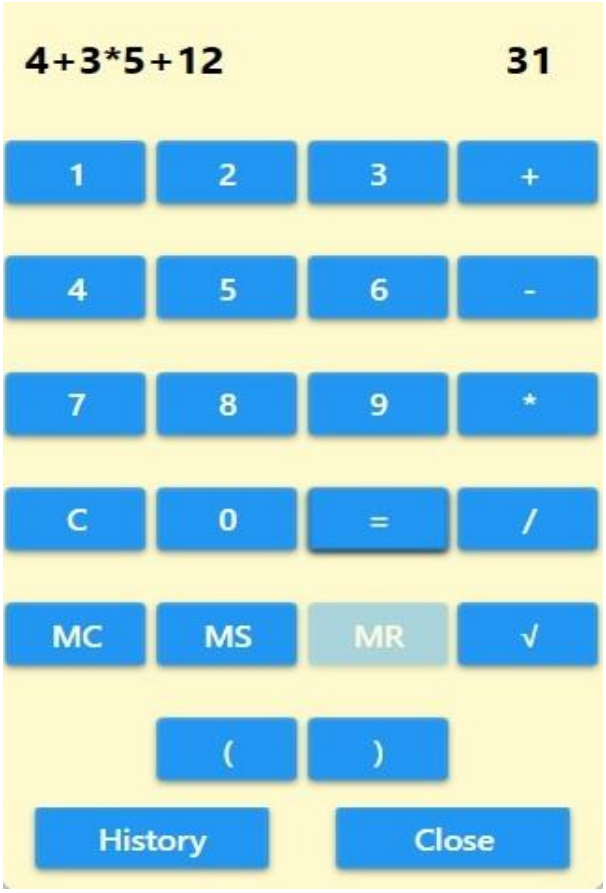


Рис. 3.11 - Скрін мобільного додатку

Економія ресурсів, перевикористання коду та простота додавання версій для нових платформ. Код C# доступний всій команді і при достатній кваліфікації може бути використаний на будь-якій іншій платформі: як мобільної, так і десктопної.

3.2.5 Серверний модуль

Клієнт-серверна архітектура є розподіленою архітектурою моделлю, яка передбачає розділення функцій програмного забезпечення на дві основні частини: клієнт і сервер. Кожна з цих частин виконує специфічні завдання та взаємодіє з іншою частиною через мережу.

Сервер - це програма або комп'ютер, яка надає певні послуги або ресурси для клієнтів через мережу. Вона служить для обробки запитів, забезпечення доступу до даних, виконання бізнес-логіки та інших операцій.

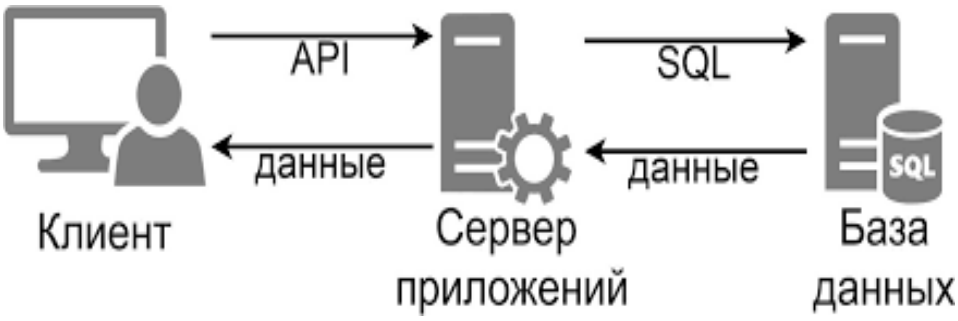


Рис. 3.12 - Схема взаємодії клієнт-серверного додатку

Основними функціями сервера є надання ресурсів, обробка запитів, забезпечення безпеки, управління ресрусами. Сервер надає різні ресурси, аткі як файли, бази даних, обчислювальні можливості, або інші послуги, які клієнти можуть використовувати. Сервер приймає запити від клієнтів і виконує відповідні дії або повертає необхідні дані. Сервер може виконувати різні заходи для забезпечення безпеки, такі як автентифікація, авторизація та шифрування даних. Також він може взаємодіяти з різними ресурсами, такими як бази даних, файлові системи, апаратне забезпечення і т.д.

Сервер приймає запити від клієнтів через мережеві канали (зазвичай за допомогою протоколів зв'язку, таких як HTTP, TCP/IP). Він аналізує отримані запити, визначає їхній тип та виконує відповідні операції для задоволення клієнтських потреб. Сервер генерує відповіді на запити та надсилає їх назад клієнтам через мережу.

Сервер може зберігати і управляти інформацією про сесії користувачів для збереження стану між запитами. Деякі сервери можуть зберігати стан або інформацію про клієнта, щоб спростити взаємодію та поліпшити продуктивність.

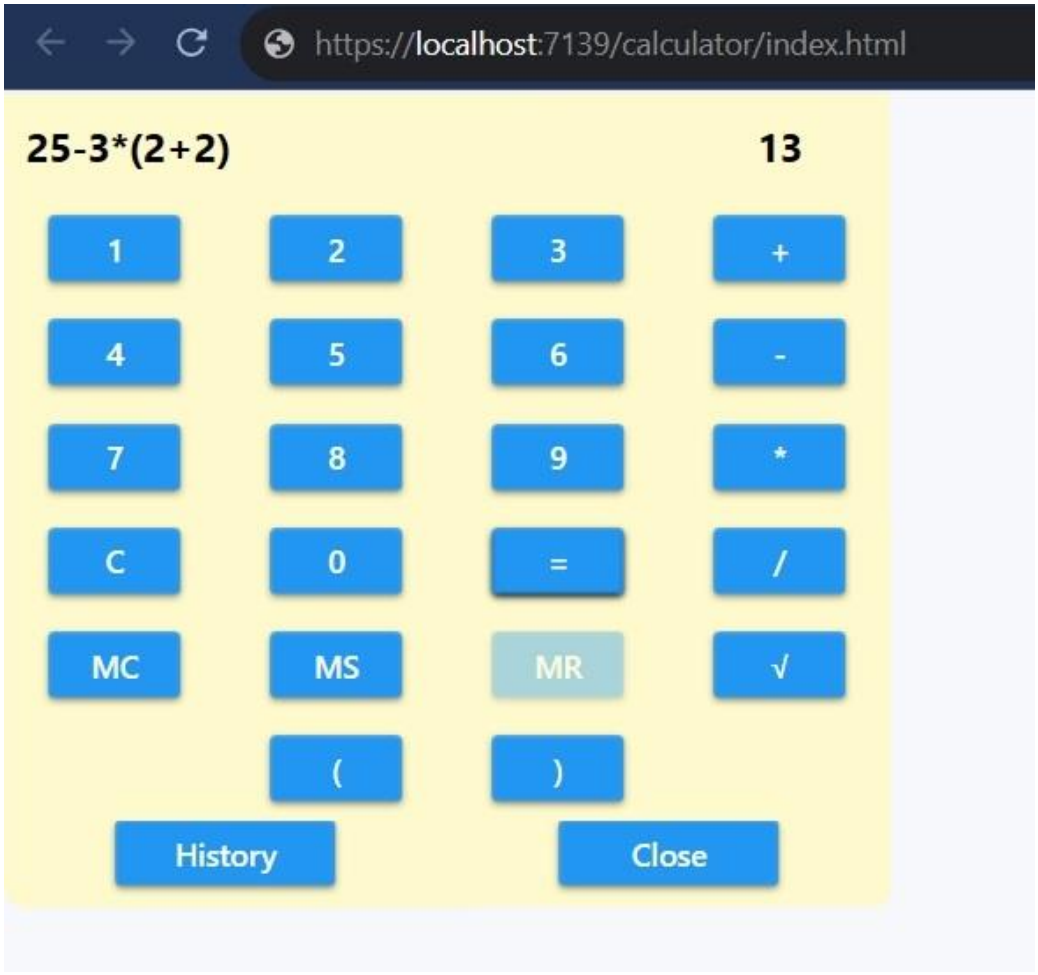


Рис. 3.13 - Браузерний тип клієнта для модулю математичних розрахунків

Сервер може бути масштабований горизонтально або вертикально для обробки збільшеної кількості запитів. Повинен надавати надійність та доступність, а також можливості моніторингу та відладки.

3.2.6 Модуль юніт-тестів

Модуль юнит-тестів був створений для того, щоб виключити можливість появи помилок на стадії релізу. Також цей модуль при розробці дозволяє скоротити час на тестування за рахунок того, що тести проводяться автоматично. В даному випадку юніт-тести написані для модуля розрахунку, але вони корисні і при розробці будь-яких інших видах програмного забезпечення.

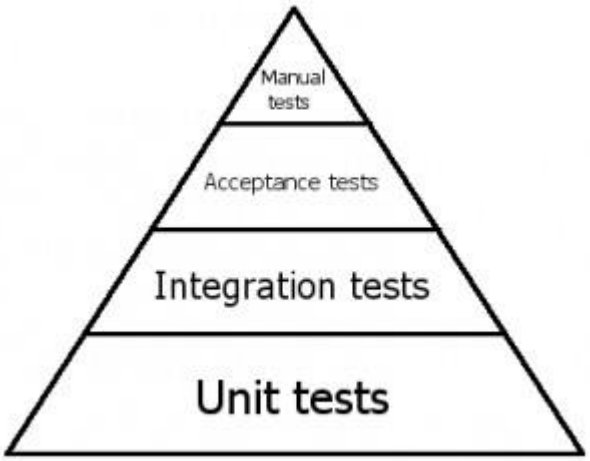


Рис. 3.14 - Ієрархія тестування програмного продукту

До запуску програми у виробництво, коли він стане доступним користувачам, важливо переконатися, що ця програма функціонує згідно до вимог, що були визначені на стадії розробки та що в ньому процедурних немає помилок. Для перевірки програми ми можемо використовувати різні схеми та механізми тестування. Одним із таких механізмів є юніт-тести.

Юніт-тести (Unit tests) у C# є частиною процесу тестування програмного забезпечення, спрямованого на перевірку окремих "юнітів" або найменших можливих фрагментів коду, таких як методи або функції. Юніт-тести дозволяють вам перевіряти правильність виконання окремих частин коду та забезпечують впевненість у його працездатності. У C# існує кілька популярних бібліотек для юніт-тестування, таких як NUnit, MSTest, xUnit і т.д. Оберіть ту, яка найкраще відповідає вашим потребам. Для даного проекту були обрані xUnit-тести.

Для кожного проекту в рішенні створюється окремий проект для юніт-тестів. Це може бути проект типу "Unit Test Project" у Visual Studio. Після чого додається посилання на бібліотеку для тестування, яку обрали. Для проведення тестів використовуються атрибути бібліотеки тестування для вказівки на тести, встановлення передумов, ініціалізації, та інші аспекти. Для кращого управління та виконання тестів їх групують, для цього використовують атрибути, які підтримують групування тестів.

Для виявлення проблем та недоліків у коді переглядають та аналізують результати тестів, для цього використовують вихідні дані, які надаються тестовим середовищем. У випадку провалених тестів помилки у коді виправляються та тести запускаються повторно. Для більш якісного тестування використовуються інструменти для визначення покриття коду тестами, щоб переконатися, що тестуються всі шляхи в коді. Якщо є потреба то додаються додаткові тести для тестування нового функціоналу або покриття певних умов.

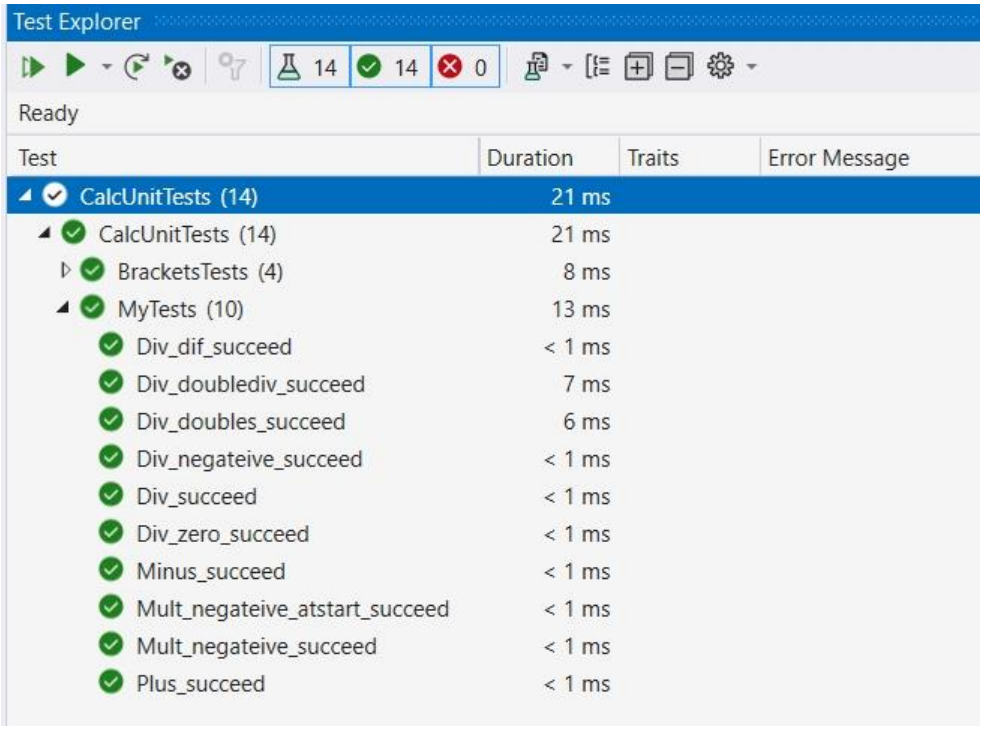


Рис. 3.15 - Набір юніт-тестів для тестування якості математичних розрахунків, що були успішно пройдені

Приклади коду для двох юніт-тестів зображено на Рис . , що нижче. У даному випадку обидва тести відпрацьовують вирази з множенням від'ємних чисел. Для виявлення проблем та недоліків у коді переглядають та аналізують результати тестів, для цього використовують вихідні дані, які надаються тестовим середовищем. У випадку провалених тестів помилки у коді виправляються та тести запускаються повторно. Для більш якісного тестування використовуються інструменти для визначення покриття коду тестами, щоб переконатися, що тестуються всі шляхи в коді. Якщо є потреба то додаються додаткові тести для тестування нового функціоналу або покриття певних умов.

```
[Fact]
| 0 references
public void Mult_negateive_atstart_succeed()
{
    BasicEquationSolver solver = new BasicEquationSolver();

    double result = solver.Solve("-2*5");
    Assert.Equal(-10, result);
}

[Fact]
| 0 references
public void Mult_negateive_succeed()
{
    BasicEquationSolver solver = new BasicEquationSolver();

    double result = solver.Solve(" 2 * - ( 5 + 5 ) ");
    Assert.Equal(-20, result);
}
```

Рис. 3.16 - Код програми, що виконує автоматичне тестування математичних розрахунків програми

Test-Driven Development (TDD) - це методологія розробки програмного забезпечення, яка базується на написанні тестів перед написанням фактичного коду продукту. Основна ідея TDD полягає у тому, щоб визначити очікувану поведінку програми за допомогою тестів, написати мінімальний код, який дозволяє пройти ці тести, а потім покращувати та рефакторювати код. TDD допомагає створювати надійний та ефективний код, забезпечуючи високий рівень тестової покритості. Вона сприяє вдосконаленню архітектури,

зменшенню кількості помилок та полегшенню розробки нових функцій. Застосування TDD сприяє створенню гнучкого та легко змінюваного коду.

3.2.7 База даних

Для реалізації збереження інформації для даної системи була обрана база даних MS SQL. Microsoft SQL Server (MS SQL) - це система управління базами даних, розроблена компанією Microsoft. MS SQL Server використовується для зберігання та управління даними у реляційному форматі. SQL Server використовує клієнт-серверну архітектуру, де сервер відповідає за зберігання та обробку даних, а клієнти здійснюють взаємодію з сервером.

База даних (БД) - це систематизована та структурована колекція даних, яка зберігається та управляється з допомогою спеціального програмного забезпечення. БД використовується для ефективного зберігання, організації, обробки та взаємодії з інформацією. Вона дозволяє зберігати величезні обсяги даних і забезпечує доступ до них за допомогою мови запитів.

Дані в БД організовані у вигляді таблиць, які мають фіксовану структуру з колонками та рядками. База даних дозволяє встановлювати зв'язки між різними таблицями, що спрощує отримання зв'язаної інформації. Здатність створювати резервні копії даних та відновлювати їх у випадку втрати або пошкодження. Індекси використовуються для швидкого пошуку даних та оптимізації запитів. Оптимізатор запитів може автоматично оптимізувати запити для покращення продуктивності.

На Рис. 3.17, що наведений нижче, зображений скрін вікна, що містить таблицю, яка відображає попередні розрахунки, які були збережені у базі даних. У цій таблиці зберігається вираз, що був розрахований, результат та час, коли це відбулось. А на Рис. Видно SQL код, що відображає створення таблиці у базі даних згідно параметрів, що потрібно зберегти.

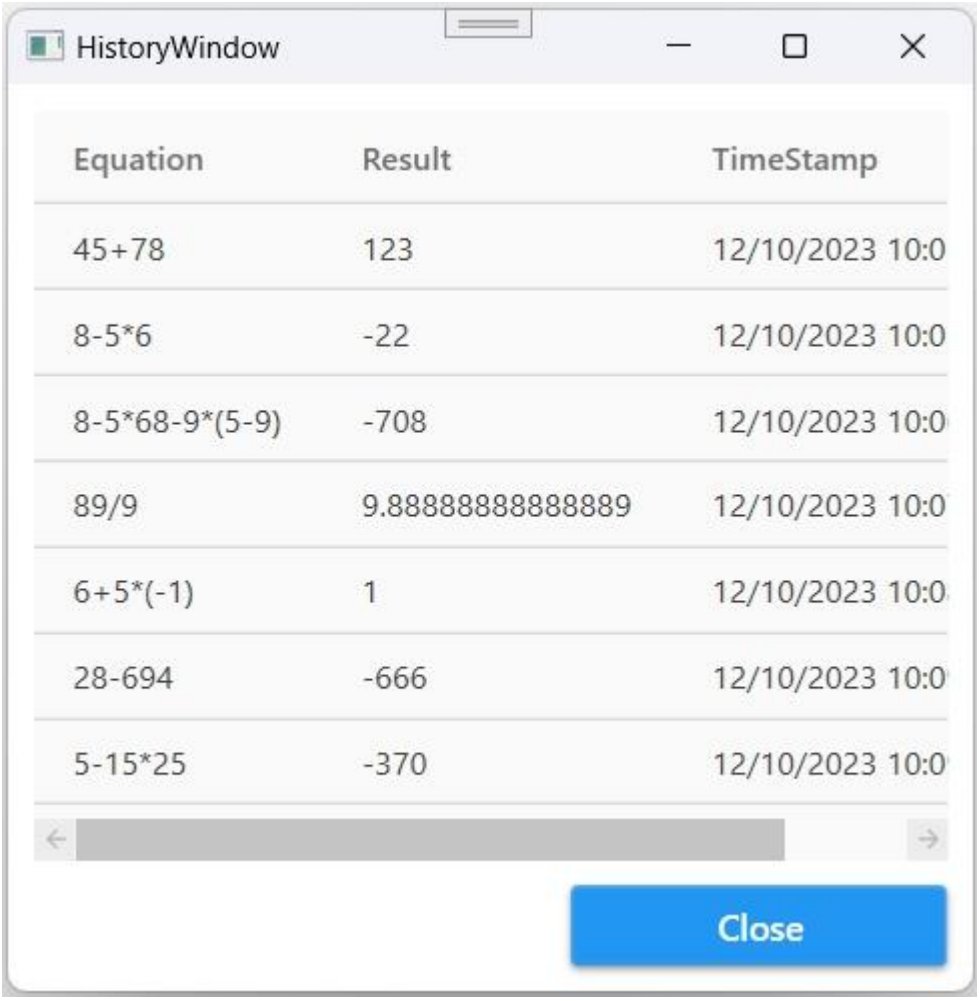


Рис.3.17 - Історія математичних розрахунків, що зберігається у базі даних

```
DROP TABLE IF EXISTS `calc_history`;
CREATE TABLE `calc_history` (
  `Id` int NOT NULL AUTO_INCREMENT,
  `TimeStamp` datetime NOT NULL,
  `Equation` nvarchar(128) NOT NULL,
  `Result` double NOT NULL,
  PRIMARY KEY (`Id`)
)
```

Рис. 3.18 - Створення SQL таблиці для збереження результатів

Мова запитів, основана на SQL дозволяє виконувати додаткові операції та управляти програмним кодом, таким як збережені процедури, тригери та функції.

Використання систем управління базами даних (СУБД) має численні переваги, які сприяють ефективному та безпечному управлінню даними. СУБД дозволяють структурувати дані за допомогою таблиць, відносин та інших об'єктів, що спрощує організацію і пошук інформації. Використання SQL або інших мов запитів забезпечує стандартизований та уніфікований інтерфейс для взаємодії з різними типами даних. СУБД дозволяють встановлювати правила доступу, обмежуючи користувачам або ролям конкретний доступ до даних чи операцій. Багато СУБД підтримують механізми шифрування для захисту конфіденційності даних та різні методи аутентифікації.

СУБД забезпечують механізми транзакцій та журналювання, що гарантує атомарність, консистентність, ізоляцію та довіреність операцій з даними. Встановлення правил та обмежень, які забезпечують цілісність даних та їх відповідність визначеним стандартам.

Використання мови запитів, такої як SQL, дозволяє виконувати складні запити та отримувати необхідні дані. СУБД мають вбудовані оптимізатори, які визначають найбільш ефективні шляхи виконання запитів для покращення продуктивності.

Мова структурованих запитів (SQL) – це мова програмування для зберігання та обробки інформації у реляційній базі даних. Реляційна база даних зберігає інформацію в табличній формі з рядками та стовпцями, що представляють різні атрибути даних та різні зв'язки між значеннями даних. Інструкції SQL можна використовувати для зберігання, оновлення, видалення, пошуку та вилучення інформації з бази даних. Можна також використовувати SQL для підтримки та оптимізації продуктивності бази даних.

Мова структурованих запитів (SQL) – популярна мова запитів, яка часто використовується у всіх типах програм. Аналітики даних та розробники вивчають та використовують SQL, тому що це рішення добре інтегрується з

різними мовами програмування. Наприклад, вони можуть впроваджувати SQL-запити з мовою програмування Java для створення високопродуктивних програм обробки даних з основними системами баз даних SQL, такими як Oracle або MS SQL Server. Рішення SQL також досить простий у освоєнні, оскільки у його твердженнях використовуються загальноприйняті англійські ключові слова.

Системи управління реляційними базами даних використовують мову структурованих запитів (SQL) для зберігання даних та управління ними. У системі зберігається кілька таблиць бази даних, пов'язаних друг з одним. MS SQL Server, MySQL чи MS Access є прикладами систем управління реляційними базами даних. Нижче наведені компоненти такої системи.

Інструкції SQL або SQL-запити є дійсними інструкціями, зрозумілими системам управління реляційними базами даних. Розробники програмного забезпечення створюють інструкції SQL за допомогою різних мовних елементів SQL. Елементи мови SQL – це такі компоненти, як ідентифікатори, змінні та умови пошуку, які формують правильну інструкцію SQL.

РОЗДІЛ 4

ЕКОНОМІЧНА ЧАСТИНА

4.1 Техніко-економічний аналіз передбачуваного проекту

4.1.1 Класифікаційна оцінка проекту

Клас – монопроект, так як це проект з чіткими завданнями.

Тип – організаційно-технічний: забезпечення якості обслуговування користувачів мережі доступу.

Вид – дослідно-інноваційний.

Тривалість – короткостроковий.

Рівень – галузевий.

4.1.3 Життєвий цикл проекту

Поняття життєвого циклу проекту важливе для дослідження й аналізу проблем фінансування пов'язаних з ним робіт і прийняття відповідних управлінських рішень під час його реалізації.

Життєвій цикл проекту (проектний цикл) – це період від народження ідеї до завершення та закриття проекту.

Проектний цикл слід віднести окремі стадії або так звані етапи розвитку, які називаються фазами. Найчастіше виокремлюють саме: доінвестиційну, реалізації та експлуатації.

До інвестиційна фаза включає аналіз умов для втілення проектного задуму; розробку концепції проекту; розробку бізнес-плану та попереднє обґрунтування інвестицій, оцінку життєздатності проекту.

На етапі розробки концепції проекту визначають кінцеві цілі проекту і виявляють можливі шляхи їх досягнення.

Цей етап охоплює в собі чітко сформульовані основні характеристики проекту, до яких в свою чергу відносяться:

– наявність альтернативних технічних и технологічних можливостей; тривалість та попит на продукцію проекту;

– рівень базових і прогнозованих цін на послуги;

– перспективи експорту продукції;

– складність проекту;

– співвідношення витрат на реалізацію проекту і його результатів.

На основі цих та інших показників попередньо аналізують можливості реалізації проекту, нерідко для таких цілей використовується допомогою експертна система. Етап оцінки життєздатності проекту передбачає стисле ТЕО. Далі формують конкретні цілі й обмеження, а також визначають вартість проекту з точністю 25 – 40%.

Результатом такої оцінки життєздатності проекту є обґрунтування переваги обраної альтернативи перед іншими. Після цього інвестор, або замовник має переконатися, що вироблена, в результаті реалізації проекту, продукція, протягом життєвого циклу, матиме стабільний попит, достатній для призначення ціни, яка б забезпечила покриття витрат на експлуатацію й обслуговування об’єктів проекту, швидку окупність капіталовкладень.

Проектний продукт має бути підданий таким видам аналізу: технічному, комерційному, екологічному, організаційному, соціальному.

Розглянемо ті критеріями життєздатності проекту, які класифіковані на початку оцінки проекту.

Технічний аналіз дає змогу виявити техніко-економічні альтернативи; варіанти місцезнаходження об’єкта; масштаб і обсяг проекту; терміни реалізації проекту загалом і за фазами; доступність і достатність сировини та інших необхідних ресурсів.

Організаційний – оцінка організаційних, правових, політичних та адміністративних умов, в яких має реалізуватися й експлуатуватися проект.

Після визначення життєздатності проекту і прийняття рішення про початок його здійснення складають план робіт, тобто структурно визначену

послідовність етапів робіт, які виконують задля досягнення визначеного комплексу цілей (хто й що має робити і в які терміни). На основі плану робіт складають докладний календарний графік робіт або мережного графіку, що дає змогу точніше оцінити вартість проекту.

4.2 Організаційне забезпечення проекту

Таблиця 4.1 – Склад робіт проекту, їх тривалість

Код роботи	Найменування роботи	Т (дні)	Код Попередньої роботи
0-1	Розробка технічного завдання	9	-
1-2	Збір даних	7	0-1
2-3	Аналіз існуючої мережі	7	1-2
3-4	Проведення аналізу положення	9	2-3
4-5	Встановлення потреби в результатах	5	3-4
4-6	Затвердження концепції	4	3-4
5-7	Встановлення ділових контактів, вивчення цілей, мотивів, вимог	5	4-5
6-7	Розвиток концепції, планування наочної області інших елементів проекту	4	4-6
7-8	Розробка і затвердження звідного плану	6	4-7; 6-7

7-9	Організація виконання робіт	10	4-7; 6-7
8-10	Інформаційний контроль за виконанням робіт	6	7-8
9-10	Детальне проектування і технічні специфікації	8	7-9
10-11	Керівництво і координація робіт, коректування основних показників проекту	10	8-10; 9-10
11-12	Підтвердження закінчення робіт	4	10-11
11-13	Експлуатаційні випробування остаточного продукту проекту	6	10-11
12-14	Підготовка кадрів по експлуатації проекту	5	11-12
13-14	Підготовка документів і здача проекту замовникові	9	11-13
14-15	Оцінка результатів проекту і підведення підсумків	4	12-14; 13-14
15-16	Оцінка підсумкових документів і закриття проекту	10	14-15
		∑=110	

4.2.3 Побудова мережного графіка

За даними таблиці 4.1 будується мережевий графік проекту:

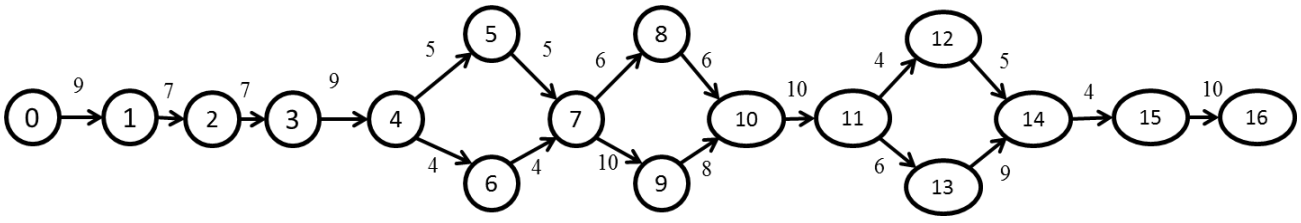


Рис. 4.1 - Мережевий графік проекту

4.2.4 Розрахунок параметрів робіт мережного графіка

Проводиться розрахунок параметрів мережного графіка:

- $t_{р.н.}$ – ранній початок роботи;
- $t_{р.о.}$ – раннє закінчення роботи;
- $t_{п.н.}$ – пізніший початок роботи;
- $t_{п.о.}$ – пізнє закінчення роботи;
- R_c – вільний резерв часу;
- R_i – резерв часу події;
- Lk_p – тривалість критичного шляху.

Розрахунок параметрів мережного графіку зведемо у таблицю 5.2:

Таблиця 4.2 – Параметри робіт мережного графіка

Попередня робота	Робота		Ранній строк			Пізній строк			R_i
	i	**j**	$\mathbf{t_{рн}}$	$\mathbf{t_{ij}}$	$\mathbf{t_{ро}}$	$\mathbf{t_{пн}}$	$\mathbf{t_{по}}$	$\mathbf{R_c}$	
-	0	1	1	9	9	2	10	100	0
0-1	1	2	9	7	15	10	16	94	1
1-2	2	3	15	7	21	16	22	88	1
2-3	3	4	21	9	29	22	30	80	1
3-4	4	5	29	5	33	30	34	76	0
3-4	4	6	33	4	36	34	37	73	1
4-5	5	7	36	5	40	37	41	69	1
4-6	6	7	40	4	43	41	44	66	1
4-7; 6-7	7	8	43	6	48	44	49	61	1
4-7; 6-7	7	9	48	10	57	49	58	52	1
7-8	8	10	57	6	62	57	63	47	1
7-9	9	10	62	8	69	63	70	40	0
8-10; 9-10	10	11	69	10	78	70	79	31	1
10-11	11	2	8	4	83	79	84	26	1
10-11	11	13	81	6	86	82	87	23	0

11-12	12	14	86	5	90	87	91	19	0
11-13	13	14	90	9	98	91	99	11	0
12-14; 13-14	14	15	98	4	101	99	102	8	1
14-15	15	16	101	10	110	102	111	0	1

Визначення критичного шляху: 0-1-2-3-4-6-7-8-10-11-12-14-15-16

4.2.5 Маркетингове обґрунтування проекту

Даний проект є актуальним, особливо в зв'язку з подіями останнього часу, коли збільшується кількість користувачів, а відповідно і навантаження на мережу. Провівши аналіз – зрозуміло, що на необхідність модернізації найбільше впливали наступні фактори. Поява нових користувачів – молодшого віку (школярі молодших класів), які раніше не користувалися інтерактивними послуга під час навчання і старшого віку (вчителі старшого віку), які раніше взагалі не користувалися онлайн послугами, а зараз змушені освоювати комп'ютери та виходити в онлайн для роботи. Необхідність в нових ІКП послугах інтерактивного характеру, саме для них важлива якість (мінімізація помилок, час доступу та ін.) роботи. Все фактори, які зазначені вище збільшили навантаження на мережу.

До чинників, що визначають доцільність впровадження проекту з модернізацією мережі відносяться:

– підвищення оперативності роботи;
– підвищення надійності ходу технологічного процесу;
– зниження до мінімального значення кількості помилок;
– вивільнення персоналу до оптимальної кількості.

4.3 Економічні розрахунки проекту

4.3.1 Визначення трудомісткості розробки ПП

Розрахунок трудомісткості програмного продукту, що розробляється, проводиться за формулою:

$$T_{ПП} = T_{ТЗ} + T_{ТП} + T_{РП} + T_{ВН}, \quad (4.1)$$

де: $T_{ТЗ}$ – трудомісткості розробки технічного завдання створення ПП;

$T_{ТП}$ – трудомісткості розробки технічного проекту ПП;

$T_{РП}$ – трудомісткості розробки робочого проекту ПП;

$T_{ВН}$ – трудомісткості впровадження розробленого ПП;

Розраховуючи трудомісткість, необхідно враховувати, що програмний засіб (ПЗ) по ступеню новизни, що розробляється, є ПЗ, яке має аналог. Крім того, по типу ПЗ даний проект належить до системи автоматизованих розрахунків, а значить його трудомісткість складає 414 людино/г.

Розрахунок трудомісткості розробки технічного проекту розраховується за наступній формулою 5.2.

$$T_{ТЗ} = T_y * L_1 * к_н, \quad (4.2)$$

де T_y – укрупнена форма часу на розробку аналога ПЗ, людино/г, яка корегується поправочним коефіцієнтом $K_к$, враховуючи умови розробки за допомогою комп'ютеру ($K_к = 0.7$).

L_1 – питома вага даного етапу розробки з урахуванням ступеню новизни становить 0.1.

$к_н$ – поправочний коефіцієнт, враховуючи ступінь новизни ($к_н = 0.7$).

$$T_{ТЗ} = (414 * 0.7) * 0.1 * 0.7 = 20.3 \quad (людина - дні).$$

Розрахунок трудомісткості розробки технічного проекту проводиться за формулою 5.3.

$$T_{ТП} = T_y * L_2 * к_н, \quad (4.3)$$

де L_2 – питома вага даного етапу розробки з урахуванням ступеню новизни становить 0.15.

$$T_{ТП} = (414*0.7)*0.15*0.7 = 30,4 \quad (людина-дні).$$

Розрахунок трудомісткості розробки робочого проекту проводиться за формулою 5.4.

$$T_{РП} = T_y * L_3 * к_н * к_T, \tag{4.4}$$

де L_3 – питома вага даного етапу розробки з урахуванням ступеню новизни становить 0.7.

$к_T$ – поправочний коефіцієнт, враховуючи ступінь використання в розробці типових програм ($к_T = 0.8$).

$$T_{РП} = (414*0.7)*0.55*0.7*0.8 = 89,3(людина-дні).$$

Розрахунок трудомісткості впровадження проводиться за формулою 4.5

$$T_{ВН} = T_y * L_4 * к_н, \tag{4.5}$$

де L_4 – питома вага даного етапу розробки з урахуванням ступеню новизни становить 0.2.

$$T_{ВН} = (414*0.7)*0.2*0.7 = 40,6 \quad (людина-дні).$$

В таблиці 5.3 приведено розрахунок трудомісткості розробки проекту

Тривалість розробки ПП визначається за формулою 5.6.

$$T_{ПП} = \sum T_{ij} / (5*0.73*360), \tag{4.6}$$

де $\sum T_{ij}$ – сумарна тривалість розробки, г;

- 6 – тривалість робочого дня, г;
- 0.73 – коефіцієнт переводу у календарні дні;
- T_{ij} – тривалість виконання j-го виду роботи по і-му етапу.

Таблиця 4.3 – Розрахунок трудомісткості проекту

Назва етапу	Розрахунок, дні
1.Технічне завдання	$T_{ТЗ} = 20,3$, $T_{КК} = 0.7*N_{ТЗ} = 0.7*20,3 = 14,21$,

	$T_{НК} = 0.15 * N_{ТЗ} = 0.15 * 20,3 = 3.05$
2.Розробка технічного проекту	$T_{ТП} = 30,4,\ T_{КК} = 0.7 * N_{ТП} = 0.7 * 30,4 = 21,28,$ $T_{НК} = 0.15 * N_{ТП} = 0.15 * 30,4 = 4,56$
3.Розробка робочого проекту	$T_{РП} = 89,3,\ T_{КК} = 0.7 * N_{РП} = 0.7 * 89,3 = 62,51,$ $T_{НК} = 0.15 * N_{РП} = 0.15 * 89,3 = 13.395$
4.Відладка та впровадження	$T_{ВН} = 40,6,\ T_{КК} = 0.7 * N_{ВН} = 0.7 * 40,6 = 28,42,$ $T_{НК} = 0.15 * N_{ВН} = 0.15 * 40,6 = 6,09$
Усього:	$\sum T_{ij} = 20,3 + 14,21 + 3,05 + 30,4 + 21,28 + 4,56 +$ $89,3 + 62,51 + 13.395 + 40,6 + 28,42 + 6,09 = 334,115$

$T_{ППП} = 334,115/(5 * 0.73 * 360) = 334,115/1314 = 0,2542(років)$.

$T_{ППП} = 0,2542 * 360 = 91,512(днів)$

4.3.2 Визначення ціни проекту

Для визначення ціни необхідно розрахувати основну заробітну плату працівників, матеріальні витрати, вартість машино-години та інші витрати на розробку продукту. Розрахунок проводиться за формулою 5.7.

$$Ц = К * С + П_р, \quad (4.7)$$

– $К$ – коефіцієнт обліку витрат на виготовлення дослідного зразка ПП як продукції виробниче технічного призначення ($К = 1.1$).

– $С$ – витрати на розробку продукції (кошторисна собівартість).

– $П_р$ – нормативний прибуток.

Розрахунок нормативного прибутку проводиться за формулою 5.8.

$$П_р = (С - С_М) * Р_н / 100, \quad (4.8)$$

– $С_М$ – матеріальні витрати, грн.

– $Р_н$ – норматив рентабельності приймається рівним 25%.

$П_р = (47474 - 3550) * 0,25 = 1098,1$

Розрахунок матеріальних витрат приведено в таблицях 5.4

Таблиця 4.4 – Розрахунок витрат на відрядження

Найменування	Кількість	Ціна за	Сума,

матеріалу		одиницю, грн	грн
Канцтовари			550
Папір	15	60	900
Виготовлення ксерокопій креслень			950
Виготовлення рекламних листівок	100	2	200
Дискети, *CD*-диски, *Flash*-накопичувачі			950
		Всього	3550

$$Ц = 1.1 \cdot 47474 + 1098{,}1 = 53319{,},5$$

Основна заробітна плата враховує основну заробітну плату виконавця, безпосередньо зайнятого розробкою даного ПП, з урахуванням його посадового окладу та часу участі в розробці. Розрахунок виконується за формулою 4.9.

$$C_{ЗО} = \sum (З_i * К_0 / Др) * \tau_i , \qquad (4.9)$$

Де:

– $З_i$ – середньомісячний оклад і-того виконавця, грн.

– $К_0$ – коефіцієнт обліку окладу керівників і консультантів проекту приймається рівним 0.1.

– Др – середня кількість робочих днів в місяці приймається рівною 21 дню.

– τ_i – трудомісткість робіт, що виконуються і-тим виконавцем, людино/дні.

У розробці задіяні постановник задачі та розробник проекту (мережі), середньомісячний оклад яких складає 10800, 5000 грн. відповідно.

Трудомісткості робіт складають 91,512 чол-дні.

$$C_{ЗО} = \frac{10800 * 0.1 * 40}{21} + \frac{5000 * 91{,}512}{21} = 2057{,}1 + 21788{,}6 = 23845{,}7 .$$

Розрахунок додаткової заробітної платні враховує всі виплати безпосереднім виконавцям за час непропрацьований на виробництві, у тому числі: оплата чергових відпусток, компенсації за недовикористану відпустку, оплати пільгового годинника підліткам та ін. і проводиться за формулою (4.10).

$$C_{ЗД} = C_{ЗО} * K_{Д}, \quad (4.10)$$

де $K_{Д}$ – коефіцієнт відрахувань на заробітну платню приймається рівним 0.1.

$C_{ЗД} = 23845,7 * 0.1 = 2385,57(грн.)$.

Розрахунок відрахувань на соціальне страхування враховує відрахування до бюджету соціального страхування за встановленим державою тарифом від суми основної і додаткової заробітної платні і розраховується за формулою (5.11).

$$C_{СС} = K_{СС} * (C_{ЗО} + C_{ЗД}), \quad (4.11)$$

Де: $K_{СС}$ – коефіцієнт відрахувань на соціальне страхування приймається рівним 0,22, він є єдиним соціальним неском.

$C_{СС} = 0,22 * (23845,7 + 2385,57) = 5770,87(грн.)$

Загальновиробничі витрати розраховуються за формулою 4.12 і враховують витрати на загальногосподарські витрати, позавиробничі витрати і витрати на управління.

$$C_{Н} = K_{Н} * C_{ЗО}, \quad (4.12)$$

де $K_{Н}$ – коефіцієнт накладних витрат приймається рівним 0.5.

$C_{Н} = 0.5 * 23845,7 = 11922,85(грн.)$.

В таблиці 4.5 представлена кошторисна собівартість ПП.

Таблиця 4.5 – Сметна вартість ПП

Назва категорії витрат	Кошторисна собівартість, грн.
Основна заробітна плата	23845,7

Додаткова заробітна плата	2384,57
Відрахування на соціальне страхування	5770,87
Матеріальні витрати	3550
Загальновиробничі витрати	11922,85
Сумарна собівартість	47474

4.3.3 Визначення капітальних і поточних витрат

Розрахунок капітальних витрат, пов'язаних з впровадженням (вдосконаленням) ІС здійснюється за формулою 5.13.

$$K_2 = K_П + K_{КО} + K_{ДО} + K_Б, \quad (4.13)$$

– $K_П$ – передвиробничі витрати;

– $K_{КО}$ – вартість комп'ютерного устаткування(

– $K_{ДО}$ – вартість допоміжного устаткування, необхідного для надійної роботи мережі доступу;

– $K_Б$ – вартість будівництва (реконструкції) у зв'язку з впровадженням мережі доступу.

Передвиробничі витрати – всі витрати, пов'язані з проектуванням, розробкою, відладкою та впровадженням програмного забезпечення, навчання обслуговуючого систему персоналу, перепідготовка частини персоналу підприємства та інші передвиробничі витрати розраховуються за формулою 4.14.

$$K_П = Ц * 1.0. \quad (4.14)$$

Таким чином, $K_П$ приймаються у розмірі 100% від вартості розробленого ПП і становлять:

$K_П = 53319,5 * 1.0 = 53319,5(грн.)$.

Спеціальне комп'ютерного устаткування вартістю 121000 грн, додаткове 8900 грн. Будівництво, пов'язане з впровадженням проекту приблизно 15000 грн. З цього виходить, що величина капітальних витрат складає:

$K_2 = 53319,5 + 121000 + 8900 + 15000 = 198219,5(грн.)$.

Розрахунок поточних витрат, пов'язаних з впровадженням інформаційної системи здійснюється за формулою 5.15. Основними користувачами програми є адміністратор та інженер самого проекту, в разі модернізації. Тобто при модернізації скорочено одного робітника.

$$C = C_{ОПЛ} + C_A + C_{ЕЛ} + C_P + C_{ДОП} + C_П, \qquad (4.15)$$

Де:

– $C_{ОПЛ}$ – річний фонд основної і додаткової оплати праці персоналу, обслуговуючого мережі;

– C_A – сума річних амортизаційних відрахувань від вартості основного і допоміжного устаткування мережі;

– $C_{ЕЛ}$ – вартість витрат на електроенергію за рік;

– C_P – вартість річного ремонту основного і допоміжного устаткування;

– $C_{ДОП}$ – річна вартість допоміжних матеріалів, пов'язаних з експлуатацією мережі;

– $C_П$ – вартість річного утримання приміщень.

Модернізована мережа дозволить скоротити робітника, залишиться 2 робітники, з оплатою 45 грн./год кожен. Тобто виходить економія за рік 82125 грн на одному робітнику, який був скорочений.

Річний фонд основної заробітної платні персоналу, обслуговуючого мережу розраховується за формулою 5.16.

$$З_{ОПЛ} = \sum Ч_{сі} * З_{сі} + \sum Ч_{рj} * t_{сj} * Ф_{рj}, \qquad (4.16)$$

– $Ч_{сі}, Ч_{рj}$ – чисельність, відповідно, фахівців і-ї категорії та j-того розряду, обслуговуючих ІС;

– $t_{сj}$ – годинна тарифна ставка робочого j-того розряду;

– $Ф_{рj}$ – річний фонд робочого часу j-того розряду приймається рівним 1825 годин.

До модернізації:

$$З_{ОПЛ} = 3 \cdot 45 \cdot 1825 = 246375(грн.)$$

Після модернізації:

$$З_{ОПЛ} = 2 \cdot 45 \cdot 1825 = 164250(грн.)$$

Фонд додаткової заробітної платні розраховується за формулою 4.17.

$$З_{ДОП} = З_{ОСН} * К_{ДОП}, \quad (4.17)$$

де $К_{ДОП}$ – коефіцієнт додаткової заробітної платні, приймається рівним 0.1.

До модернізації:

$$З_{ДОП} = 246375 * 0.1 = 24637,5(грн.)$$

Після модернізації:

$$З_{ДОП} = 164250 * 0.1 = 16425(грн.).$$

Розрахунок нарахувань на заробітну платню проводитися за формулою 4.18.

$$З_{НАР} = (З_{ОСН} + З_{ДОП}) * К_{В}, \quad (4.18)$$

де $К_{В}$ – коефіцієнт відрахувань на соціальні потреби, приймається рівним 0.22.

До модернізації:

$$З_{НАР} = (246375 + 24637,5) * 0.22 = 59842,75(грн.)$$

Після модернізації:

$$З_{НАР} = (164250 + 16425) * 0.22 = 39708,9(грн.).$$

Таким чином, загальні витрати на оплату праці розраховуються за формулою 5.19.

$$С_{ОПЛ} = З_{ОСН} + З_{ДОП} + З_{НАР}, \quad (4.19)$$

До модернізації:

$$C_{ОПЛ} = 246375 + 24637,5 + 59842,75 = 330855,25(грн.)$$

Після модернізації

$$C_{ОПЛ} = 164250 + 16425 + 39708,9 = 220383,9(грн.).$$

Розрахунок амортизаційних відрахувань розраховується за формулою 4.20.

$$C_a = K_{КО} * H_a / 100, \quad (4.20)$$

Де H_a – норма амортизаційних відрахувань, приймається рівної 60% для комп'ютерного устаткування.

$$C_a = 121000 * 60 / 100 = 72600(грн.).$$

Річна вартість споживаної електроенергії визначається за формулою 4.21.

$$C_{ЕЛ} = M_y * T_{КО} * Ц_О * K_И, \quad (4.21)$$

– M_y – встановлена сумарна потужність комп'ютерного устаткування, приймається рівною 5.2 кВт, хоча раніше була 8.1 кВт;

– $T_{КО}$ – річний фонд роботи ЕОМ з урахуванням часу на профілактичні огляди складає 220*6=1320 годин, хча раніше було 245*6=1470;

– $Ц_О$ – вартість 1 кВт електроенергії на даний момент складає 3,05грн.;

– $K_И$ – коефіцієнт інтенсивного використовування потужності, приймається рівним 0.9.

Після модернізації

$$C_{ЕЛ} = 8,1 * 1470 * 3,05 * 0.9 = 32684,72(грн.)$$

Після модернізації

$$C_{ЕЛ} = 5,2 * 1320 * 3,05 * 0.9 = 18841,68(грн.).$$

Вартість річного ремонту основного і допоміжного устаткування становить 6 % від вартість комп'ютерного і допоміжного устаткування і складає:

$$C_P = K_{КО} * 0.06 = 121000 * 0.06 = 7260(грн.).$$

Річна вартість допоміжних матеріалів, пов'язаних з експлуатацією ІС становить 1.5 % від вартості комп'ютерного і допоміжного устаткування і складає:

$$C_{ВСП} = K_{КО} * 0.015 = 121000 * 0.015 = 181{,}5(грн.).$$

Витрати на виробниче приміщення складають 1950 грн. в рік за 1 м^{2,} раніше було 19 м^2:

$$C_{П} = 19 * 1950 = 37050(грн.)$$

а зараз 12 м^2 и складає:

$$C_{П} = 12 * 1950 = 23400(грн.).$$

До впровадження проекту витрати складали:

$$C_1 = 330855{,}25 + 72600 + 32684 + 7260 + 181{,}5 + 37050 = 480630{,}75(грн.)$$

Таким чином, поточні витрати після впровадження даного проекту складають:

$$C_2 = 220383{,}9 + 72600 + 18841{,}68 + 7260 + 181{,}5 + 23400 = 321607{,}08(грн.)$$

4.3.4 Визначення показників економічної ефективності проекту

Очікуваний економічний ефект розраховується за формулою 4.22.

$$E_o = E_г - E_н * K_п, \qquad (4.22)$$

- $E_г$ – річна економія на поточних витратах;
- $E_н$ – нормативний коефіцієнт ефективності єдиноразових витрат – 0.25;
- $K_п$ – єдиноразові витрати на проект.

Річна економія складається з поточних витрат та приросту прибутку у зв’язку із впровадженням проекту, та обчислюється за формулою 4.23.

$$E_г = (C_1 - C_2) + \Delta П, \qquad (4.23)$$

- $C_{1,}\, C_2$ – відповідно поточні витрати, відповідно до та після впровадження проекту (грн);

– $\Delta П$ – приріст прибутку господарюючого суб'єкта або його структурного підрозділу при впровадженні проекту (грн) в даному випадку неможливо підрахувати приріст прибутку, тому він не буде рахуватися.

$$E_г = (480630{,}75 - 321607{,}08) = 159023{,}67(грн.)$$

Тобто було зекономлено 159023,67 грн.

$$E_о = 159023{,}67 - 0{,}25 * 321607{,}08 = 80401{,}77$$

Розраховуємо коефіцієнт ефективності єдиноразових витрат за формулою 4.24

$$E = E_г / K_п, \qquad (4.24)$$

$$E = E_г / K_п = 159023{,}67 / 198219{,}5 = 0{,}802$$

$E > E_н$, тому проект економічно ефективний.

Термін окупності одноразових витрат обчислюється за формулою 4.25

$$T = 1/E\ , \qquad (4.25)$$

$$T = 1/0{,}802 = 1{,}25^{\rightarrow}\ (років)$$

$$T = 1{,}25 \cdot 365 == 465(дні)$$

Техніко-економічні показники проекту, , відображені в таблиці 5.6

Таблиця 4.6 – Техніко-економічні показники проекту

/п	Найменування показника	Одиниця вимірювання	Значення показника після впровадження проекту
	Час розробки проекту	дні	91,512
	Ціна ПП	грн.	53319,5
	Капітальні витрати	грн.	198219,5
	Поточні витрати	грн/рік	321607,08
	Економічний ефект від реалізації проекту	грн/рік	159023,67
	Термін окупності	Рік	1,25

Висновки:

Усі отримані показники знаходяться в рамках норми. Впровадження проекту є економічно ефективним. Після впровадження проекту значно зменшились час виконання даної роботи, а також підвищилась ефективність за рахунок оновлення обладнання та модернізації мережі взагалі. Термін окупності складає приблизно 465 дні. Що також каже про великі можливості проекту та можливості його подальшого розвитку. Завдяки модернізації за рік експлуатації мережі буде зекономлено 159023 грн.

РОЗДІЛ 5

ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ТЕХНІКА БЕЗПЕКИ, ЕКОЛОГІЯ

Тема даного дипломної роботи є «Модернізація мережі доступу у селищі Холодна Балка із аналізом топологічної структури». У зв'язку із цим у розділі охорони праці будемо розглядати офісне приміщення, в якому буде проводитись аналіз, розрахунки та моделювання, з наступними характеристиками: 12х9х3,5м.

$$12 \cdot 9 = 108(м^2)$$

$$12 \cdot 9 \cdot 3{,}5 = 378(м^3)$$

Виходить: 108 м^2; 378 м$^{3.}$ За нормативними даними на одну людину має бути не менш 20 м^3 та не менш 6 м^2. Тому максимальна кількість людей у цьому приміщені 18.

5.1 Шкідливі та небезпечні фактори в роботі у робочій зоні

До небезпечних і шкідливих виробничих факторів у зоні робочих місць експлуатуючі системи з комп'ютерами відносять: статична електрика, електромагнітні випромінювання, шум, вібрації, недостатнє висвітлення, вентиляція та ін. Персонал піддається також і впливу шкідливих психофізіологічних виробничих факторів, які у свою чергу підрозділяються по характері дії на фізичні перевантаження й нервово-психічні перевантаження. Фізична й нервово-психічні: розумова перенапруга; монотонність праці; емоційні перевантаження.

Професійні шкідливості – несприятливі для здоров'я фактори трудового (виробничого) процесу або незадовільні санітарно-гігієнічні умови.

Професійно шкідливий чинник – виробничий фактор, дія якого в певних умовах може мати несприятливий вплив на працездатність і стан здоров'я працівника аж до виникнення професійного захворювання.

Професійне захворювання – хвороба, викликана дією специфічного шкідливого фактора в умовах виробництва, яка підтверджена в установленому порядку.

Специфічний чинник виробничого середовища – фактор виробничого середовища, який не може бути виключений з виробничого середовища без зупинки процесу виробництва.

Залежно від часу і інтенсивності впливу на працівника, виробничі фактори можуть бути небезпечними або шкідливими.

При миттєвій дії фактор стає небезпечним, а при тривалому впливі — шкідливим.

Небезпечним називають виробничий фактор, вплив якого на організм працюючого у відповідних умовах праці може призвести до травм або іншого раптового, різкого погіршення стану здоров'я.

Шкідливим називають виробничий фактор, вплив якого на організм працюючого може призводити в певних умовах до захворювання або зниження рівня працездатності.

Однією із основних цілей охорони праці на підприємстві є оцінка обстановки та характеристик трудового процесу в частині його впливу на здоров'я і життя працівника.

Для досягнення цього завдання державою встановлено низку критеріїв оцінки, які допомагають визначити ступінь небезпечності умов праці на підприємствах, що використовують працю найманих робітників.

Класифікація виробництва по ступені пожежної, вибухової і вибухопожежної небезпеки

По ступені вибуховий, вибухопожежної і пожежної небезпеки приміщення із ВДТ ЕОМ ставляться до категорії Д, у яких використовуються негорючі речовини й матеріали в холодному стані.

5.2 Методи зниження шкідливих та небезпечних факторів в роботі

Вимоги електробезпечності при експлуатації ВТД ЕОМ

Лінія електромережі для ЕОМ виконується як окрема групова трьохпровідна мережа, шляхом прокладки фазового, нульового робочого, нульового захисного проводів.

Нульовий захисний провід використається тільки для занулення электроприємника.

Використання нульового робочого проводу в якості нульового захисного забороняється.

Штепсельні з'єднання й розетки повинні мати спеціальні контакти для підключення нульового захисного проводу.

Не допускається підключення ПЕОМ до звичайного двохпровідної мережі, у тому числі з використанням перехідних пристосувань. Є неприпустимим експлуатація кабелів і проводів з ушкодженою ізоляцією, саморобних подовжувачів.

Захисне заземлення - навмисне з'єднання металевих неструмоведучих частин, які можуть виявитися під напругою зі штучним заземленням. Застосовується в мережах з напругою до 1000 В з ізольованої нейтраллю, і вище 1000 У с будь-яким режимом нейтралі.

За способом захисту людини від поразки електричним струмом ЕОМ повинні відповідати першому класу захисту.

Розрахунок заземлюючого пристрою

Вихідні данні:

– d - зовнішній діаметр труби d = 0.04м;

– Довжина вертикальних заземлювачей l = 2,2 м; Відношення l/l’ = 2;

– $\rho_ф$ – питомий опір ґрунту суглинок в місці пристрою заземлення $\rho_ф$ = 100 Ом·м;

– φ - безрозмірний кліматичний коефіцієнт, що враховує сезонні коливання вологості ґрунту 1,3.

Розрахункове значення питомого опору ґрунту обчислюємо по формулі:

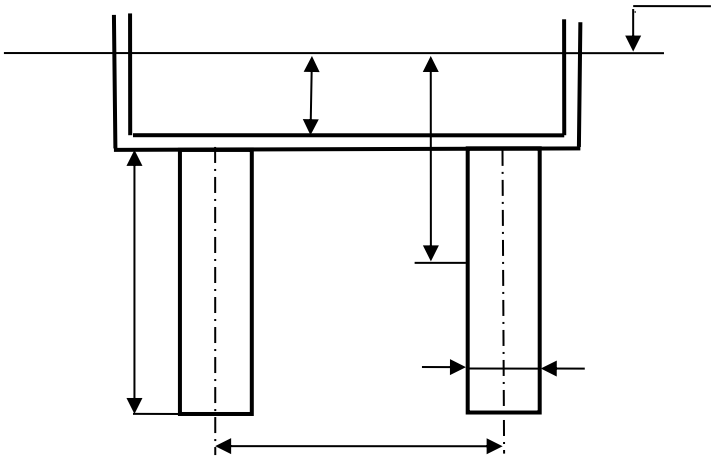
$$\rho_р = \rho_ф \cdot \varphi \qquad (5.1)$$

Де:

– $\rho_ф$ – питомий опір ґрунту в місці пристрою заземлення ;

– φ - безрозмірний кліматичний коефіцієнт, що враховує сезонні коливання вологості ґрунту

$$\rho_р = 100 \cdot 1,3 = 130 \text{ Ом·м}$$



Рисю 5.1 – Устройство заземлителей

$t_о$ = 0,5 м, l = 2,2 м

l - довжина заземлювачей, м;

l' - відстань між заземлювачами, м

Тоді $t = t_о + l/2 = 0,5 + 2,2/2 = 1,6$ м

Опір одного вертикального заземлювача обчислюємо по формулі:

$$\mathbf{R_{тр} = \frac{\rho_{расч}}{2\pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + 0,5\ln\frac{4t + l}{4t - l}\right),} \qquad (5.2)$$

$$R_0 = \frac{130}{2 \cdot 3,14 \cdot 2,2} \cdot \left(\ln\frac{2 \cdot 2,2}{0.04} + 0.5 \cdot \ln\frac{4 \cdot 1.6 + 2,2}{4 \cdot 1.6 - 2,2}\right) = 47,6 Ом$$

Визначаємо необхідна кількість вертикальних заземлювачей:

$$n = \frac{R_0}{R_{тр}} = \frac{47,6}{4} = 11,9 \qquad (5.3)$$

Де: $R_{тр} = 4$ Ом – необхідний опір.

Округляємо n до найближчого стандартного числа n‘= 20 шт. Визначаємо загальний опір системи вертикальних заземлювачей:

$$R = \frac{R_0}{n' \cdot \eta_в} \qquad (5.4)$$

Де: $\eta_в$ – коефіцієнт використання вертикальних заземлювачей, тому що заземлювачі розташовані в ряд і відношення відстані між заземлювачами до їхньої довжини дорівнює 2, то за табличним значенням приймаємо $\eta_в = 0,67$ (Юдин «Охрана труда в машиностроении» с.270 таб.19)

$$R_{об} = \frac{47,6}{20 \cdot 0,67} = 3,55 Ом$$

Визначаємо опір сполучної смуги. При розміщенні в ряд довжина смуги

$$L = l' \cdot (n‘-1) = 4,4 \cdot (20-1) = 83,6 \text{ м} \qquad (5.5)$$

$$\mathbf{R_{пол} = \frac{\rho_р}{2\pi \cdot L \cdot \eta_г} \cdot \ln \frac{L^2}{d \cdot t_о}}, \qquad (5.6)$$

Де: $\eta_г$ – коефіцієнт використання горизонтальних заземлювачей, приймаємо $\eta_г = 0,56$ (Юдін «Охороона праці в машинобудуванні» с.270 таб.20);

$$R_{пол} = \frac{100}{2 \cdot 3,14 \cdot 83,6 \cdot 0,56} \cdot \ln \frac{83,6^2}{0,04 \cdot 0,5} = 4,34 Ом$$

Загальний опір системи обчислюється по формулі:

$$R = \frac{3,55 \cdot 4,34}{3,55 + 4,34} = 1,95 Ом$$

$$R_c = \frac{R_{об} \cdot R_{пол}}{R_{об} + R_{пол}} (Ом) \qquad (5.7)$$

Прийнята система заземлення задовольняє технічним вимогам, тому що загальний опір системи заземлення менше припустимого значення $R_{тр} \leq 4$ Ом

Пожежна профілактика

Пожежі становлять більшу небезпеку для працюючих і можуть заподіяти величезний матеріальний збиток. Пожежна безпека може бути забезпечена мірами пожежної профілактики й активного пожежного захисту. Поняття пожежної профілактики включає комплекс заходів, необхідних для попередження виникнення пожежі або зменшення його наслідків. Під активним пожежним захистом розуміються міри, що забезпечують успішну боротьбу з виникаючими пожежами або вибухонебезпечною ситуацією.

Причини пожеж в електроустановках

Пожежі в електроустановках відбуваються через:

– короткого замикання;

– перевантаження мереж;

– більших перехідних опорів;

– від електронагрівальних приладів.

Перевантаження мереж відбувається в результаті проходження по них напруги, що перевищує номінальний. Таке може відбутися у випадку підключення великої кількості споживачів. Це приведе до руйнування, плавлення й згоряння ізоляції, що спричиняє коротке замикання.

Коротке замикання відбувається в тому випадку, коли крапки різних фаз мережі з'єднуються через малий опір. Внаслідок чого миттєво збільшується струм, відбувається виділення великої кількості тепла.

Міри захисту:

– дотримання нормальних режимів експлуатації;

– своєчасне проведення регламентних робіт;

– застосування плавких запобіжників і автоматів.

Більшу роль у пожежонебезпеці грає правильний вибір використання електроустаткування.

Система висвітлення з лампами накалювання найнебезпечніші через те, що температура поверхні колби лампи приблизно дорівнює 500 ^{0}С. Більшу роль у безпеці грає правильний вибір типу світильника.

Засоби пожежогасіння

Як засоби пожежогасіння на даному об'єкті застосовуються вуглекислотні вогнегасники, призначені для гасіння загорянь установок напругою до 1000 В: ручні ВВ-2А, ВВ-5, ВВ-8.

У вуглекислотних вогнегасниках застосовують зріджений двоокис вуглецю. Вогнегасяща дія його полягає в розведенні повітря й зниженні в ньому змісту кисню до концентрації, при якій припиняється горіння. Вогнегасящий ефект обумовлюється втратами теплоти на нагрівання двоокису вуглецю й зниженням теплового ефекту реакції.

Розрахунок вуглекислотної установки для гасіння пожежі в приміщенні

Кількість вогнегасящого газового складу $G_г$ (кг) визначається за залежністю

$$G_г=1,25\bullet (G_в\bullet V_п\bullet K_у), \text{ кг} \quad (5.8)$$

Де:

– $G_в$- вогнегасяща концентрація газового складу для вуглекислоти;

– ($G_в$= 0,7 кг/м^3);

– $V_п$ – обсяг приміщення, що захищається, м^3;

– $К_у$ – коефіцієнт, що враховує особливості процесу газообміну, джерела вуглекислоти крізь нещільності й щілині приміщення, що захищається, приймається 1...1,2

$$G_г=1,25\bullet (0,7\bullet 379\bullet 1) =331,625 \text{ кг}$$

Потрібна кількість робочих балонів Nб (од.) з вуглекислотою визначається як

$$N_б = \frac{Gг}{V_б \cdot \rho \cdot \alpha_н}, од. \quad (5.9)$$

Де:

– $V_б$ =40 л – обсяг балону;

– ρ= 0,625 кг/л – щільність засобу гасіння;

– α_v=*1* – коефіцієнт наповнення.

$$N_б = \frac{331{,}625}{40 \cdot 0{,}625 \cdot 1} \approx 13, \text{од.}$$

Кількість резервних балонів варто прийняти такий же як число робочих балонів. $N_б$ = $N_р$ = 13 балонів. Загальна кількість балонів 26 шт.

Виробнича санітарія – це система організаційних та технічних заходів, які спрямовані на усунення потенційно небезпечних факторів і запобігання професійних захворювань та отруєнь.

До організаційних заходів належать:

– дотримання вимог охорони праці жінок та осіб віком до 18 років;

– проведення попередніх та періодичних медичних оглядів осіб, які працюють у шкідливих умовах;

– забезпечення працюючих у шкідливих умовах лікувально-профілактичним обслуговуванням тощо.

Технічні заходи передбачають:

– систематичне підтримання чистоти у приміщеннях і на робочих місцях;

– розробку та конструювання обладнання, що вилучає виділення пилу, газів та пари, інших шкідливих речовин у виробничих приміщеннях;

– забезпечення санітарно-гігієнічних вимог до повітря виробничого середовища;

– улаштування систем вентиляції та кондиціювання робочих місць зі шкідливими умовами праці;

– забезпечення захисту працюючих від шуму, ультра- та інфразвуку, вібрації, різних видів випромінювання.

Таким чином, запобігання професійних захворювань і отруєнь здійснюється через здійснення комплексу організаційних і технічних заходів, які спрямовані на оздоровлення повітряного середовища, виконання вимог гігієни та особистої безпеки працюючих.

Розрахунок вентиляції приміщення

Розрахувати продуктивність системи вентиляції в приміщенні із заданими параметрами, з урахуванням максимального розташування робочих місць із персональними комп'ютерами.

Визначаємо площу приміщення:

$$S = A \times B = 12 \times 9 = 108 \text{ м}^2 \quad (5.10)$$

З обліком, що кількість людей ($n_л$) і кількість ЕОМ ($n_к$) дорівнює 23, визначимо кількість тепла виділюване людьми:

$$Q_л = n_л \times q_л; \quad (5.11)$$

$q_л$ – питоме тепловиділення 1 чоловік (170 Вт);

$$Q_л = 18 \times 170 = 3060 \text{ Вт};$$

Визначимо кількість тепла виділюване від устаткування:

$$Q_{об} = n_к \times q_к; \quad (5.12)$$

$q_к$ – питоме тепловиділення 1 комп'ютера (400 Вт);

$$Q_{об} = 18 \times 400 = 7200 \text{ Вт};$$

Визначимо кількість тепла, що виділяє освітлення:

$$Q_{осв} = S_{пр} \cdot q_{осв} ; \quad (5.13)$$

Де:

– $S_{пр}$ – площа приміщення,

– $q_{осв}$ – кількість тепла, що приходить на 1 м^2 площини приміщення від штучного освітлення.

$$Q_{осв} = 108 \times 10 = 1080 \text{ Вт};$$

Визначимо кількість тепла, що надійшло через конструкції, що обгороджують:

$$Q_{осв} = V_{пр} \cdot q_{пр} \quad (5.14)$$

Де:

– $V_{п\$р}$ – об’єм приміщення,

– $q_{осв}$ – кількість тепла, що приходить на 1 м^3 обєму приміщення від навколишнього середовища.

$$Q_{огр.кон} = 108*3,5*18 = 6804 \text{ Вт};$$

Уважаємо загальне:

$$Q = Q_л + Q_{об} + Q_{осв} + Q_{огр.кін};$$

$$Q = 3060+7200+1080+6804 = 18144 \text{ Вт}$$

Об'ємна витрата повітря в приміщенні:

$$L = \frac{Q}{C_p * \rho(t_{выт} - t_{пр})}, \text{ (м3/ч);} \quad (5.15)$$

- ρ - щільність повітря (1,2 кг/м^3);
- c - питома теплоємність повітря;
- $t_{выт}$, $t_{пр}$ – температура витяжного й припливного повітря.

$$L = \frac{18,14}{1.0 \cdot 1.2 \cdot (24-16)} = 1,89 \frac{м^3}{с} = 6804 \frac{м^3}{ч}$$

Визначимо потужність, споживану вентилятором, по формулі

$$N = \frac{L \Delta P_1 k * 10^{-6}}{\eta_в \eta_п * 3.6}, \text{ [Вт]} \quad (5.16)$$

Де:

- L - об'ємна витрата повітря, [м3/з];
- ΔР - втрата тиску у повітроводі,(300) [Па];
- $\eta_в$ - КПД вентилятора (0,6);
- $\eta_п$ - КПД приводу, при приєднанні колеса через муфту (0,9);
- k - коефіцієнт запасу (1,2);

$$N = \frac{1.2 \cdot 6804 \cdot 300 \cdot 10^{-6}}{3.6 \cdot 0.6 \cdot 0.9} = 1,26 \text{ кВт.}$$

Розрахунок освітлення приміщення

Завданням розрахунку штучного освітлення є визначення потрібної потужності електричної освітлювальної установки для створення у виробничому приміщенні з персональними комп'ютерами заданої освітленості.

Підбираємо систему висвітлення для приміщення:

Довжина приміщення, А = 12 м

Ширина приміщення, В = 9 м

Висота приміщення, $H_п$ = 3,5 м

Ен - нормована освітленість (Ен = 350)

k - коефіцієнт запасу (k = 1,6)

Визначимо площу приміщення:

$$S = A \times B = 12 \times 9 = 108 \text{ м}^2$$

Обчислимо висоту підвісу світильника над робочою поверхнею по формулі

$$Hp = H_п - h_{р.п.}, \qquad (5.17)$$

Де:

– $h_{р.п.}$– висота робочої поверхні;

– $H_р$ = 3,5 – 1 = 2,5м.

Для досягнення рівномірної освітленості необхідно, щоб відношення відстаней між центрами світильників до висоти їхнього підвісу над робочою поверхнею рівнялася конкретному числу, характерному для типу обраного світильника.

У приміщеннях із ВДТ ЕОМ обрані світильники серії ЛД. Для світильників ЛПО відношення $L_k/H_р$ = 1,4

Визначимо L_k – відношення між центрами світильників розраховуємо по формулі

– $L_k = H_р \cdot 1,4$

– $L_k = 2,5 \cdot 1,4 = 3,5$ м

Визначимо кількість світильників:

$$N=A \cdot B/L_k^2 = 108/3,5^2 = 8,82 \qquad (5.18)$$

Для визначення коефіцієнта використання світлового потоку необхідно обчислити індекс приміщення по формулі

$$i = \frac{a \cdot b}{H_{раб} \cdot (a+b)} \qquad (5.19)$$

$$i = \frac{12 \cdot 9}{2.5 \cdot (12+9)} = 2{,}05$$

При i = 2 коефіцієнт використання світлового потоку η=57.

Світловий потік одного світильника визначається формулою

$$\Phi_C = \frac{E_H \cdot k \cdot S \cdot z \cdot 100}{N \cdot \eta}, \qquad (5.20)$$

Де:

- Ен - нормована освітленість (Ен = 350);
- k - коефіцієнт запасу (k = 1,6);
- S - площа освітлюваного приміщення;
- z - коефіцієнт нерівномірності висвітлення
- (для люмінесцентних ламп z = 1,1);
- N – число світильників (округлімо до 12 шт).

$$\Phi_C = \frac{350 \cdot 1{,}6 \cdot 108 \cdot 1{,}1 \cdot 100}{12 \cdot 57} = 9726\ Лм$$

Приймаємо до установки лампи ЛБ40 у кількості 3-х штук зі світловим потоком $\Phi = 3120\ \ лм$ у кожному світильнику.

$$\Delta = \frac{9360 - 9726}{9360} * 100\% = -3{,}91\%$$

На практиці допускається відхилення світлового потоку реального від розрахункового в межах від -10% до +20%

Наша погрішність становить -3,91%. Підрахована погрішність задовольняє умові.

Розрахуємо потужність освітлювальної системи по формулі

$$Pc = N \cdot n \cdot P_1, \qquad (5.21)$$

Де:

- N - кількість світильників;
- n - мінімальна кількість світильників;
- P_1 – мінімальна потужність.

$$Pc = 12 \cdot 3 \cdot 40 = 1440 \text{ Вт}$$

Загальні положення гігієнічних вимог до організації роботи із ВДТ ЕОМ

Облаштованість робочих місць ВДТ ЕОМ повинне забезпечити :

– належні умови висвітлення й відсутність відблисків;

– оптимальні параметри мікроклімату;

– належні ергономічні характеристики: наявність шуму й вібрації, рентгенівські випромінювання, електромагнітні випромінювання, електростатичне поле, наявність пилу, озону й оксиду азоту.

Вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ

– Розміщення робочих місць із ВДТ ЕОМ у підвальних і цокольних поверхах заборонено.

– Мінімальна площа на одне робоче місце 6 м, мінімальний обсяг 20.

– Приміщення повинні бути обладнані системами природного й штучного висвітлення.

– Природне висвітлення повинне здійснюватися через світлові прорізи, орієнтовані переважно на північ, північний схід.

– Коефіцієнт природної освітленості (КЕО) повинен бути не нижче 1.5%.

– Приміщення із ВДТ ЕОМ не повинні граничити із приміщеннями, у яких рівень шуму й вібрацій перевищує припустимі норми.

– Віконні прорізи повинні бути обладнані регульованими пристосуваннями (жалюзі, штори, козирки, маркізи).

– Поверхня підлоги повинна бути рівної, не слизькою, з антистатичним покриттям.

Гігієнічні вимоги до параметрів виробничого середовища приміщень

Таблиця 5.1 - Мікроклімат приміщення

Пора року	Категорія роботи	Температура повітря,ºС	Відносна вологість,%	Швидкість повітря в робочій зоні, м/с
холодне	легка 1а	22 – 24	40 – 60	0.1
	легка 1б	21 – 23	40 – 60	0.1
тепле	легка 1а	23 – 25	40 – 60	0.1
	легка 1б	22 - 24	40 – 60	0.2

Легка 1а - робота, сидячи, що не потребує більших фізичних витрат, тепловиділення порядку 140 Вт.

Легка 1б - робота, що виконується сидячи або пов'язана з невеликими переміщеннями, не потребуючих більших фізичних витрат, тепловиділення порядку 175 Вт.

Таблиця 5. 2 - Рівні іонізації повітря для приміщень із ВДТ ЕОМ:

Рівні	Число іонів в 1 повітря	
	n +	n –
min необхідні	400	600
оптимальні	1500 – 3000	3000 – 5000
max припустимі	50000	50000

Гігієнічні вимоги до організації й устаткування робочих місць із ВДТ ЕОМ

Робоче місце необхідно розташовувати щодо світлових прорізів, щоб природне висвітлення падало з боку, переважно ліворуч. Відстань від тильної поверхні одного до екрана іншого повинне бути не менш 2.5 м. Стандартна висота стола 680 - 800 мм.

Захист відстанню полягає в тім, щоб оператор перебував на відстані не менш 600 - 700 мм від монітора з урахуванням розмірів знаків і символів. Клавіатуру необхідно розташовувати на поверхні стола на відстані від 100 до

300мм від краю поверхні. Поверхня клавіатури повинна бути матової з коефіцієнтом відбиття 0,4.

Вимоги до режимів роботи й відпочинку при роботі із ВДТ ЕОМ. При виконанні робіт протягом дня, які ставляться до різних видів трудової діяльності. За основну роботу з ЕОМ варто вважати таку, котра займає не менш 50% робочого часу. Для програміста - рекомендується 15 хв. перерва через щогодини роботи. Для оператора ЕОМ при 8-ми часовому робочому дні передбачена всередині зміни перерва - 15 хвилин через кожні 2 години роботи.

Долікарняна допомога потерпілому. Перша долікарняна допомога при нещасних випадках від електричного струму складається з 2-х етапів: звільнення потерпілого від дії струму й надання йому медичної допомоги.

Міри першої медичної допомоги потерпілому від електричного струму залежать від його стану. У свідомості – але до цього був у непритомності або тривалий час перебував під струмом, йому необхідно забезпечити повний спокій до прибуття лікаря або терміново доставити в лікувальну установу.

При відсутності свідомості, але збереженому подиху й роботі серця потрібно рівно й зручно укласти потерпілого на м'яку підстилку, розстебнути пояс і одяг, забезпечити приплив свіжого повітря. Варто давати нюхати нашатирний спирт. Якщо потерпілий погано дихає - рідко, судорожно - або якщо подих поступово погіршується, у той час як при цьому триває нормальна робота серця, необхідно робити штучне дихання.

При відсутності ознак життя треба робити штучне дихання й непрямий масаж серця до появи ознак життя або до прибуття лікаря.

Висновок. Більшу частину свого життя людина проводить на роботі, де його можуть чекати не тільки приємні моменти, але й перевтома, стреси й виробничі травми. Тому дуже важливо створити необхідні умови для нормальної роботи людини й збереження його здоров'я під час трудової діяльності на підприємстві. Для цього й створена система охорони праці, що займається перевіркою дотримання всіх описаних вище вимог і розрахунком всіх технічних показників.

ВИСНОВКИ

Результатом дипломної роботи є дослідження процесу клієнт-серверної взаємодії при виконанні математичних обчислень.

Основні завдання, вирішені в роботі: аналіз клієнт-серверної архітектури, аналіз особливостей взаємодії клієнта та серверу; розробка схеми процесу клієнт-серверної взаємодії при виконанні математичних операцій; розробка інтерфейсу користувача системи виконання математичних операцій; описання програмної частини для розробника; реалізація клієнт-серверної системи для виконання математичних операцій.

У розділі Реалізація клієнт-серверної системи наведено Опис роботи програмного забезпечення; Опис інтерфейсу користувача; Опис системи для розробника; Технології реалізації автоматизованої системи.

Для реалізації програмного забезпечення та вирішення задачі даного дипломного проекту була обрана клієнт-серверна технологія ASP.NET WEB API та мова програмування C#. Середою розробки для обраної технології є Microsoft Visual Studio, що має безліч інструментів для створення програмного забезпечення. Інтегроване середовище розробки Visual Studio використовується для редагування, налагодження та складання коду, а також для публікації програм.

Також виконано економічний розрахунок та розглянуто питання охорони праці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мережі зв’язку / Б. С. Гольдштейн, Н. А. Соколов, Г. Г. Яновский – 2010. – С. 19-20;

2. *EPON* [Електронний ресурс]. - Режим доступу до ресурсу: *https://uk.wikipedia.org/wiki/EPON*

3. *Stallings W. Foundations of Modern Networking: SDN, NFV, QoE, IoT, andCloud / W. Stallings. – Pearson Education, Inc., OldTappan, New Jersey*, 2016.

4. Вся статистика інтернету на 2020 рік [Електронний ресурс]. - Режим доступу до ресурсу: *https://adviana.ru/blog/internet-2020-v-rossii-i-mire-statistika-i-trendy.html*

5. Магазин все для телекомунікацій [Електронний ресурс]. - Режим доступу до ресурсу: *https://lantorg.com/products/kommutator-eltex-mes*5312

6. Кількість користувачів інтернетом в Україні [Електронний ресурс]. - Режим доступу до ресурсу: *https://www.epravda.com.ua/rus/news/2019/10/11/652498/*

7. Технологія *GPON, GEPON* або *EPON*

8. *World Internet Users and* 2020 *Population Stats* [Електронний ресурс]. - Режим доступу до ресурсу: *http://www.internetworldstats.com/stats.htm*

9. *International Telecommunication unit* [Електронний ресурс] – Режим доступу: *http://www.itu.int/*

10. Ресурс присвячений технології PON [Електронний ресурс] *http://ic-line.ua/wiki/1-etapy-proektirovaniya-pon-setej*

11. *Kurose J. F. Computer Networking: A Top-Down Approach, 7th Ed / James F. Kurose, Keith W. Ross. – Pearson Education, Inc.*, 2017.

12. Як COVID-19 змінив інтернет і нас - соцмережі і онлайн-рітейл після пандемії [Електронний ресурс]. - Режим доступу до ресурсу: *https://www.web-canape.ua/business/kak-covid-19-izmenil-internet-i-nas-statistika-interneta-i-socsetej-posle-pandemii/*

13. *Tanenbaum A. S. Distributed systems: principles and paradigms /Andrew S. Tanenbaum, Maarten Van Steen. – Pearson Education. Inc. Pearson Prentice Hall, Upper Saddle River, NJ* 07458, 2007.

14. *Number of internet users worldwide from* 2005 *to* 2019 [Електронний ресурс]. - Режим доступу до ресурсу: *https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/*

15. *Report on the state of the Internet environment – Digital* 2020 [Електронний ресурс]. - Режим доступу до ресурсу: *https://wearesocial.com/digital*-2020

16. Roussaki, I. Multi-terminal and Multi-network Access to Virtual Home Environment / I. Roussaki, H. Jormakka, S. Xynogalas, A. Laikari, M. Chantzara, M. Anagnostou.

17. *Göransson P. Software Defined Networks: A Comprehensive Approach, 2nd ed. / Paul Göransson, Chuck Black, Timothy Culver.– Morgan Kaufmann, US,* 2017. – 409 p.

18. *Coronavirus: the consumer impact* [Електронний ресурс]. - Режим доступу до ресурсу: *https://www.globalwebindex.com/coronavirus*

19. Системи доступу користувача. Модеми цифрового доступу: навчально-методичний посібник до лабораторних робот/Гайворонська Г.С., Сахаров В.І., Котова О.І. – [2-е вид.]. – Одеса 2008.