Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем
(повне найменування інституту, факультету)

Кафедра телекомунікацій
(повна назва кафедри)

До захисту допущено
В.о. завідувача кафедри
_____ Валерій ЯВІСЯ
(підпис) (Ім'я, прізвище)
"04" червня 2020_р.

Дипломна робота

на здобуття освітнього ступеня "бакалавр"
(назва ОС)

Спеціальність 172 Телекомунікації та радіотехніка,
(код і назва)

на тему: Аналіз протоколів взаємодії клієнтів з сервером повідомлень

Виконав: студент 4 курсу, групи ТМ-61
(шифр групи)

Коваль Олександр Володимирович
(прізвище, ім'я, по батькові) (підпис)

Керівник Доцент, к.т.н. доцент Явіся В.С.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2020_ року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем
(повна назва)

Кафедра телекомунікацій
(повна назва)

Освітній ступінь бакалавр

Спеціальність 172 Телекомунікації та радіотехніка
(код і назва)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

___ Валерій_Явіся
(підпис) (ім'я, прізвище)

"_22 "_січня_2020 р.

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Коваль Олександр Володимирович _____
1. Тема роботии "Аналіз протоколів взаємодії клієнтів з сервером повідомлень",
керівник роботи Доцент, к.т.н. доцент Явіся В.С. _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 30 березня 2020 р. №924-с

2. Термін подання студентом роботи 04 червня 2020
3. Вихідні дані до роботи протоколи взаємодії SOAP, REST, XML-RPC
4.Зміст роботи :
1) ОСНОВИ ТЕХНОЛОГІЇ КЛІЄНТ-СЕРВЕР
2) ПРОТОКОЛИ ВЗАЄМОДІЇ ТА СХЕМИ ВЗАЄМОДІЇ ОСНОВНИХ МОДЕЛЕЙ КЛІЄНТ-СЕРВЕР І СЕРВІС-ОРІЄНТОВАНОЇ АРХІТЕКТУРИ ВЕБ-ДОДАТКІВ
3) АНАЛІЗ ПРОТОКОЛІВ ВЗАЄМОДІЇ КЛІЄНТІВ З СЕРВЕРОМ ПОВІДОМЛЕНЬ
4) ВИСНОВКИ
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо):
1) Модель файлового сервера (FS); 2) Модель доступу до віддалених даних(RDA); 3) Модель сервера бази даних (DBS); 4) Модель сервера додатків (AS); 5) Дворівнева модель взаємодії клієнт-сервер; 6) Багаторівнева архітектура взаємодії клієнт-сервер;

7) Спрощена схема реалізації взаємодії між клієнтом і сервером; 8) Модель взаємодії відкритих систем ISO/OSI; 9) Вкладеність повідомлень різних рівнів; 10) Мережна взаємодія, яка поділена на чотири рівні; 11) Схема взаємодії FS-моделі; 12) Схема взаємодії RDA-моделі; 13) Схема взаємодії

DBS-моделі; 14) Схема взаємодії AS-моделі; 15) Концепція веб-сервісу; 16) Протоколи веб-сервісів (реалізація веб-сервісів); 17) Концепція XML-RPC; 18) Структура протоколу WSDL; 19) Місце UDDI в стеку протоколів веб-служб; 20) Принцип доступу до даних через ODBC; 21) Схема взаємодії прикладної та інтерфейсної частин; 22) Обробка даних на базі хоста; 23) Обробка даних на базі сервера; 24) Обробка даних на базі клієнта; 25) Спільна обробка даних; 26) REST відрізняється від SOAP; 27) Презентація.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 24.10.2019_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Пошук та аналіз літртури.	01.01.2020-31.01.2020	Виконано
2	Аналіз основних тенологій клієнт сервер.	01.02.2020-29.02.2020	Виконано
3	Аналіз схем взаемодії основних моделей клієнт сервер.	01.03.2020-31.03.2020	Виконано
4	Аналіз протоколів взаємодвї клєнта з сервером повідомлень.	01.04.2020-30.04.2020	Виконано
5	Загальний висновок.	01.05.2020-10.05.2020	Виконано
6	Оформлення дипломної роботи.	11.05.2020-31.05.2020	Виконано

Студент _____ _____Коваль О.В____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____Явіся В.С_____
(підпис) (прізвище та ініціали)

* Консультантом не може бути зазначено керівника дипломної роботи.

ANNOTATION

Graduate work dedicated analysis of client interaction protocols with the message server.In particular, the existing models of client-server communication were considered, the characteristics and classification of message servers were provided, Internet protocols were described, the model of open systems interaction - OSI model and its importance in determining different levels of system interaction, etc.A comparative analysis of some interoperability protocols identifies differences and features in their remote access techniques, and identifies the need to consider in the context of the implemented system and its limitations to answer the question of the best way to implement. Studies show the relevance of the problem under consideration and the need to take into account the conditions and areas of use of technological components of the system.The choice of interaction protocols requires considerable attention - this is a very important architectural decision on which the development of the project depends. Keywords: network, interaction protocol, interaction model, client, server, web service. The qualification work contains 61 pages, 26 figures and 1 table. 43 scientific and technical publications were used in the work.

АНОТАЦІЯ

Дипломна робота присвячена аналізу протоколів взаємодії клієнтів з сервером повідомлень. Зокрема були розглянуті існуючи моделі взаємозв'язку клієнт-сервер, надана характеристика і класифікація серверів повідомлень, описані протоколи Інтернету, модель взаємодії відкритих систем -модель OSI та її важливе значення при визначенні різних рівнів взаємодії систем, тощо. Завдяки порівняльного аналізу деяких протоколів взаємодії визначені відмінності і особливості у їхніх техніках видаленого доступу, та визначено необхідність розглядання в контексті реалізованої системи і її обмежень для надання відповіді на питання про кращій спосіб реалізації. Дослідження свідчать про актуальність проблеми, що розглядається, та потребу врахування умов та галузі використання технологічних ланок системи. Вибір протоколів взаємодії потребує значної уваги - це дуже важливе архітектурне рішення, від якого залежить розвиток проекту. Ключові слова: мережа, протокол взаємодії, модель взаємодії, клієнт, сервер, веб-служба. Кваліфікаційна робота містить 61 сторінок, 26 рисунків та 1 таблицю. В роботі використано 43 науково-технічних видань.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД - бази даних;

ЕОМ - електронно-обчислювальні машини;

СКБД- системи керування базами даних;

СУБД - (СУБД, СКБД-Database Management System, DBMS) система управління базами даних;

SQL - (Structured Question Language) мова структурованих запитів;

DBMS - (*Database Management System)* - система управління базами даних СУБД/Менеджер баз даних;

ArcSDE - (Spatial Database Engine) это підсистема програмного забезпечення сервера;

TCP - (Transmission Control Protocol) міжмережний протокол транспортного рівня;

IP - (Internet Protocol) протокол мережного рівня;

браузер - програма пошуку;

RDA - (Remote Date Access) модель доступу до віддалених даних;

DBS - (DateBase Server) модель сервера бази даних;

AS - (Application Server) модель сервера додатків ;

API - спеціальний інтерфейс прикладного програмування;

ОС - операційна система;

SMS – (System Management Server);

RAS - (Remote Access Service) сервіс віддаленого доступу;

RPC - (Remote Procedure Call) протокол віддаленого виклику процедур с помощью XML;

CORBA - (Common Object Request Broker Architecture) архітектура посередника об'єктних запитів;

RMI (Remote Method Invocation) – технологія віддаленого виклику методів;

DCOM (Distributed Component Object Model) розподілена модель компонентних об'єктів;

HTTP - (Hypertext Transfer Protocol) протокол передачі гіпертекстових файлів, это протокол, который позволяет отправлять документы в Интернете;

SOAP - (Simple Object Access Protocol) Версія 1.1 - протокол доступу до об'єктів, протокол обміну повідомленнями між споживачем і постачальником веб-сервісу, Версія 1.2 – це протокол представницького рівня, простий текстовий протокол, який використовує XML для передавання інформації між додатками;

XML/A (XML for Analysis)- протокол прикладного рівня, визначає методи доступу до багатовимірними даними і метаданими Analysis Services;

SMTP - (Simple Mail Transfer Protocol протокол пересилання пошти;

SOA -(Service-oriented architecture) сервіс-орієнтована архітектура;

FS - (File Server) модель файлового серверу;

RDA - (Remote Data Access) модель віддаленого доступу до даних;

datagram sockets - датаграммні гнізда - гнізда з віртуальним з'єднанням (у початковій термінології stream sockets);

DBS - (Data Base Server) модель сервера БД;

AS - модель сервера додатків (Application Server);

TPM-(Transaction Processing Monitor) менеджери транзакцій, монітори транзакцій;

API- (Application Program Interface) інтерфейс прикладного програмування;

RPC - (Remote Procedure Calling) видалений виклик процедур;

DAO - технологія «об'єктного доступу до даних»;

WSDL- (Web Services Description Language) мова опису зовнішніх інтерфейсів веб-служби;

UDDI - (Universal Discovery, Description and Integration) - універсальний інтерфейс розпізнавання, опису і інтеграції, використовуваний для формування каталогу веб-сервісів і доступу до нього;

ODBC - (Open Database Connectivity) відкритий інтерфейс доступу до бази даних);

POP3 (Post Office Protocol) Поштовий Офісний Протокол, дозволяє клієнтові мати вибірковий доступ до повідомлень на сервері;

SMTP - Простий протокол передачі пошти;

IMAP4- Протокол доступу до повідомлень в Інтернет;

IIS- (Internet Information Services) середній рівень архітектури, що використовується Internet додатками;

OLE – (Object Linking and Embedding) впровадження і зв'язування об'єктів;

LDAP - (Lightweight Directory Access Protocol) - полегшений протокол доступу до списків;

IDL - (Interface Definition Language) мова специфікації Інтерфейсів об'єктів;

ORB - (Object Request Broker) брокер об'єктних заявок;

Inetd - (internet super-server daemon) демон сервісів IP;

OSI - (Open System Interconnection) модель взаємодії відкритих систем;
CRUD -(Create Read Update Delete);

PDU- (ProtocolData Unit) протокол пользовательских дейтаграмм.

ЗМІСТ

ВСТУП

Розширення ринку інформаційних послуг привело до випуску удосконалених інтелектуальних програмних комплексів, які ставали об'ємними. Це було перетином для розміщення придбаних продуктів на власних ЕОМ. Для обміну інформацією [1,2] і її поширення були створені мережі ЕОМ, а узагальнюючі програми і дані почали встановлювати на спеціальних файлових серверах.

Завдяки працюючим з файловими серверами СУБД, безліч користувачів отримують доступ до одних і тих же БД. Уся обробка запитів з програм або з терміналів призначених для користувача ЕОМ виконується на них. Збереження інформації забезпечують сервери, обладнані засобами віддаленого доступу на основі провідних або бездротових комунікаційних мереж [3]. Сучасні програмні засоби здатні обробляти дані, розподілені на безлічі баз даних, керування якими здійснюють СКБД, що працюють на апаратних платформах з різними ОС, об'єднаними комунікаційними мережами. Це стало можливим завдяки збільшенню об'ємів внутрішньої та зовнішньої пам'яті, підвищенню швидкодії ЕОМ, збільшенню швидкості передачі даних. Робота з віддаленою БД здійснюється шляхом відправлення запиту, який звуть клієнтом, він обслуговується сервером. Сервер може обслуговувати декілька запитів одночасно, а клієнт теж може одночасно звертатися до декількох серверів. Схема файл - сервер при невеликих обсягах даних задовольняє користувачів. З розробкою й впровадженням на рівні серверів БД механізму збережених процедур з'явилася технологія активного сервера БД. В якому частина функцій прикладного компонента реалізовані у вигляді збережених процедур, виконуваних на стороні сервера, а інша - виконується на клієнтській стороні. Протокол взаємодії - відповідний діалект мови SQL [5].

Файл-сервер призначений для зберігання файлів та організації загального доступу до них, а файл-серверна технологія - робота в мережевому просторі з доступом до файлів СУБД, що зберігаються на сервері [4].

Позитивним є: низька вартість розробки; швидкість розробки; невисока вартість оновлення та змін.

Негативним є: низька продуктивність, що залежить від продуктивності мережі, сервера, клієнта; великий час очікування клієнтів. Для задоволення запиту записуються файли на сервер, що перевантажує мережу, з'являються вразливості.

Перерви в роботі скорочуються при використанні технології клієнт-сервер.

Архітектура клієнт-сервер - це новий етап в розвитку мережевих інформаційних технологій, стала відома з початку 90-х років XX століття. Особливість технології- додаток ділиться на подання даних клієнтом і зберігання БД сервером. Завдяки протоколам відбувається взаємодія між клієнтом і сервером. Запити, надіслані клієнтом, класифікують. Сервер після аналізу і обробки запиту, відповідає. Стандартне програмне забезпечення, що реалізує технологію клієнт – сервер, має нарощене ефективне апаратне забезпечення, стійкість і потужність в роботі і з величезними об'ємами інформації [6]. Забезпечується безпека БД від збоїв обладнання, від несанкціонованого доступу, висока продуктивність, знижаються навантаження на мережу [4].

Серверні СУБД мають широкі можливості управління призначеними для користувача правами доступу до різних об'єктів БД, резервного копіювання й архівації даних, та оптимізації виконання запитів, надають можливість паралельної обробки даних. Мережі передачі даних включають велику кількість клієнтів і велику кількість серверів. В якості серверів використовуються потужні комп'ютери з розділеною пам'яттю та периферією. Системне програмне забезпечення вибране з багатозадачних ОС.

За технологією клієнт-сервер користувач не керує ходом виконання поставленої проблеми, автоматично її вирішує система, використовуючи ресурси технічних засобів, БД та засобів телекомунікації. Одна задача може вирішуватись багатьма робочими станціями та серверами [4,5].

Архітектура клієнт-сервер - це шаблон програмного забезпечення для розподілених мережних застосувань, передбачає взаємодію та обмін даними [5].

Систему можна розділити на функціональні блоки за рахунок взаємодії додатків серверів, які мають настройку для користування лише загальнодоступним функціоналом - деталі реалізації машини сервера недоступні, з нею в’яжуть окрему БД. Системи легко адаптуються під веб. На веб-технологію можна просто перевести багаторівневу систему. Обмін інформацією одночасно відбувається між однією серверної машиною додатків і декількома серверами.

Актуальність теми. Вся мережа Інтернет побудована за принципом клієнт-серверної архітектури. Розширення областей застосування обчислювальної техніки і складність завдань вимагають вдосконалення методів інформаційних систем. Тенденції розвитку зумовлені необхідністю підвищення швидкості доступу кінцевого користувача до потрібної інформації. Багато інформаційних систем, що використовуються, є відкритими з клієнт-серверною технологією, яка розділяє роботу інформаційної системи в комп’ютерній мережі між сервером (забезпечує функціональність інформаційної системи) та клієнтом (здійснює введення, попередню обробку та передачу інформації на сервер і кінцеву обробку та візуалізацію отриманих з сервера даних). В рамках загальної структури додатків робота може бути розділена між клієнтом і сервером по-різному.

Технології побудови клієнтського програмного забезпечення ґрунтуються на використанні різних типів протоколів. Не існує загальноприйнятої класифікації методів взаємодії клієнта з сервером. Питання клієнт-серверної взаємодії з використанням протоколів взаємодії при активізації удосконалення клієнт-серверних технологій і розвитку комп’ютерних мереж є важним, розробка механізмів завантаження і клієнтського програмного забезпечення та його взаємодії з серверами теж є актуальною задачею.

Завдання. В ході виконання роботи поставлене завдання з розширення теоретичних знань в області клієнт-серверної взаємодії на базі протоколів обміну, отриманих під час аналізу проблеми.

Об’єктом дослідження є набір правил, визначений телекомунікаційними протоколами і форматами для здійснення взаємодії клієнтів з сервером повідомлень, та які визначають поведінку функціональних блоків мережі при передачі даних, що забезпечує їхню передачу в потрібних напрямках і правильну інтерпретацію даних усіма учасниками процесу інформаційного обміну.

Предметом дослідження є протоколи взаємодії (обміну).

Метою дослідження є аналіз існуючих протоколів взаємодії клієнтів з сервером повідомлень, виявлення переваг і недоліків протоколів, опис використання.

Методи дослідження. Під час проведеного аналізу використовувалися статистичні і логічний методи, метод елементарно-теоретичний і класифікації.

Наукова новизна одержаних результатів. В результаті виконання даної роботи були з’ясовані тонкощі і особливості протоколів взаємодії, які вивчалися шляхом дослідів документації в поєднанні з досвідом науковців і розробників, що викладений в науково-технічній інформації, яка аналізувалася.

В основі будь-якого сервера повідомлень лежить протокол взаємодії і будь-який сервер виконує вимоги вказаного протоколу, а відповіді містять потрібні для передачі по протоколу параметри. В роботі охарактеризовано будови протоколів, а для доказу актуальності теми здійснено порівняння з іншими протоколами.

Під час аналізу доведено популярність досліджень щодо вибору протоколів взаємодії, це є важливим архітектурним рішенням і потребує значної уваги. Результат цього вибору має вплив на подальший розвиток перспективного технічного напряму та формує вимоги до клієнт-серверної архітектури.

1 ОСНОВИ ТЕХНОЛОГІЇ КЛІЄНТ-СЕРВЕР

1.1 Файловий сервер та модель клієнт-сервер

1.1.1 Інтернет побудований за принципом клієнт-серверної архітектури.

Базовою перспективною технологією для локальних мереж була найперша FS - модель файлового сервера, поширена серед вітчизняних розробників, які використовували такі системи, як FoxPro, Clipper, Clarion, Paradox, тощо [7-15]. Взаємодія клієнта і сервера - запит направляється на FS, який передає СУБД необхідний блок даних. Вся обробка здійснюється на терміналі (рисунок 1.1).

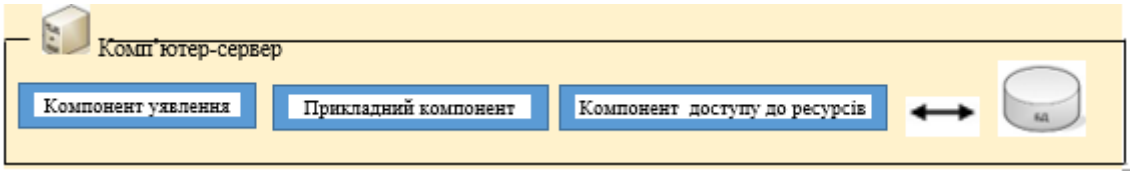

Рисунок 1.1 Модель файлового сервера (FS)

Протокол обміну це набір викликів до файлової системи на файл-сервері.

Для СУБД архітектура клієнт-сервер стала стандартом, прикладні програми матують розподілений характер. Виділяють підходи, що реалізовані у моделі технології клієнт - сервер: модель доступу до віддалених даних (RDA); модель сервера бази даних (DBS); модель сервера додатків (AS).

Моделі RDA, DBS, AS є мережеві архітектури технології клієнт – сервер.

У моделі RDA коди компонента уявлення і прикладного компонента суміщені і виконуються на комп'ютері-клієнті, доступ до інформаційних ресурсів забезпечено мовою SQL - запитів для БД, або викликами функцій API [15].

Запити до інформаційних ресурсів направляються по мережі віддаленого комп'ютера (рисунок 1.2).

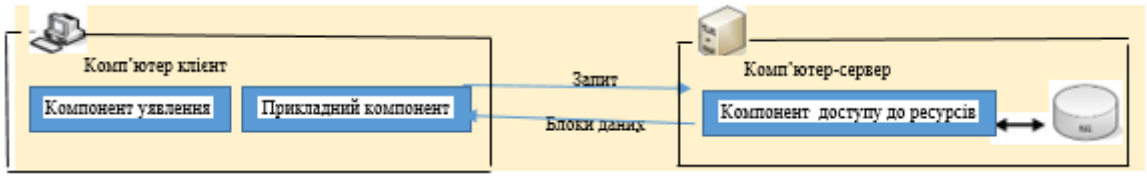

Рисунок 1.2 Модель доступу до віддалених даних (RDA)

Модель DBS має основу механізм збережених процедур; поняття інформаційного ресурсу звужено до БД, додаток розподілений. Компонент уявлення виконується на комп'ютері-клієнті, прикладний компонент - набор збережених процедур, функціонує на комп'ютері-сервері БД [15]. Збережені процедури -процедури БД (рисунок 1.3).

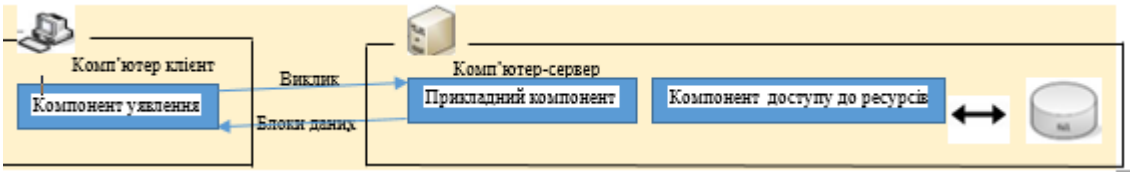

Рисунок 1.3 Модель сервера бази даних (DBS)

AS-модель є удосконаленням DBS. AS - є процес, що виконується на комп'ютері-клієнті і відповідає за інтерфейс з користувачем - введення і відображення даних. Основний елемент є прикладний компонент - сервер додатка, що функціонує на віддаленому комп'ютері. Сервер додатків є група прикладних функцій у виді сервісів, кожен надає послуги бажаним програмам.

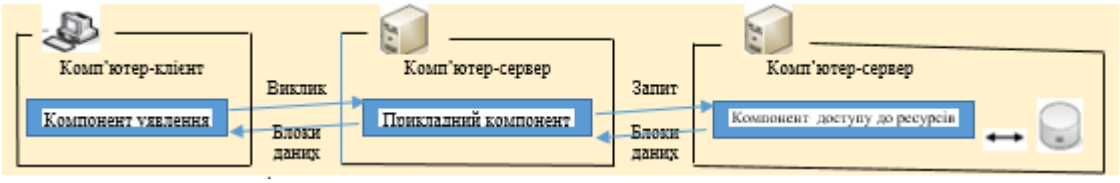

Рисунок 1.4 Модель сервера додатків (AS)

Може бути кілька додатків серверів, кожен надає певний набір послуг, програма є клієнт додатком. В СУБД забезпечується захист даних вбудованою перевіркою повноважень клієнта і дані прикриті за сервером додатків.

1.1.2 Технології файловий сервер і клієнт – сервер

Баланс між клієнтом і сервером може бути, коли на сервері виконується компонент додатка [15]. При роботі з невеликими інформаційними системами ефективно використати FS – модель, проблему графічного інтерфейсу вирішує RDA-модель, DBS-модель є гарним варіантом для СУБД, AS-модель – для створення великих інформаційних систем та під час використання низько швидкісних каналів зв'язку. Моделі можна використовувати в об'єднанні.

Помилки при визначенні моделі і платформи можуть перекреслити всі переваги прикладної частини інформаційної системи.

Перспективним напрямком є розподіл функцій системи між клієнтською і користувача частинами СУБД визначається під час установки системи [15].

Основоположним механізмом організації обробки даних в СУБД з архітектурою клієнт-сервер є механізм транзакцій [15,16].

За технологією клієнт-сервер автоматично керується хід виконання проблеми в системі за допомогою ресурсів технічних засобів, баз даних та засобів телекомунікацій. Одна задача вирішується багатьма робочими станціями та серверами, які є територіально розподіленими [16].

1.2 Модель взаємозв’язку клієнт-сервер. Основні визначення.

Будь-яка програма обов’язково здійснює взаємодію у мережі.

Не існує нормативно визначених понять «клієнт» і «сервер». При проведенні досліджень важливо уявляти і ув’язувати ці поняття [7].

Під поняттям «клієнт» в клієнтський комп'ютер, що звертається до інших комп'ютерів; клієнтське та серверне програмне забезпечення; людей, які за допомогою програмно-апаратного забезпечення отримують доступ до інформації

Сервером (в перекладі обслуговуючий [9]) від призначення називають:

/мережу/ сервер - логічний або фізичний вузол мережі, що обслуговує запити адресу і/або доменному імені, що складається з одного або з системи апаратних серверів, на якому виконуються одна або система серверних програм;

/програмне забезпечення/ сервер - програмне забезпечення, що приймає запити від клієнтів в архітектурі клієнт-сервер;

/апаратне забезпечення/ сервер - комп'ютер або спеціальне комп'ютерне обладнання, що виділений і/ або спеціалізований для виконання певних функцій; сервер в IT - програмний компонент обчислювальної системи, що виконує сервісні функції по запиту клієнта, надання йому доступу до певних ресурсів.

Взаємозв'язок понять: серверний додаток – сервер, що запускається на комп'ютері, і його називають «сервер»; при розгляді топології мережі, вузол також називають «сервером». У загальному випадку може бути, що серверний

додаток запущений на робочій станції, або серверний додаток, запущений на серверному комп'ютері, в рамках даної топології виступає в ролі клієнта - не є сервером з точки зору мережевої топології [9].

В технічній літературі [5,10] під програмними модулями мають на увазі клієнти і сервери, які найчастіше фізично розміщені на різних комп'ютерах, але, якщо сервер є локальним, то програми – і клієнтська, і серверна, розміщуються на одному комп’ютері. Процеси, що реалізують служби файлової системи або БД, називаються серверами. Процеси, що запитують служби у серверів і в подальшому очікують відповіді від сервера, називаються клієнтами. Такими є системи обробки даних на основі СУБД.

Модель клієнт-серверної взаємодії визначається розподілом [7] обов'язків між клієнтською та серверною програмами. Для запит-відповідь використовуються протоколи: tcp / ip, http, rdp і інші. Процеси клієнта і сервера незалежні один від одного. Відповідно до ступеня поділу процесів між клієнтом і сервером вони бувать слабкими (тонкими) або сильними (товстими). Слабкий клієнт виконує мінімум обробки на стороні клієнта, а сильний - відносно велику частину обробки даних. Сильний сервер несе основне навантаження по обробці даних, а на слабкий сервер навантаження невелике. Система з мейнфреймів є прикладом максимально сильного сервера і максимально слабкого клієнта [11].

Архітектура клієнт-сервер визначає загальні принципи взаємодії між комп'ютерами, деталі взаємодії визначають протоколи. Взаємодію починає клієнт, а правила, за якими відбувається взаємодія описує протокол.

Клієнт-серверні системи бувають двохрівневі (клієнт запитує сервіси у сервера) і трирівневі (запити обробляються проміжними серверами, які координують виконання клієнтських запитів з підлеглими їм серверами) [4].

Трирівневі архітектури взаємодії клієнт-сервер мають назву багаторівнева архітектура клієнт-сервер (іноді називають трьохланцюгова архітектура).

У дворівневій архітектурі обробка запиту відбувається на одній машині без сторонніх ресурсів. Пред'являються жорсткі вимоги до продуктивності сервера. Дворівневу модель взаємодії зображена на рисунку 1.5.

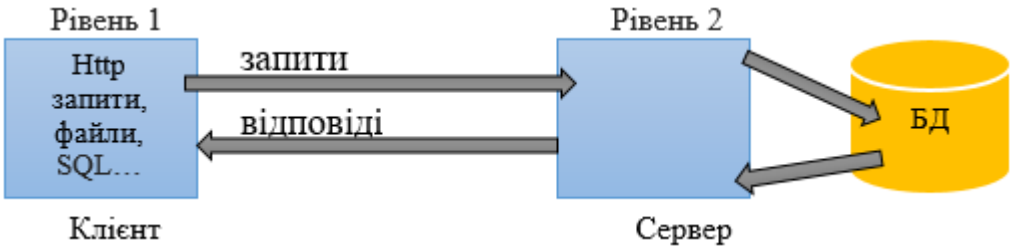

Рисунок 1.5 Дворівнева модель взаємодії клієнт-сервер

Перший рівень –робить запит клієнт, другий - обробляє цей запит сервер.

Суть багаторівневої архітектури [4] - запит клієнта обробляється відразу декількома серверами. Розподіл операцій значно знижує навантаження на сервер, та даний підхід менш надійний ніж двохланцюгова архітектура. На рисунку 1.6 наведено приклад багаторівневої архітектури клієнт-сервер.

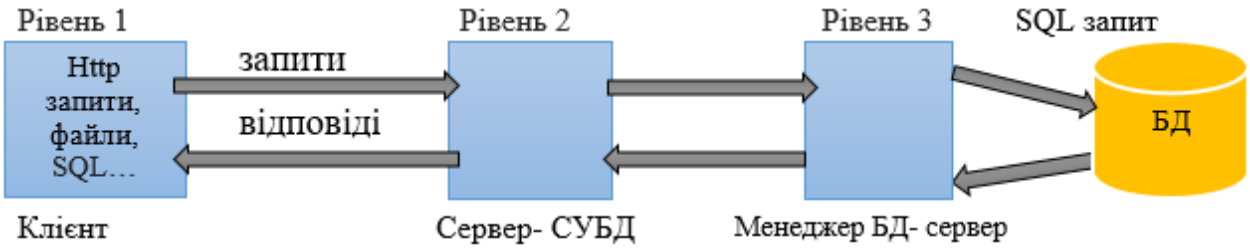

Рисунок 1.6 Багаторівнева архітектура взаємодії клієнт-сервер

В контексті систем управління базами даних перший рівень - це клієнт, що надсилає SQL запити до бази даних; другий рівень - це СУБД, який реалізує взаємодію між клієнтом і файлової системою, а третій рівень - це сховище даних.

Якщо розглядати архітектура з позиції сайту: перший рівень - це браузер, за допомогою якого відвідувач заходить на сайт, другий рівень - це зв'язка Apache + PHP, а третій рівень - це база даних.

Клієнт-серверна архітектура - концепція інформаційної мережі, в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів.

Архітектура визначає типи компонентів: набір серверів, які надають інформацію чи інші послуги програмам і які звертаються до них; набір клієнтів, які використовують сервіси, що надаються серверами; мережа, яка забезпечує взаємодію між клієнтами та серверами.

Спрощена схема взаємодії між клієнтом і сервером, показана на рисунку1.7 [7].

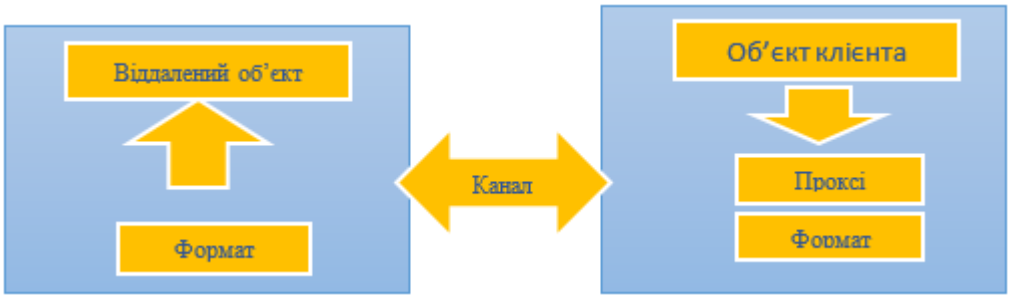

Рисунок 1.7 Спрощена схема реалізації взаємодії між клієнтом і сервером

Серверна програма надає інформацію/інші послуги програмам, які звертаються до неї, а клієнтська програма використовує ці сервіси.

У файл-серверної системи дані зберігаються на файловому сервері - Novell NetWare або Windows NT Server. Їхня обробка здійснюється на робочих станціях, на яких, функціонує одна з СУБД: Access, FoxPro, Paradox і т.п. Додаток на робочій станції відповідає за формування для користувача інтерфейсу, за логічну обробку даних, за безпосереднє маніпулювання даними. Файловий сервер надає послуги найнижчого рівня - відкриття, закриття та модифікацію файлів [9, 12].

У клієнт-серверної системи функціонують не менше ніж два додатки - клієнт і сервер, що ділять між собою функції, які в файл-серверній архітектурі виконує додаток на робочій станції. Зберіганням і маніпулюванням даними займається сервер БД, наприклад, Microsoft SQL Server, Oracle, Sybase і т.п .

Формуванням для користувача інтерфейсу займається клієнт, для побудови якого використовувати ряд спеціальних інструментів, а також більшість настільних СУБД. Логіка обробки даних може виконуватися і клієнтом, і сервером. Клієнт посилає на сервер запити, сформульовані, як правило, на мові SQL. Сервер обробляє їх і передає клієнту/клієнтам результат.

Безпосереднім маніпулюванням даними займається один процес, обробка даних відбувається там, де дані зберігаються - на сервері, що виключає необхідність передачі великих обсягів даних через мережу.

Взаємодія між клієнтом і сервером відбувається за допомогою стандартних протоколів: TCP / IP і z39.50. Протоколи розрізняються за рівнями [13,14].

1.3 Програмне забезпечення технології клієнт - сервер

Взаємодія клієнтського і серверного процесів виконується програмним забезпеченням передачі даних та складається з рівнів програмного забезпечення. Програмне забезпечення прив'язане до мережі. Всі клієнтські запити і відповіді сервера передаються по мережі в формі повідомлень [11].

У архітектурі клієнт-сервер комп'ютери-сервери є потужнішими ніж комп'ютери-клієнтів. Завданням серверів є адміністрування, одночасне виконання запитів, захисту інформації і інше. Використовується програмне забезпечення, що включає клієнтську і серверну частини [15].

На сервері розміщені збережені процедури - попередньо написані короткі процедури обробки даних, які використовуються будь-яким клієнтом. Збережені процедури допомагають обробляти дані, зменшуючи довжину коду і дискового об’єму на комп'ютерах-клієнтах. Збережена процедура викликається будь-якою кількістю клієнтів. Вказані процедури зменшують мережевий трафік, єдине звернення клієнта призводить до виконання серії команд збереженої процедури, може проводитися контроль безпеки.

Інструментальні засоби, додатки та утиліти для інтерфейсної частини доповнюють можливості моделі клієнт-сервер. Це засоби запитів, які спрощують доступ до даних сервера, використовуючи зумовлені запити і вбудовані можливості для побудови звітів для користувача програми, вони можуть бути в якості інтерфейсів. Додатки Microsoft Access, тощо мають свій власний SQL, це є можливістю доступу до систем управління базами даних від різних виробників.

Засоби розробки програм (Microsoft Visual Basic) полегшують створення додатків, які відповідають за доступ до серверів БД.Від вибору ОС і поставлених завдань визначається програмне забезпечення:

ОС Windows на комп'ютері - клієнті використовується пакет Microsoft Office з текстовим процесором Word, табличним процесором Excel, системою PowerPoint та управління базами даних Access і програма управління інформацією Outlook;

Windows Server 2000/2003/2008 забезпечує спільне використання файлів, друкуючих пристроїв, надає послуги по з'єднанню з клієнтськими комп'ютерами.

Для мережевій операційній системи використовують Windows 2000/2003/2008 Server, може використовуватися і на робочій станції для реалізації додаткових можливостей, забезпечує спільне використання безліч процесів і ресурсів багатьма користувачами. Можливість з'єднання з віддаленими мережами реалізується через сервіс віддаленого доступу - RAS, та через засоби зв'язку з мережами інших фірм -Novell, Digital Pathworks і Apple.

SMS дозволяє адміністратору централізовано керувати всією мережею, надає різні види сервісу: автоматизацію установки і поширення програмного забезпечення, оновлення, віддалене усунення несправності, надання повного контролю за пристроями введення і екранами всіх комп'ютерів в мережі.

SQL Server - система керування БД на принципах технології клієнт-сервер.

MS SQL Server підтримує систему обробки транзакцій і механізм розподілених транзакцій, систему збереження посилальної цілісності і тиражування даних.

SNA Server дозволяє декільком настільним ПЕОМ, які працюють під управлінням різних операційних систем взаємодіяти з хост-комп'ютерами.

Exchange Server забезпечує засоби передачі і прийому повідомлень в інформаційній мережі організації. Сервіс включає електронну пошту і обмін інформаційними повідомленнями для робочих груп.

Microsoft Exchange Server масштабується у відповідності з зростанням обчислювальних можливостей мережі.

Internet Information Server забезпечує створення Web-, FTP- і Gopher-серверів, за допомогою програми Internet Service Manager підтримує управління.

Програмним забезпеченням комп'ютерних мереж називають комплекс програм, що підтримують функції обміну інформацією між окремо розташованими ЕОМ. Це є складовою частиною операційних систем [5].

Програмне забезпечення клієнта надає користувачу сервіси прямого мережевого обміну з допомогою протоколів: TCPЯP, IPX, NetBios, Ethernet, X.25.

1.4 Модель взаємодії відкритих систем OSI

Інформаційний обмін це багатофункціональний процес. Функції групуються по призначенню в групи - рівні взаємодії. Уніфікація рівнів дозволяє створювати мережі зі складною топологією [5], в основі якої є поняття еталонної мережної моделі, яка описує порядок мережної взаємодії, що реалізується у вигляді стека протоколів. Протокол є угодою, прийнятою двома взаємодіючими в мережі комп'ютерами, не є обов'язково стандартним. При реалізації мереж найчастіше використовують стандартні протоколи (фірмові, національні або міжнародні).

На початку 80-х років розроблена модель OSI [17], яка визначає рівні взаємодії систем, дає стандартні імена і вказує, які функції виконує кожний рівень. В OSI представлені правила управління потоками даних в мережі клієнт-серверних систем. Модель основана на семи незалежних рівнях. Верхні рівні додатка надають сервіси локального комп'ютера для підготовки і форматування даних, що відсилаються; забезпечують користувача функціями для зв’язку з додатками. Сервіси рівнів сеансовий, транспортний, мережний, канальний, подання даних і фізичний, забезпечують безпеку доставки даних від одного мережевого вузла до іншого. Рівень сеансу використовується для управління зв'язком програма - програма. Нижчі рівні відносяться до мережі [11].

За моделлю (рисунок 1.8) взаємодії відкритих систем ISO/OSI [17] виділяються засоби взаємодії, що відповідає одному аспекту взаємодії мережевих пристроїв.

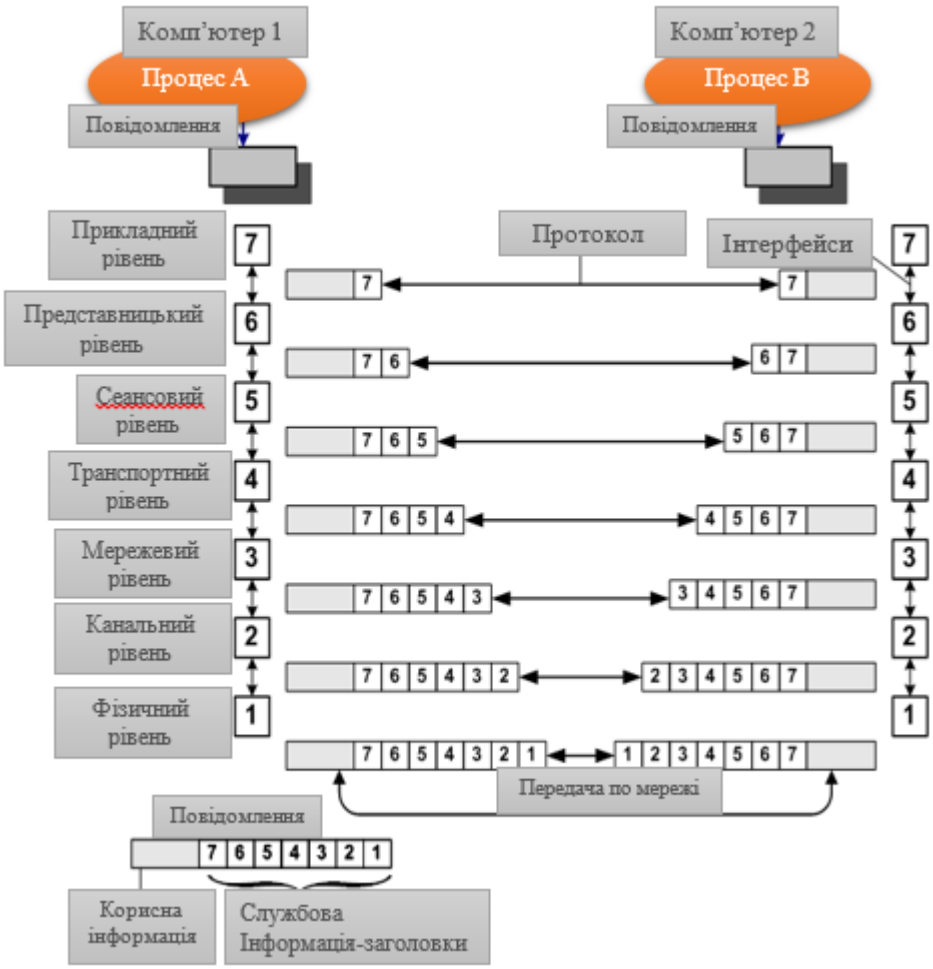

Рисунок 1.8. Модель взаємодії відкритих систем ISO/OSI

Модель OSI описує системні засоби взаємодії, що реалізовуються операційною системою, системними утилітами і апаратними засобами. Не включаються засоби взаємодії додатків кінцевих користувачів. Власні протоколи взаємодії додатку реалізовуються під час звернення до системних засобів. Розрізнюють рівень взаємодії додатків і прикладний рівень. Додаток може взяти на себе функції верхніх рівнів моделі OSI.

У стандартах ISO для позначення одиниць даних, з якими мають справу протоколи різних рівнів, використовується загальна назва протокольний блок

даних, (PDU). Для позначення блоків даних рівнів використовуються назви: кадр (frame), пакет (packet), дейтаграмма (datagram), сегмент (segment) [17].

У моделі OSI розрізнюються два основних типи протоколів:

Перший тип - протоколи з встановленням з'єднання (connection-oriented) перед обміном даними. Відправник і одержувач встановлюють з'єднання, вибирають параметри протоколу, після закінчення діалогу з'єднання розривається.

Другий тип – протоколи дейтаграмні без попереднього встановлення з'єднання (connectionless). Відправник передає готове повідомлення.

При взаємодії використовуються протоколи обох типів.

На рисунку 1.9 зображено вкладеність повідомлень різних рівнів

Рисунок 1.9 Вкладеність повідомлень різних рівнів

Повідомлення по мережі поступає на машину-адресата, воно приймається фізичним рівнем і послідовно піднімається з рівня на рівень. Кожний рівень аналізує і обробляє заголовок свого рівня, виконує відповідні даному рівню функції, видаляє цей заголовок і передає повідомлення вище - наступному рівню.

1.5. Класифікація серверів повідомлень [9].

Як правило, майже кожен сервер – є сервером повідомлень, обслуговує один/або кілька схожих протоколів, їх класифікують за типом послуг.

1.5.1 Універсальні сервери - вид серверної програми, він не надає послуг самостійно, надає серверам послуг спрощений інтерфейс до ресурсів між процесами взаємодії і/або уніфікований доступ клієнтів до послуг. Види серверів:

Inetd - стандартна програма UNIX-систем, дозволяє писати сервери TCP/IP протоколів, що працюють через inetd потоки вводу і виводу (stdin і stdout);

RPC - система інтеграції серверів у виді процедур доступних для виклику віддаленим користувачем через інтерфейс, який винайдений Sun Microsystems для своєї операційної системи (SunOS, Solaris; Unix-система, Windows).

Прикладні клієнт-серверні технології Windows:

(D-) COM - дозволяє програмам виконувати операції над об'єктами даних використовуючи процедури інших програм. Використовується для впровадження і зв'язування об'єктів (OLE), дозволяє писати прикладні сервери. COM працює в межах одного комп'ютера, DCOM доступна віддалено через RPC.

Active-X - розширення COM і DCOM для створення мультимедіа-додатків.

Універсальні сервери використовуються для написання інформаційних серверів, яким не потрібна специфічна робота з мережею, а мають завдання обслуговування клієнтів. Серверами для inetd є консольні програми та скрипти.

Більшість серверів Windows працюють через сервери (RPC, (D-) COM).

Мережеві служби забезпечують функціонування мережі. Сервери тунелювання (VPN-сервери) і проксі-сервери забезпечують зв'язок з мережею, недоступною роутингом. Сервери AAA і Radius забезпечують в мережі єдину аутентифікацію, авторизацію і ведення логів доступу.

1.5.2 Інформаційні служби. [9].

1.5.2.1 До інформаційних служб відносять сервери, які повідомляють інформацію про хости (time, daytime, motd), користувачів (finger, ident), а також сервери для моніторингу (SNMP).

Більшість інформаційних служб працюють через універсальні сервери.

Особливим видом інформаційних служб є сервери синхронізації часу – NTP.

1.5.2.2 Файл-сервери для забезпечення доступу до файлів на диску сервера.

Файл-сервери для забезпечення доступу до файлів на диску сервера -це сервери передачі файлів на замовлення, по протоколах FTP, TFTP, SFTP і HTTP. Протокол HTTP орієнтований на передачу текстових файлів, але сервери

можуть віддавати в якості запитаних файлів і довільні дані - динамічне створені веб-сторінки, картинки, музику і т. п. Інші сервери дозволяють монтувати дискові розділи сервера, вони працюють через інтерфейс RPC. Найпростішим є LDAP.

Не існує єдиного протоколу доступу до серверів БД, всіх об'єднує - мова SQL.

Сервери електронної пошти працюють по протоколу SMTP.

Для організації конференцій є сервери новин- працюють по протоколу NNTP.

Для обміну повідомленнями в реальному часі є сервери чатів, стандартний чат-сервер працює по протоколу IRC - розподілений чат для Інтернету.

Існують інші чат-протоколи, наприклад ICQ або Jabber.

1.5.2.3 Сервери віддаленого доступу [9].

Сервери віддаленого доступу, через відповідну клієнтську програму, забезпечують користувача консольним доступом до віддаленої системи. Для доступу до командного рядка служать сервери telnet, RSH, SSH. Інтерфейс для Unix- систем- X Window System є вбудований сервер віддаленого доступу.

Windows називається термінальний сервер. Протокол SNMP надає різновид управління-моніторингу і конфігурації, якщо комп'ютер або апаратний пристрій мають SNMP-сервер.

Ігрові сервери, служать для гри кількох користувачів в єдиній ігровій ситуації.

Серверні рішення - операційні системи та/або пакети програм, оптимізовані під виконання комп'ютером функцій сервера і/або містять комплект програм для реалізації типових сервісів. В інтегрованих серверних рішеннях установка всіх компонентів виконується одноразово, всі компоненти тісно інтегровані і попередньо налаштовані один на одного, тому заміна одного з серверів або вторинних додатків буває проблемою. Компонування окремих серверних додатків призначається для виконання більшості типових завдань; зниження складності розгортання і зниження загальної вартості IT-інфраструктури.

Проксі-сервер - служба в комп'ютерних мережах для виконання непрямих запитів до інших мережних служб.

Класифікації засобів модульної взаємодії наведені в таблиці 1.1 [7].

Таблиця 1.1 Класифікації засобів модульної взаємодії між клієнтом і сервером

Класифікація модульної взаємодії між клієнтом і сервером			
Активація режиму	Стан віддаленого об’єкта	Основа протоколів	Форматування
Сервер активних об’єктів	За значенням	HTTP-канал	XML формат
Клієнт активних об’єктів	За посиланням	ТСП-канал	Двійковий формат

1.6. Протоколи Інтернет [18].

Для побудові Інтернет мереж є важливими протоколи: мережного рівня IP; протокол транспортного рівня TCP. Усі параметри (і швидкість передачі даних, і методи адресації при транспортуванні повідомлень) визначаються і задаються протоколами даної конкретної мережі.

Протоколи Інтернету — це сукупність погоджень для визначення обміну даними між програмами. Вони задають способи передачі даних, повідомлень, обробку помилок мережі, дозволяють розробити стандарти не пов'язані з конкретною апаратною платформою. Базовий протокол - це протокол TCP/IP. TCP/IP не є єдиним протоколом, що дозволяє з'єднувати різні мережі. Інтернет з багатопротокольною мережею інтегрує й інші стандарти. Основні серед них — стандарти взаємодії відкритих систем OSI. Системи, які засновані на інших протоколах, підключаються до Інтернет через шлюзи.

Протокол TCP/IP використовує сокети [7]. Сокети стандартизовані і широкодоступні. Програмування з застосуванням сокетів розглядається як низькорівневе, це перешкоджає написанню стійких розподілених додатків.

Стандарт взаємодії прикладної програми з ядром ОС, що є точкою входу в мережу для додатка - є інтерфейс BSD сокетів. Сокети розраховані на роботу в клієнт-серверній парадигмі взаємодії: активний клієнт підключається до пасивного сервера, який одночасно обробляє багато клієнтських з'єднань. Для ідентифікації сервера при сокетних з'єднаннях використовується пара IP-адреса-порт. Порт - це унікальне число в рамках одного хоста. Один порт може одночасно використовувати тільки один процес ОС.

Відзначено [37], що клієнт і сервер є типовими користувальницькими процесами, у той час як програмні модулі, що реалізують функції протоколів

TCP і IP, є частиною ядра ОС Linux. Стандарт взаємодії прикладної програми з ядром ОС з'єднує прикладний рівень стека TCP/IP, який реалізується в просторі користувача, з нижнім рівнями, які реалізуються в ядрі ОС [36].

На рисунку 1.10 показана взаємодія клієнт - сервер в мережі Інтернет, які використовують для з'єднання протокол TCP.

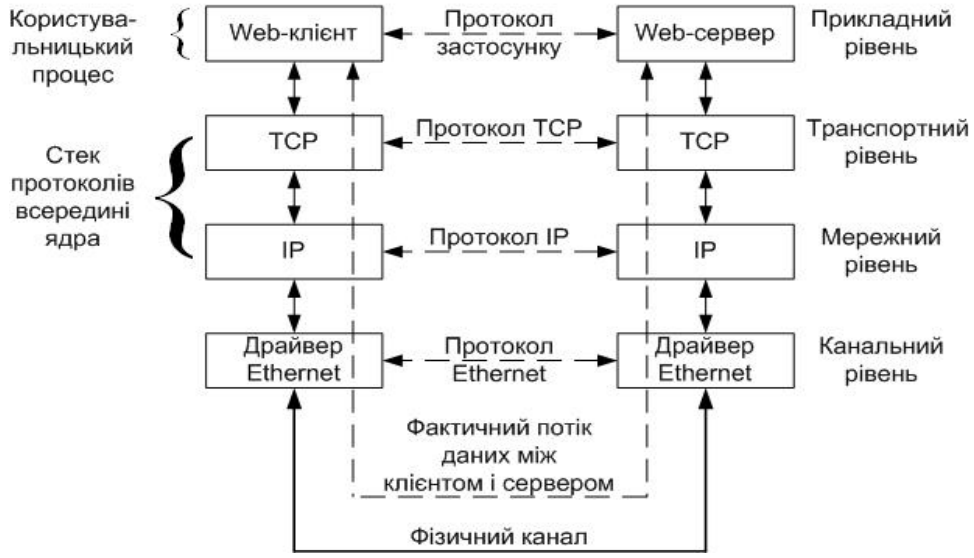

Рисунок 1.10 Мережна взаємодія, яка поділена на чотири рівні

Іншим типом протоколу є протокол віддаленого виклику процедур RPC, він складний, має велику кількість різновидів. Популярні протоколи високого рівня: CORBA,RMI і DCOM. Недоліки: є складними; для організації роботи потрібне спеціальне середовище як на стороні сервера, так і на стороні клієнта; виникають проблеми під час проходження пакетів даних через брандмауер.

Поширення набув протокол HTTP [6], був розроблений протокол SOAP, у якому для кодування запитів методів об'єктів і супутніх даних використовуються тексти мовою XML. SOAP є простим, легко реалізовується на багатьох пристроях, може працювати на верхньому рівні будь-якого стандартного протоколу, якщо на HTTP і протокол SMTP, то є змога пакетам даних проходити під час з'єднання через системи мережного захисту.

Тенденції розвитку клієнт-серверних систем приводять до переносу функціональності клієнта в спеціалізоване програмне забезпечення.

Розробка механізмів завантаження і виконання клієнтського програмного забезпечення та його взаємодії з серверами є актуальною науковою та прикладною задачею.

ВИСНОВКИ

В першому розділу дипломної роботи описані основи технології клієнт-сервер, вона вважається базою, на ній тримається сучасний світ комп'ютерних мереж. Але завдання, для вирішення яких вона була розроблена, стають історією. Удосконалення, що відбуваються, значно підвищують ефективність і функціональність, ставляться нові завдання і технології, що вимагають переосмислення принципів клієнт-серверних систем. Наприклад, технологія World Wide Web. Використання технології гіпертекстових документів для побудови внутрішньої інформаційної інфраструктури компанії стимулювало бурхливий розвиток систем: технологія Web є розвитком даної архітектури. Система Web має архітектуру клієнт-сервер, тому за допомогою одного клієнта можна підключитися до багатьох серверів. Web-браузер, який забезпечує зручний інтерфейс з користувачем для доступу до інформації, - це самий верхній рівень системи Web. Крім інтерфейсу будь-яка інформаційна система повинна мати рівні обробки даних і їх зберігання. У розробників мереж дуже часто виникає проблема узгодження роботи Web з іншими елементами системи (базами даних). Одним з перспективних способів вирішення цієї проблеми є багаторівневі архітектури клієнт-сервер. Подальший розвиток систем типу клієнт-сервер підтверджує актуальність досліджень.

2. ПРОТОКОЛИ ВЗАЄМОДІЇ ТА СХЕМИ ВЗАЄМОДІЇ ОСНОВНИХ МОДЕЛЕЙ КЛІЄНТ-СЕРВЕР І СЕРВІС-ОРІЄНТОВАНОЇ АРХІТЕКТУРИ ВЕБ-ДОДАТКІВ

2.1. Поняття протоколу взаємодії

Протоколом взаємодії називають зв'язок між компонентами, що здійснюється за певними правилами [21]. Існують три компоненти реалізації моделей взаємодії клієнт-сервер, засновані на поділі структури СКБД, це такі: введення і відображення даних; прикладний компонент; функції керування ресурсами.

Найпоширенішими є використання таких моделей взаємодії клієнт-сервер: модель FS - файлового серверу; модель RDA - віддаленого доступу до даних; модель DBS - сервера БД; модель AS - сервера додатків.

2.2 Схеми взаємодії основних моделей клієнт-сервер

Схеми взаємодії основних моделей клієнт-сервер поширено описані в [21].

2.2.1 Модель файлового серверу FS

На системах БД в задачах обробки інформації існують два варіанти розташування даних: локальний і віддалений.

Дистанційні дані відокремлені від комп'ютера користувачів - на FS мережі.

FS модель – це розширення персональних СКБД для підтримки багатокористувацького режиму. Особливості FS-моделі: всі основні компоненти розміщуються на клієнтському комп'ютері; модель характеризує загальний спосіб взаємодії комп'ютерів в локальній мережі; комп'ютер визначається сервером FS, який є загальним сховищем даних; він виконує пасивну функцію.

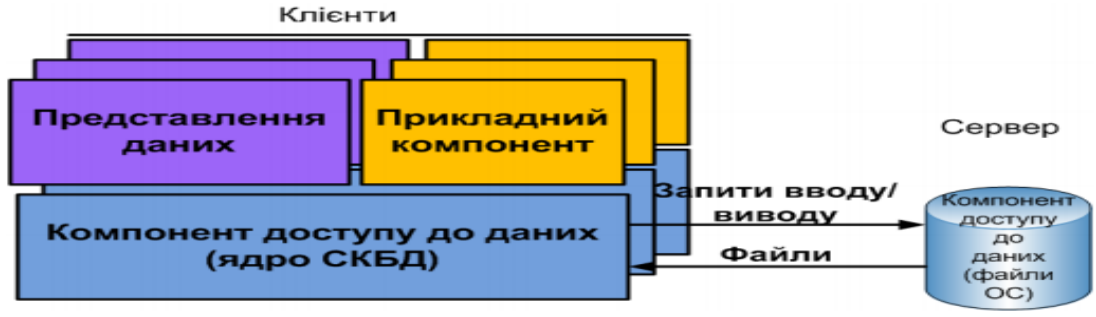

Рисунок 2.1 Схема взаємодії FS-моделі [21].

За отримання даних, обробку, підтримку цілісності БД відповідає додаток, запущений з робочої станції. При обробці даних по мережі проходить необхідна інформація, хоча потрібен значно менший обсяг даних, тому стало необхідним створення інших способів зменшення навантаження на мережу.

Додатки, що виконуються на робочих станціях, включають модулі для організації діалогу з користувачем, бізнес правила (транзакції), що забезпечують необхідну логіку обчислень, і ядро СКБД, яке в FS-моделі не є вираженим. Ядро СКБД являє собою набір функцій, пов'язаних з іншими компонентами додатка. Додаток дублюється на різних робочих станціях. На FS-моделі зберігаються тільки файли бази даних і деякі технологічні файли. Оператори SQL-звернення до СКБД, закодовані в прикладній програмі, обробляються ядром СКБД на робочій станції. СКБД організовує доступ до файлів БД для оператора. По мережі передаються запити на читання/запис даних, індекси, проміжні та результуючі дані, блоки технологічних файлів.

На основі FS-моделі функціонують СКБД - FoxPro (Microsoft), dBase (Borland), CF-Clipper (Computer Associates International), Paradox (Borland) тощо. Вказані СКБД недорогі, прості в установці та освоєнні, відсутні високі вимоги до продуктивності сервера, не розподілені програмні компоненти СКБД.

Недоліки: системи на базі СКБД мають низьку продуктивність - всі проміжні дані передаються по низькошвидкісній шині мережі, а прикладні програми і ядро СКБД виконуються на малопотужних робочих станціях; високий мережевий трафік; низька масштабованість; відсутність механізмів безпеки БД; СКБД не підтримують розподілену обробку. FS-модель майже не використовується в розподілених інформаційних системах.

2.2.2 Модель файлового серверу RDA

У моделі RDA для обробки даних є ядро - SQL-сервер, його функції - обробка запитів клієнтів. Додатки-клієнти посилають з робочих станцій запити на вибірку даних, а сервер здійснює відбір та відправляє тільки необхідні данні. Це зменшує навантаження на мережу та вимоги до комп'ютерів-клієнтів, та

підвищує надійність і збереження логічної цілісності БД. Додатки мають модулі для організації діалогу з користувачем і транзакції. Ядро СКБД є загальним для всіх робочих станцій і функціонує на сервері. Оператори звернення до СКБД закодовані в транзакції, пересилаються для обробки на сервер. Ядро СКБД транслює запит і виконує з звертанням до індексів та проміжних даних. На робочу станцію передаються тільки результати обробки оператора.

RDA-модель – заснована на обліку специфіки розміщення і фізичного маніпулювання даними у зовнішній пам'яті для реляційних СКБД (рисунок 2.2).

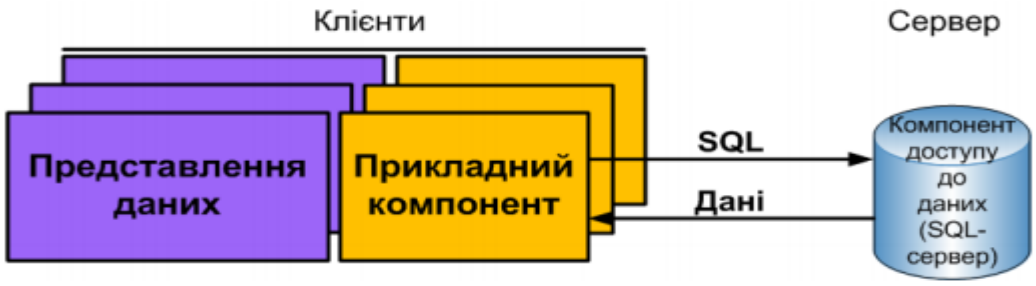

Рисунок 2.2 Схема взаємодії RDA-моделі

На сервері знаходиться системний каталог БД, де поміщенівідомості про зареєстрованих користувачів і привілеї користувачів. На клієнтських комп’ютерах встановлюються частини СКБД, які реалізують інтерфейсні і прикладні функції. Прикладний компонент має бібліотеки запитів, процедури обробки даних. Прикладний компонент розміщується на клієнтській частині - формує SQL-інструкції, що направляються SQLсерверу. SQL-сервер - програмний компонент, який орієнтований на інтерпретацію SQL-інструкцій, високошвидкісно виконує низькорівневі операції з даними, координує SQL-інструкції від клієнтів, виконує SQL-запити, перевіряє та забезпечує виконання обмежень цілісності даних, направляє результати обробки SQL-інструкцій.

Перевагами RDA-моделі є: зменшення завантаження мережі; забезпечення SQL-сервером виконання обмежень цілісності та безпеки даних; є уніфікований інтерфейс взаємодії прикладної частини ІС із загальними даними; в рамках SQL взаємодія реалізована через ODBC-протокол і називається інтероперабельністю.

Недоліками RDA-моделі є: високі вимоги до клієнтських комп'ютерів; великий обсяг технологічних змін; зміна структури управління та бізнес-процесів; значний мережевий трафік.

2.2.3 Модель сервера бази даних DBS

Для поліпшення RDA використовується DBS. На сервері запускаються DBS збережені процедури і тригери, які з ядром СКБД утворюють сервер DBS (рисунок 2.3).

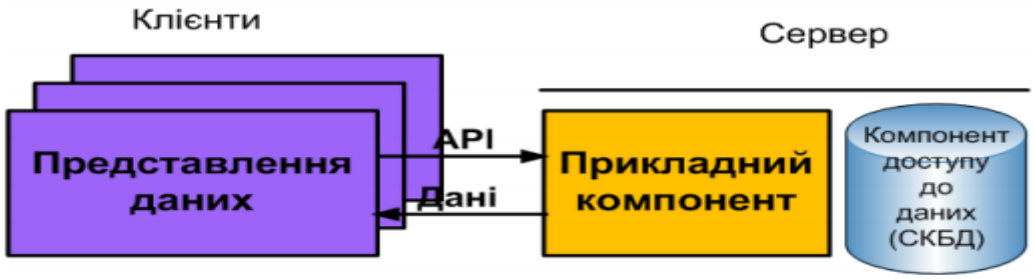

Рисунок 2.3 Схема взаємодії DBS-моделі

З додатків звертаються до збережених процедур, скорочуючи розмір коду прикладної програми і зменшуючи потік SQL-операторів, групу SQL-запитів кодується в збереженій процедурі. Автоматично підтримувати цілісність БД дозволяють тригери – програми (UPDATE, INSERT, DELETE) [21], які виконуються ядром СКБД перед/після оновлення БД. Модель сервера БД підтримують СКБД: Oracle, DB2 (IBM), MS SQL Server (Microsoft), MySQL (Oracle), FireBird, PostgreSQL, Sybase (SAP). Переваги СКБД: мають високу продуктивність системи - запити виконуються на високошвидкісних серверах; зниження мережевого трафіку - по шині передаються тільки SQL-запити і результати з виконання; підтримують розподілену обробку; гнучке настроювання на предметну область; забезпечення узгодженого стану даних; надійність зберігання і обробки даних; ефективна координація колективної роботи користувачів із загальними даними; пропонується багато сервісних продуктів, що полегшують розробку додатків і створення розподіленої системи.

Сервер БД (SQL-сервер) повертає клієнтському додатку тільки витискання переглянутого в базі, яка є малою частиною від загального обсягу. Мережа при

значної кількості клієнтів майже не навантажується. Додаткову обробку отриманих від сервера результатів запиту клієнтські комп'ютери не виконують.

Недоліки: набагато дорожчі СКБД попереднього класу, складні в освоєнні; потрібні дорогі високошвидкісні сервери та мережі.

Перехід в архітектуру клієнт-сервер гарантує збереження логічної цілісності БД, система є стійкою і захищеною. Сервери мають вбудовані механізми для захисту системи від помилкових дій клієнтів.

2.2.4 Модель сервера додатків AS

У моделі AS вимоги до обчислювальних ресурсів сервера у відношенні швидкодії і пам'яті розносяться за різними машинами. AS-модель зберігає сильні сторони DBS-моделі. Особливості AS-моделі: перенесення прикладного компонента AIC на спеціалізований сервер; на клієнтських машинах - тільки інтерфейсна частина системи; виклики функцій обробки даних спрямовуються на сервер додатків; низькорівневі операції з даними виконує SQL-сервер.

На рисунку 2.4 представлена схема взаємодії AS -моделі.

Рисунок 2.4 Схема взаємодії AS-моделі

Використання AS -моделі дозволяє розвантажити робочі станції - перейти до тонких клієнтів. Сервер додатків організовують за допомогою збережених процедур з використанням мови високого рівня (Oracle - мови PL/SQL). Можливості таких мов широкі: від циклів до обробки даних на рівні бітів. Збережені процедури не підтримують розподілені додатки, не забезпечують автоматичний запуск програми на іншому сервері. СКБД використовують процедурні мови: Transact-SQL (MS SQL Server, Sybase); PL/SQL (Oracle); PSQL (FireBird); SQL PL (IBM DB2); JetSQL (MS Access); PL/pgSQL (PostgreSQL).

Якщо інформаційна система об'єднує велику кількість різних інформаційних ресурсів і серверів додатків, то для оптимального управління всіма її компонентами використовують програмні засоби TPM, які розміщуються на сервері додатків. Менеджер транзакцій TPM і додатки можуть запускатися на одному комп'ютері, щоб зменшити потік SQL-запитів по мережі.

В моніторах транзакцій поняття транзакції трактується широко – це будь-яка дія в системі – видача повідомлення, запис в індексний файл, друк звіту тощо, але традиційно транзакцією [21] називають послідовну сукупність операцій над даними (SQLінструкцій), що має окреме смислове значення.

Для спілкування прикладної програми з монітором транзакцій використовується API у вигляді бібліотеки з викликами основних функцій: встановити з'єднання, викликати певний сервіс тощо.

Сервери AS (сервіси) створюються за допомогою API. TPM отримує запит від прикладної програми і передає її виклик сервісу, після обробки запиту AS - сервер повертає результати клієнту.

Для взаємодії TPM з серверами DBS розроблено протокол XA. Наявність цього інтерфейсу дозволяє використовувати в рамках однієї програми кілька різних СКБД. TPM усуває витрати спільної обробки: втрачені зміни; неузгоджені дані; неповторювані читання.

Використання TPM у великих системах має переваги: концентрація всіх прикладних функцій на AS – сервері, забезпечує значну незалежність і від реалізації інтерфейсу з користувачем і від конкретного способу керування ресурсами. Забезпечується централізоване адміністрування додатків; TPM сам запускає і зупиняє сервери додатків; забезпечується динамічна конфігурація системи; підвищується надійність системи; є можливість керування розподіленими БД. Недоліки AS-моделі: необхідні додаткові потужності; підвищується трафік в мережі.

2.3 Міжнародні стандарти протоколів

Найбільш важливі віхи в процесі забезпечення здібності до взаємодії розподілених програмних компонентів виділенні в роботі [21].

2.3.1 Програмні гнізда (sockets)

Автори [21] вважають, що першим кроком в процесі забезпечення здібності до взаємодії розподілених програмних компонентів був механізм програмних гнізд (sockets), розроблений в університеті Беркл.

Операційна система UNIX проектувалася як мережева ОС для забезпечення взаємодії процесів, які виконуються на різних комп'ютерах, з'єднаних мережею передачі даних. Можливість базувалася на забезпеченні файлового інтерфейсу для пристроїв на базі поняття спеціального файлу про домовленість щодо способу взаємодії на основі використання можливостей відповідних мережевих драйверів для процесів, розташованих на різних комп'ютерах [22]. Базові можливості були достатніми для створення мережевих утиліт; на їх основі був створений вихідний в ОС UNIX механізм мережевих взаємодій uucp.

Взаємодія процесів на основі програмних гнізд заснована на моделі клієнт-сервер. Сервер (процес) "слухає (listens)" своє програмне гніздо - одну з кінцевих точок двунаправленого шляху комунікацій, а клієнт (процес) спілкується з сервером через інше програмне гніздо. Ядро підтримує внутрішні з'єднання і маршрутизацію даних від клієнта до сервера.

Типи програмних гнізд: гнізда з віртуальним з'єднанням (stream sockets); датаграмні гнізда (datagram sockets).

При використанні програмних гнізд з віртуальним з'єднанням забезпечується передача даних від клієнта до сервера у вигляді безперервного потоку байтів з гарантією доставки. До початку передачі даних встановлене з'єднання, яке підтримується до кінця комунікаційної сесії. Датаграмні програмні гнізда не гарантують абсолютну надійну доставку повідомлень і не гарантують відсутність дублікатів пакетів даних - датаграм. Для використання датаграмного режиму не потрібне попереднє встановлення з'єднань.

Основним недоліком програмних гнізд [21,22] є те, що це чисто транспортний механізм, подання переданих по мережі даних є машинно-залежною формою.

Організація мережевих взаємодій користувацьких процесів не дуже ефективна тому, що при використанні конкретної мережної апаратури і конкретного мережевого протоколу потрібно виконувати багато системних викликів ioctl, це ставить в залежність програми від специфічного мережевого середовища. Потребується підтримуваний ядром механізм для приховання особливості середовища (в UNIX BSD 4.1 1982р.) і необхідність одноманітно взаємодіяти процесам, що виконується на одному комп'ютері, в межах локальної мережі/рознесеним на різні комп'ютери розподіленої мережі.

На рівні ядра механізм програмних гнізд підтримується компонентом рівня програмних гнізд, компонентом протокольного рівня і компонентом рівня управління мережним пристроєм [22]. Комбінації протоколів і драйверів задаються під час конфігурації, та їх міняти не можна при роботі системи.

2.3.2 Протокол віддаленого виклику процедур – RPC є важливим в процесі забезпечення здібності до взаємодії розподілених програмних компонентів.

Другим важливим [21] моментом стала поява протоколу віддаленого виклику процедур - RPC і його реалізація. Взаємодія програмних компонентів найчастіше є асиметричною, тому завжди можна виділити клієнта, якому потрібна послуга, і сервер, який її надає. Клієнт не може продовжувати своє виконання, поки сервер не виконає дію, яку від нього вимагають - типовою взаємодією клієнта та сервера є процедурна взаємодія. Спільним рішенням проблеми мобільності систем, заснованих на архітектурі клієнт-сервер є програмні пакети, що реалізують протоколи RPC. При використанні цих засобів, звернення до сервісу в віддаленому вузлі є викликом процедури. Виклик переводять в послідовність мережевих взаємодій, а специфіка мережного середовища і протоколів прихована від прикладного програміста. При виклику віддаленої процедури програми RPC виконується перетворення форматів даних клієнта в проміжні машинно-незалежні формати і потім перетворення у формати даних сервера. При передачі параметрів є аналогічні перетворенню. Система на основі пакета RPC, може переноситься у відкрите середовище. Звертання до сервера не відрізняється від виклику

локальної процедури. При реалізації виклику віддалених процедур на підставі специфікації інтерфейсу, на стороні клієнта генерується локальний представник віддаленої процедури - stub, а на стороні сервера - спеціалізований перехідник - proxy.

2.3.3 Визначення сервер баз даних DBS, поточні стандарти системи.

Визначення сервер БД використовують для позначення всієї СКБД, заснованої на архітектурі клієнт-сервер, включаючи і серверну, і клієнтську частини. Призначено систему для зберігання і забезпечення доступу до БД. Основним інтерфейсом між клієнтською і серверною частинами є мова БД SQL, це поточний стандарт інтерфейсу СКБД у відкритих системах. Назва SQL-сервер відноситься до всіх серверів БД, заснованих на SQL. Сервери DBS з стандартним інтерфейсом, та у яких клієнтські частини SQL-орієнтовані СКБД, працюють з будь-яким SQLсервером незалежно від виробника [21].

Перспективним напрямом СКБД є гнучке конфігурування системи.

2.4 Сервіс-орієнтована архітектура веб-застосувань SOA, концепції веб-служби і протоколи

2.4.1 До появи поняття WEB- сервісів існували технології взаємодії видалених застосувань, у яких одна програма може викликати який-небудь метод в іншій програмі, запущеній на видаленому комп'ютері. Це видалений виклик процедур RPC [24]. Формат обміну даними при моделі RPC (DCOM, CORBA) бінарний. Працювати з даними складніше і не дуже підходить, при організації роботи розподіленої системи. Технологія DCOM реалізована тільки для Windows-систем, CORBA же функціонує на різних платформах, але найбільш повноцінна її реалізація на J2EE. Є багато різних підходів, кожен з яких в чомусь є кращім ніж інші. Та єдине, на чому сходиться переважна більшість дослідників і розробників розподілених програмних систем, це схильність до використання парадигми об'єктної орієнтації [21]. Багато користувачів вважають перспективним підхід об'єктно-орієнтований стиль проектування та розробки програмних систем, який стимулює повторне

використання програмних компонентів, забезпечує незалежність використання інформаційних ресурсів від особливостей їх реалізації**,** а при проектуванні і розробці інформаційних систем дозволяє виробляти комплексну розробку структур даних і керуючих ними процедур із застосуванням технології об'єктноорієнтованих систем БД.

Але проблема є у відсутності загально об'єктної моделі. У різних об'єктно-орієнтованих системах програмування і системах БД набагато більше спільного, ніж різного, але їхні відмінності часто принципові настільки, що об'єкти, розроблені та створені в різних системах, не здатні взаємодіяти. Вони не інтероперабельні [21]. Виходом стала мінімальна об'єктна модель, що володіє обмеженими можливостями та має явні аналоги в найбільш поширених об'єктних системах. В архітектурі CORBA модель називається Core Object Model, і їй відповідає мова IDL. В архітектурі CORBA є компонент - ORB для передачі даних в машинонезалежному форматі.

Недоліком CORBA є те, що працює через свої мережеві протоколи, тому знайдеться така конфігурація мережі/платформ, що реалізація розподіленої системи буде дуже скрутна. Ще проблемою стандарту CORBA стала відсутність стандартизації взаємодій рівня ORB-ORB. У 1997 р. був прийняла новий стандарт CORBA-2, в якому є специфікований протокол взаємодій між ORB-ами. А стандарт CORBA 3.0, що використовується, прийнятий у 1999р. Це складна і незрозуміла проблема - семантична інтероперабельність об'єктних систем.

2.4.2 Постійне нарощування можливостей Інтернет надає можливість використання інформаційних ресурсів незалежно від їх реального розташування [21]. Інтернет є готовою платформою для створення і використання розподілених машинно-орієнтованих систем на основі веб-сервісів є [25].

Веб-сервер виступає сервером додатків, це дозволяє багаторазово використати функціональні елементи, усунути дублювання коду, спростити

рішення завдань інтеграції додатків. Веб-служба, веб-сервіс є мережевою технологією, що забезпечує міжпрограмну взаємодію на основі веб-стандартів.

Визначається веб-сервіс, як «програмна система, розроблена для підтримки інтероперабельної міжкомп'ютерної взаємодії через мережу»[25].

В роботі [19] звернена увага на два типи додатків Інтернет: обчислювальні вузли, котрі реалізують нетривіальні функції, і прикладні веб-ресурси. Веб-ресурси використовують послуги, що надають обчислювальні вузли. Для зв’язування і надання одним додаткам можливості обмінюватися даними з іншими, використовують веб-сервіси, з появою яких розвинулася ідея SOA — сервіс-орієнтованої архітектури веб-додатків. Веб-сервіси виступають серверами, постачальниками послуг, для інших сервісів, які виконують клієнтські функції [20]. SOA передбачає модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабо пов'язаних замінних компонентів, постачених стандартизованими інтерфейсами для взаємодії за стандартизованими протоколами.

Веб-сервіси — це реалізація чітких інтерфейсів обміну даними між різними додатками, котрі написані різними мовами і розподілені на різних вузлах мережі.

Програмні комплекси, розроблені відповідно до SOA, реалізуються як набір веб-служб, взаємодіючих по протоколу SOAP. Інтерфейси компонентів в SOA [19] інкапсулюють деталі реалізації від інших компонентів і забезпечують комбінування і багаторазове використання компонентів для побудови складних розподілених програмних комплексів, незалежно від платформ та інструментів розробки, сприяють масштабованості і керованості систем [24].

Існуючи протоколи взаємодії веб-сервісів відрізняються і контекстом, і використанням. Найбільш широко використовуються SOAP, XML-RPC та REST.

Веб-сервіс ідентифікується рядком URI [25]. Веб-сервіс має програмний інтерфейс, представлений в машинно-оброблюваному форматі WSDL. Інші системи взаємодіють з цим веб-сервісом шляхом обміну повідомленнями

протоколу SOAP. Транспортом для повідомлень є протокол HTTP. Опис веб-сервісів і їх API можуть бути знайдені засобами UDDI. Концептуальна схема технології приведена на рисунку 2.5, а зв'язок між протоколами - на рисунку 2.6.

Рис. 2.5 Концепція веб-сервісу

Усі специфікації, використовувані в технології, ґрунтовані на XML, вони наслідують його переваги: структурованість, гнучкість тощо, і недоліки: громіздкість, повільність [25].

Рис. 2.6 Протоколи веб-сервісів (реалізація веб-сервісів)

SOAP - простий протокол доступу до об'єктів, ґрунтований на обміні структурованими повідомленнями. SOAP використовується з будь-яким протоколом прикладного рівня: SMTP, FTP, HTTPS, але частіше усього використовується поверх HTTP.

2.4.5 XML-RPC - альтернатива SOAP XML - RPC - дуже простий і ефективний протокол взаємодії веб-сервісів. Він не призначений для вирішення глобальних завдань, як SOAP, але широко використовується у багатьох веб-розробках [25].

Рис. 2.7 Концепція XML-RPC

XML - RPC - це "... специфікація і набір реалізацій, які дозволяють програмному забезпеченню, працюючому на різних операційних системах і в різних умовах, викликати процедури через Інтернет. Це видалений виклик процедури з використанням HTTP як транспорту і XML як способу кодування. XML - RPC розроблений настільки простим, наскільки це можливо для складних структур даних, що підлягають передачі, обробці і прийому". - [xmlrpc.com] [25].

2.4.6 Реалізація веб-сервісів

Web-сервіси є програмними компонентами, що мають ідентифікатор URI, і взаємодія з якими здійснюється по Інтернету за допомогою відкритих протоколів. Комунікація з Web-сервісами може виконуватися за допомогою різних транспортних протоколів, таких як HTTP, HTTPS, FTP, SMTP, BEEP, при цьому Web - сервіси можна підрозділити на три види: SOAP Web- сервіси, орієнтовані на модель RPC, - виклик видалених процедур, XML Web-сервіси, орієнтовані на повідомлення, і RESTful Web-сервіси.

Програмна система Веб-сервіс має уніфікований ідентифікатор URI і загальнодоступні інтерфейси на мові XML. Опис програмної системи може бути знайдено іншими програмами, які взаємодіють з нею у відповідності з описом за допомогою повідомлень, заснованих на XML, і переданих інтернет-

протоколами. Веб-служба є одиницею модульності при використанні сервіс-орієнтованої архітектури програми [43]. Веб-сервіси є концепцією створення таких прикладних програм, функції яких можна використовувати за допомогою стандартних протоколів Інтернет. В даний час цю концепцію застосовують і розвивають багато провідних компаній в IT-галузі. Концепція веб-сервісів реалізується за допомогою ряду технологій: World Wide Web Consortium (W3C). Робота веб-сервісів побудована на використанні відкритих стандартів і технологій: XML - розширювана мова розмітки, призначена для зберігання і передачі структурованих даних; SOAP - протокол на базі XML; WSDL - мова опису зовнішніх інтерфейсів веб-сервісів на базі XML; UDDI - універсальний інтерфейс розпізнавання, опису та інтеграції (Universal Discovery, Description, and Integration). Каталог веб-сервісів і відомостей про компанії, що надають веб-сервіси в загальне користування або конкретним компаніям.

Взаємозв'язок цих технологій, реалізація веб-сервісів, умовно представлено на рисунку 2.6. Веб-сервіси це варіант реалізації компонентної архітектури. XML є фундаментом для створення більшості технологій, пов'язаних з веб-сервісами.

Для віддаленої взаємодії з веб-сервісами використовується SOAP, який забезпечує взаємодію розподілених систем, незалежно від об'єктної моделі, операційної системи або мови програмування. Дані передаються у вигляді особливих XML документів особливого формату.

За визначенням W3C, веб-сервіси це прикладні програми, які доступні стандартним протоколам Інтернет. Немає вимоги до того, щоб веб-сервіси використовували якийсь певний транспортний протокол. Специфікація SOAP визначає, яким чином зв'язуються повідомлення SOAP і транспортний протокол.

Найбільш часто реалізується передача SOAP повідомлень по протоколу HTTP, також поширене використання транспортних протоколів SMTP, FTP, TCP.

Документ WSDL повністю описує інтерфейс веб-сервісу, надає інформацію про послуги, які можна отримати з сервісу і способах звернення до цих методів.

Технологія UDDI припускає ведення реєстру веб-сервісів. Підключившись до цього реєстру, споживач зможе знайти веб-сервіси, які найкращим чином задовольняють його потребам. UDDI дає можливість пошуку і публікації потрібного сервісу, як людиною, так і програмою-клієнтом. Пошук і публікація в реєстрі надається програмі-клієнтові як набір веб-сервісів реєстру UDDI.

Веб-сервіси позиціонуються як програмне забезпечення проміжного шару.

Веб-сервіси розміщуються на серверах прикладних програм.

Веб-сервіси: забезпечують взаємодію програмних систем незалежно від платформи; засновані на базі відкритих стандартів і протоколів; використання HTTP дозволяє додаткам взаємодіяти через міжмережевий екран.

Веб-сервіси мають недоліки: менша продуктивність і більший об'єм мережевого трафіку в порівнянні з такими технологіями як CORBA або DCOM.

Web-сервіси покликані погоджувати роботу великих застосувань, що складаються з безлічі частин, надаючи для додатків бізнес-функції обміну даними. Окрім функції обміну даними між різними застосуваннями і платформами, Web-сервіси можуть виступати як повторно-використовувані компоненти додатки, що надають різноманітні сервіси.

2.4.7 Мова опису веб-сервісів WSDL призначена для уніфікованого представлення зовнішніх інтерфейсів веб-служби. Версія протоколу WSDL 2.0 має деякі відмінності від попередніх версій.

У специфікації WSDL 1.1 було визначене 4 шаблони обміну повідомленнями: однонапрямлені операції (One - way): операція може приймати повідомлення, але не повертатиме відповідь; запит-відповідь (Request - response): операція може прийняти запит і повинна повернути відповідь; питання-відповідь (Solicit - response): операція може послати запит і чекатиме відповідь на нього; сповіщення (Notification): операція може послати повідомлення, але не чекатиме відповідь.

У версії WSDL 2.0 ці шаблони змінені і розширені у бік підтримки повідомлень про помилки, але для зворотної сумісності підтримуються типи WSDL 1.1.

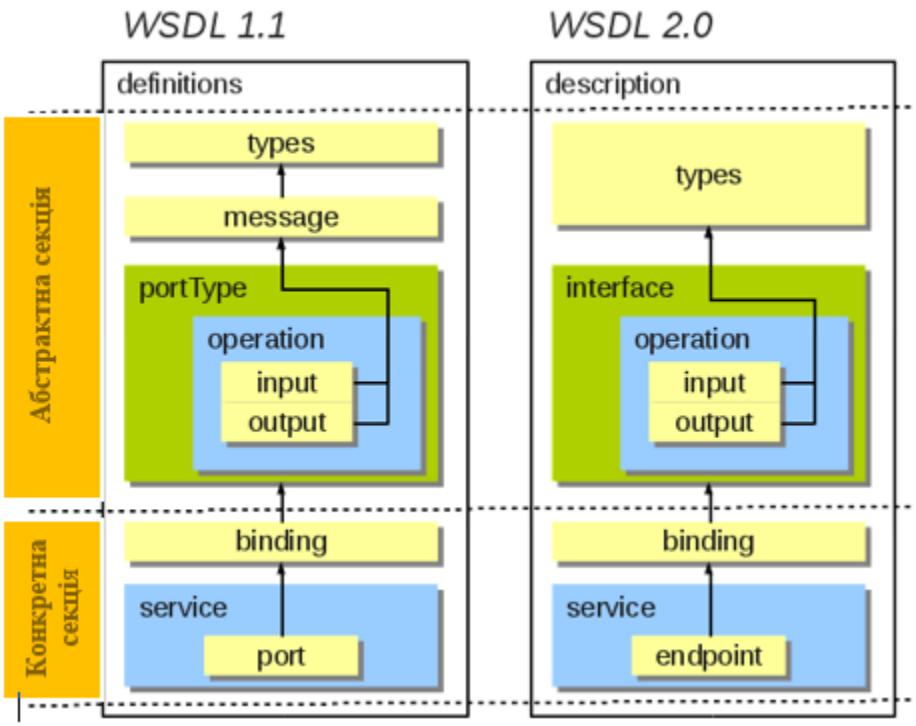

Рисунок 2.8 Структура протоколу WSDL

2.4.8 UDDI, універсальний інтерфейс розпізнавання, опису і інтеграції - відкритий стандарт, затверджений OASIS, що визначає методи публікації і виявлення мережевих програмних компонентів сервіс-орієнтованої архітектури (SOA). У практичній реалізації UDDI є мережевим реєстром, що представляє дані і метадані про веб-сервіси і доступний за адресою, http://uddi.xml.org/services.

UDDI спирається на галузеві стандарти HTTP, XML, XML Schema (XSD), SOAP і WSDL. Концептуальний зв'язок між UDDI і іншими протоколами стека веб-сервісів показаний на рисунку 2.9.

Рисунок 2. 9 Місце UDDI в стеку протоколів веб-служб

Функціональне призначення реєстру UDDI - уявлення даних і метаданих про веб-служби. Він може використовуватися як в мережі загального користування, так і в межах внутрішньої інфраструктури організації. Реєстр UDDI пропонує ґрунтований на стандартах механізм класифікації, каталогізації і управління веб-службами, що дозволяє наділяє веб-сервіси іншими застосуваннями. Цей механізм включає засоби для пошуку сервісу, виклику цієї служби, а також для управління метаданими про цю службу. Ключовими функціями UDDI є публікація інформації про службу в реєстрі і пошук цієї інформації сторонніми застосуваннями. Разом з цими, реалізовані і типові для служби каталогів функції: представлення моделі даних, що зберігаються, і структури інформаційної бази, стосунки між об'єктами реєстру, реплікація, забезпечення безпеки і так далі. Основні функції реєстру у вигляді веб-сервісів і доступні через API UDDI.

2.5 Технології об’єктного зв’язку даних DOA

Уніфікація взаємодії прикладних компонентів з ядром ІС для клієнт-серверних систем дозволила виробити аналогічні рішення і для інтеграції розрізнених локальних БД під керуванням СКБД в складні децентралізовані гетерогенні розподілені системи - об'єктне зв'язування даних [21].

Технологія об'єктного зв'язування вирішує завдання забезпечення доступу з однієї локальної БД, відкритої одним користувачем, до даних в іншій локальної БД, можливо знаходиться на іншій машині та відкритої іншим користувачем.

Об'єктний доступ до даних DOA: «настільні» СКБД підтримують технологію «об'єктного доступу до даних» - DAO; технічно DAO заснована на протоколі ODBC (рисунок 2.10) [21], який прийнятий як стандарт для доступу до даних на SQL-серверах і до будь-яких даних під керуванням реляційних СКБД - ODBC-драйвера; сучасні «настільні» СКБД забезпечують прямий доступ до об'єктів зовнішніх БД «своїх» форматів - зв'язані об'єкти; для доступу до БД найбільш поширених форматів і файлів електронних таблиць - ISAM драйвера.

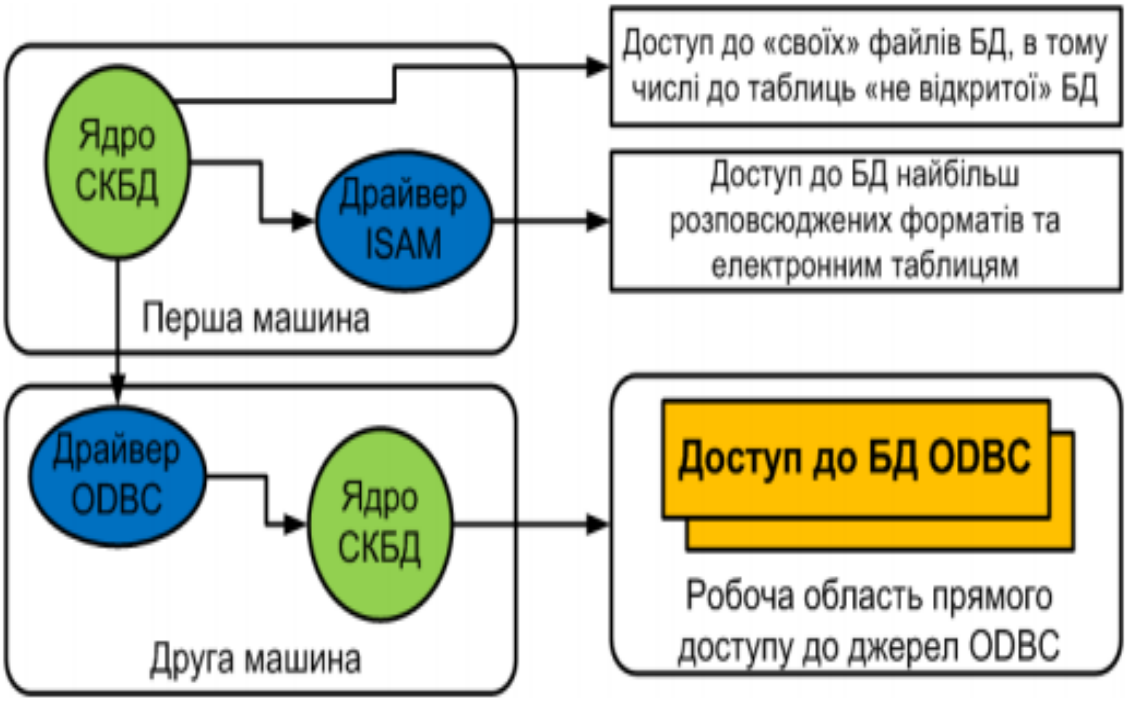

Рис. 2.10 Принцип доступу до даних через ODBC

Недоліки зв’язку даних: при великих обсягах даних у зв'язаних таблицях мережевий трафік істотно збільшується; відсутність надійних механізмів безпеки даних; відсутність надійних механізмів забезпечення обмежень цілісності.

2.6 Технологічні особливості використання баз даних в мережах[16].

Фінансові установи використовують в своїй роботі інформацію, доступ до якої корисний не тільки внутрішнім користувачам, та й великій кількості зовнішніх користувачів, що зв'язані з даною організацією. Їхні проблеми вирішуються з допомогою створення та використання локальних і глобальних мереж передачі даних, а також надання відповідного доступу до баз даних користувачам [16].

Зазвичай такі проблеми вирішуються з допомогою телекомунікаційних технологій та технологій клієнт-сервер.

Поширеними та ефективними є сервери БД: Oracle, Informix, Sybase, Interbase тощо. Сервер БД створюється на робочому місці адміністратора БД, а клієнти отримують доступ до таблиць, згідно з посадовими обов'язками та статусом [16].

Технологія ODBC забезпечує можливість доступу до віддалених БД з допомогою драйвера. Драйвер ODBC використовується інтерфейсом для отримання доступу до віддаленої БД, шляхом передачі запиту до БД і повернення результату його виконання. ODBC - є стандартом, ця технологія використовується виробниками систем [16]. Технологія ODBC забезпечує доступ до віддалених БД з допомогою драйвера, який використовується інтерфейсною частиною, шляхом забезпечення передачі запиту до БД і повернення результату. ODBC - використовується виробниками програмних продуктів. Виробники систем управління БД теж враховують цей стандарт та залишають відкритий інтерфейс для ODBC. Процес взаємодії прикладної та інтерфейсової частин зображено на рисунку 2.11

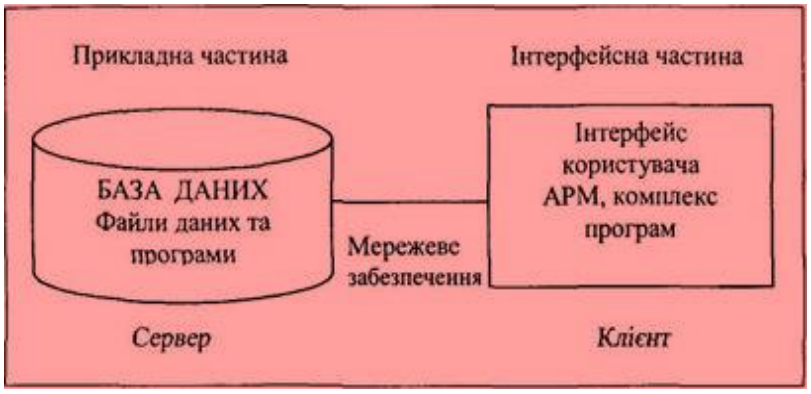

Рисунок 2.11. Схема взаємодії прикладної та інтерфейсної частин.

Виробники СУБД пропонують засоби підключення до віддалених БД. Для цього використовують продукт корпорації Oracle Net8 з будь-яким мережевим протоколом (TCP/IP, OSI, SPX/IPX тощо), він може працювати під управлінням поширених операційних систем, також доступ до віддалених БД може здійснюватися за допомогою Web-технологій.

ВИСНОВКИ

В другому розділі дипломної роботи описані схеми взаємодії основних моделей клієнт-сервер, сервіс-орієнтована архітектура веб-застосувань SOA, концепції веб- служби і протоколи, міжнародні стандарти протоколів та досліджені протоколи взаємодії клієнтів з сервером повідомлень. Результати досліджень показали актуальність роботи. Багато дослідників сьогодні намагаються вирішити проблему щодо переведення на нову клієнт-серверну технологію з новими принципами взаємодії, та виробничий процес не дозволяє припинити або призупинити використання морально застарілих інформаційних систем, що стає перешкодою для поширення нових технологій. Серйозність проблеми виникає і тому, що інформаційні системи і БД, які лежать в їх основі, є дорогими продуктами, щоб можна було переробити при зміні апаратної платформи/системного програмного забезпечення. Вимоги до програмного продукту є гнучкість - надання властивості легкого переставлення з одної апаратно-програмної платформи на іншу, та при переносі можуть знадобитися якісь зміни у вихідних текстах, головне, щоб не було корінної переробки системи. Крім того, розвиток інформаційних систем можливий за рахунок включення додаткових програмних та інформаційних компонентів. Задовольнити ці вимоги можна вже на стадії проектування та розробки інформаційної системи шляхом використання принципів «Відкритих Систем» і відповідних міжнародних або фактичних загальновизнаних стандартів у всіх необхідних видах забезпечення. Технології та стандарти забезпечують реальну і перевірену світовою практикою можливість виробництва системних і прикладних програмних засобів з бажаними властивостями.

3. АНАЛІЗ ПРОТОКОЛІВ ВЗАЄМОДІЇ КЛІЄНТІВ З СЕРВЕРОМ ПОВІДОМЛЕНЬ

3.1 Класи клієнт-серверних додатків. Технологія ZeroMQ

3.1.1.Схеми основних класів додатків

В загальній структури додатків клієнт-сервер, робота, може бути розділена між клієнтом і сервером по-різному. Частка операцій і обсяг даних, що передані по мережі, залежать від природи інформації, яка міститься в БД, типів додатків, доступності обладнання, від характеру використання даних [20].

Схеми основних класів додатків показані на рисунках 3.1.

Рисунок 3.1а)
Обробка даних на базі хоста

Рисунок 3.1б)
Обробка даних на базі сервера

Рисунок 3.1в)
Обробка даних на базі клієнта

Рисунок 3.1г)
Спільна обробка даних

Обробка даних на базі хоста, рисунок 3.1а). Схема не є додатком клієнт-серверу, відноситься до мейнфрейма; вся обробка даних здійснюється на головній обчислювальної машині. В обчислювальному середовищі інтерфейс користувача у вигляді примітивного терміналу.

Обробка даних на базі сервера, рисунок 3.1б). Найпростішим класом конфігурації клієнт-сервер є схема, в якій клієнт відповідає лише за надання графічного інтерфейсу користувача, тоді вся обробка даних здійснюється на сервері. Це дозволяє знизити навантаження на серверну частину.

Обробка даних на базі клієнта, рисунок 3.1в). Схема демонструє зворотній підхід. Вся обробка даних здійснюється на стороні клієнта, крім процедур перевірки цілісності даних та іншої логіки, що відноситься до обслуговування БД, які краще виконувати на сервері. Складні функції для роботи з БД розташовуються на стороні клієнта. Реалізації схеми є найбільш поширеними реалізаціями архітектури клієнт-сервер.

Спільна обробка даних, рисунок 3.1г). У даній конфігурації обробка даних оптимізована так, щоб використовувати сильні сторони і клієнта, і сервера, і самого факту розподілу даних. Такі зміни набагато складніші в установці і обслуговуванні. Вони забезпечують кращі показники продуктивності та ефективності використання мережевих ресурсів, ніж інші методи.

Можливі і інші варіанти розподілу задач між сервером і клієнтом.

3.1.2. ZeroMQ

Розвитком парадигми сокетних з'єднань за рамками моделі взаємодії клієнт-сервер є технологія ZeroMQ, яка надає вдосконалений інтерфейс сокетів з підтримкою більшої кількості протоколів взаємодії, а також з підтримкою інших схем роботи: публікація-підписка (pub-sub); тягни-штовхай (push-pull); дилер-маршрутизатор (dealer-router); екскюзівная пара; схема запит-відповідь (req-rep) - це класична клієнт-серверна схема з'єднання) [36].

3.2 Моделі взаємодії клієнта і сервера, протоколи взаємодії

3.2.1 Аналіз протоколів взаємодії клієнтів з сервером повідомлень свідчить про актуальність досліджень. В технічній літературі серед найпоширеніших запитань і досліджень визначено API, REST, XML-RPC та SOAP [7-34]. Треба обирати, коли використати SOAP, а коли використати REST/RESTful чи інші стилі або моделі тощо. Якій з них є кращим, якщо йдеться мова про продуктивність / швидкість або обробку запитів.

Відомий розвиток в галузі досліджень протоколів взаємодії стосовно SOAP-протоколу, який утворився від XML-RPC, а зараз опинився на наступному етапі його розвитку тощо. Концепція REST, в основі якої лежить архітектурний стиль ніж нова технологія, базується на теорії маніпуляції об'єктами CRUD в контексті концепцій WWW. Існують і інші протоколи, але вважається, що вони не набули широкого поширення, то їм в літературі приділяється найменша увага.

Найчастіше здійснюється огляд і порівняння таких: SOAP і REST, XML-RPC і SOAP. Та при цьому XML-RPC вважають «застарілим».

3.2.2 Що таке SOAP

Стандарт SOAP [29,34] - протокол взаємодії сервісів, протокол обміну структурованими повідомленнями в розподіленім обчислювальнім середовищі, створений в 1998р.

Розробки у сфері комп'ютерних мереж поступово популяризували концепції комп'ютерних мереж LAN, MAN та WAN, даючи можливість спілкуватися комп'ютерам, що фізично знаходяться у різних місцях. Протоколи веб-сервісів дали можливість комп'ютерам взаємодіяти на однакових платформах, технологіях та операційних системах. Потреба у протоколі, незалежному від цих обмежень породила протокол SOAP, котрий використовує XML для спілкування поверх транспортного рівня [19,20, 23].

Спочатку SOAP призначався для реалізації видаленого виклику процедур RPC. SOAP до версії 1.2 був протоколом обміну структурованими повідомленнями в розподіленому обчислювальному середовищі [29,34].

Остання версія стандарту SOAP специфікації 1.2 використовується для обміну довільними повідомленнями у форматі XML, а не тільки для виклику процедур. SOAP є розширенням протоколу XML - RPC. Мова для опису веб-сервісів WSDL використовується разом з SOAP, щоб зробити повідомлення доступними у Web за допомогою веб-сервісів [19]. SOAP сумісно з WSDL називається SOAP Web services. Протокол SOAP не розрізняє виклик процедури і відповідь на нього, а визначає формат послання (message) у вигляді XML-

документа. Послання може містити виклик процедури, відповідь на нього, запит на виконання якихось інших дій або просто текст. SOAP задає оформлення послання. Протокол SOAP заснований на XML, розширює протоколи прикладного рівня HTTP, FTP, SMTP тощо. Найчастіше використовується HTTP. Замість використання HTTP-запиту HTML-сторінки, яка буде показана в браузері, SOAP відправляє за допомогою HTTP-запиту XML-повідомлення і отримує результат у HTTP-відгук. Для обробки XML-повідомлень процес-«слухач» HTTP надає SOAP-процесору, який матє можливість обробляти XML. SOAP є найголовнішою частиною технології Web-сервісів. Дані переносяться по мережі, забезпечується доставка даних веб-сервісів, що дозволяє відправнику і одержувачу XML-документів підтримувати протокол передачі даних, забезпечує ефективність зв'язку.

SOAP – це базова односпрямована модель з'єднання, що забезпечує узгоджену передачу повідомлення, допускає наявність посередників, які можуть обробляти частину повідомлення або додавати до нього додаткові елементи. Специфікація SOAP містить угоди по перетворенню односпрямованого обміну повідомленнями в відповідності з принципом запит-відповідь, а також визначає як здійснювати передачу XML-документа. SOAP призначений для підтримки незалежного абстрактного протоколу зв'язку, який забезпечує комунікацію додатків, сайтів, реалізованих на різних технологіях і апаратних засобах.

SOAP може використовуватися з будь-яким протоколом рівня: SMTR, FTR, HTTP, HTTPS та ін. Але його взаємодія з кожним з цих протоколів має свої особливості, які визначені окремо. Частіше SOAP використовується поверх HTTP [29, 31], але може використовувати інші протоколи передачі даних.

SOAP характеризується: протокольною і мовною незалежністю, незалежністю від ОС і платформи та підтримкою SOAP XML – повідомлень взаємодіючих частин (використовуючи багатоскладну структуру MIME).

Призначення SOAP - надати мінімальний транспортний функціонал, поверх якого можуть бути побудовані більш складні протоколи та процеси взаємодії.

Для розробки клієнта треба знати URL кінцевої точки сервера, яка відповідає за обробку повідомлень. Це стає полегшенням з використанням WSDL, форматом XML схеми, який надає можливість представити мережеві сервіси у вигляді множини кінцевих точок, що обробляють повідомлення [19,20, 23, 34].

Протокол був стандартизований консорціумом W3C у 2003 році, використовується для представлення бізнес даних через інтерфейси і сервіси, що обмінюються цілими документами чи відображають дані на об'єкт, використовуючи назви методів, вхідні та вихідні параметри [19,20].

SOAP має перевагу (елемент attachment) — надсилання прикріплених файлів, що вважається найкращим рішенням для сервісів, які мають справу з прикріпленими файлами. Він незалежний від HTTP.

Архитектура клієнт-сервер є разноманітною.

Через свою надійність та рівень безпеки протоколи SOAP широко використовуються у банківських системах та корпоративних додатках.

3.2.3 Що таке REST

REST - передача стану представництва. REST - архітектурний стиль взаємодії компонентів розподіленого застосування в мережі [19,20,23]. REST - це архітектура для розробки веб-сервісів, архітектури яких використовують HTTP чи схожі протоколи, шляхом обмеження інтерфейсів набором стандартних операцій (GET, PUT/ PATCH, POST та DELETE для HTTP).

REST є погодженим набором обмежень, що враховують при проектуванні розподіленої гипермедіасистеми, у певних випадках підвищується продуктивність і спрощення архітектури. У широкому сенсі компоненти в REST взаємодіють подібно взаємодії клієнтів серверів в Інтернет. REST є альтернативою RPC. У мережі Інтернет виклик видаленої процедури може бути звичайним HTTP - запитом, зазвичай GET чи POST; а запит називають REST - запит, необхідні дані передаються в якості параметрів запиту.

Взаємодії з ресурсами мають збережений стан, а не з повідомленнями та операціями. Ця архітектура показує, що існуючого HTTP [19, 30, 31]. достатньо для побудови веб-сервісу, та демонструє масштабування.

Численні інфраструктури для REST інтегрують в стандарт Java ™ 6 JSR-311.

Головною ідеєю REST було використання розвиненого HTTP для передачі даних між пристроями, ніж використання протоколу, побудованого поверх рівня HTTP, для передачі повідомлень. Додаток, розроблений за принципами даного архітектурного підходу, використовує HTTP для виконання викликів між пристроями не спираючись на складні об'єктні механізми CORBA, RPC чи SOAP.

REST-додатки використовують функції HTTP запиту для надсилання, зчитування та видалення даних, використовуючи повну функціональність HTTP CRUD (Create, Read, Update і Delete) операцій. До того ж REST може використовувати HTTPS, надаючи безпечне передавання даних.

Для веб сервісів, побудованих з урахуванням REST, що не порушують обмежень, які накладаються їм, застосовують термін RESTful. Для створення RESTful сервісів використовують HTTP методи. [31].

У відповідності з технічними умовам HTTP, GET (також як і HEAD) запити використовуються тільки для читання даних без змін, тому при дотриманні даної угоди, вони вважаються безпечними. GET та HEAD запити є ідемпотентними, це означає отримання ідентичних даних при використанні одних і тих же запитів.

Використовувати GET для небезпечних операцій над даними не варто, запити не повинні бути модифікованими [32].

HTTP методи класифікують, як безпечні, які ніколи не змінюють ресурси, й небезпечні (метод GET).

Ідемпотентні методи, що досягають того ж результату, незалежно від того, скільки разів запит повторюється: це GET, PUT і DELETE. Вони можуть бути несподіваними, тому що повторення методу PUT з точно таким же body модифікує ресурс таким чином, щоб він залишається ідентичним в попередньому запиті PUT. Аналогічним чином, немає сенсу двічі видаляти ресурс. Незалежно від того, скільки разів запит PUT або DELETE повторюється, результат є таким же, як якщо б це було зроблено тільки один раз.

У реалізаціях HTTP немає нічого, що автоматично призведе до створення ресурсів, їх перерахування, видалення або оновлення.

Представництва: HTTP-клієнт і HTTP-сервер обмінюються інформацією про ресурси, визначених URL-адресами. Можна відправляти текст HTML і XML будь-якою мовою. Форми HTML не підходять для рахунку, тому що вони дозволяють робити тільки запити GET і POST. Та API-інтерфейси програмно доступні через окрему клієнтську програму або через JavaScript в браузері.

На відміну від SOAP, REST принципи легкі для розуміння та застосування розробнику, котрий знайомий з використанням HTTP. CRUD операції разом з функціями HTTP REST та відповідними SQL операціями показані у таблиці 3.1.

Таблиця 3.1 Відповідність HTTP-функцій SQL-операціям

CRUD операції	Ключові слова REST (HTTP)	Оператори SQL
CREATE- створити чи додати нові сутності	POST	INSERT
UPDATE-оновити чи редагувати існуючі дані	PUT	UPDATE
READ-зчитати, отримати дані	GET	SELECT
DELETE=видалити існуючи дані	DELETE	DELETE

Технологія REST - це спосіб проектування Web-сервісів, менш залежний від закритого проміжного програмного забезпечення ніж моделі SOAP і WSDL. REST, завдяки уваги на ранні Інтернет-стандарти URI і HTTP, є поверненням до Web, до появи великих серверів додатків [19, 31].

Для задоволення постійно зростаючих вимог до продуктивності, Web-сервіси REST повинні бути масштабованими. Для формування топології сервісів, що дозволяє при необхідності направляти запити з одного сервера на інший з метою зменшення загального часу реакції на виклик Web-сервісу, застосовують кластери серверів для розподілу навантаження і аварійного перемикання на резерв**,** проксі-сервери і шлюзи. Використання проміжних серверів для покращення масштабованості вимагає, щоб клієнти Web-сервісів REST відправляли повні самодостатні запити, що містять усе необхідне для їхього виконання, щоб компоненти на проміжних серверах могли перенаправляти, маршрутизувати і розподіляти навантаження без локального збереження стану між запитами. При обробці повного запиту серверу не потрібно витягувати стан або контекст програми.

В сервісі, що зберігає стан, програма може запросити наступну сторінку в багатосторінковому наборі результатів, вважаючи, що сервер зберігає послідовність переходів програми з цього набору результатів. У цій моделі зі збереженням стану сервіс нарощує і зберігає змінну previousPage, щоб бути в змозі відповідати на запити наступних сторінок.

Подібні сервіси, що зберігають стан, є складними. На платформі Java EE (Java Platform, Enterprise Edition) такі сервіси вимагають великої кількості попередніх угод щодо ефективного зберігання і синхронізації сеансових даних в кластері Java EE-контейнерів. Синхронізація сеансів збільшує накладні витрати, що негативно позначається на продуктивності сервера.

На противагу цьому, серверні компоненти, які не зберігають стан, більш прості. Цей сервіс покладає основну відповідальність за збереження стану на клієнтський додаток. У Web-сервісі RESTful сервер відповідає за генерування відповідей і за надання інтерфейсу, що дозволяє клієнтського додатка самому зберігати свій стан.

Web-сервіс, що не зберігає стану, генерує відповідь, де містить посилання на номер сторінки в наборі і дозволяє клієнтові самостійно подбати про збереження цього значення. Цей аспект моделі Web-сервісів RESTful можна

розділити на дві сфери відповідальності, що пояснюють суть функціонування вказаного сервісу:

Сервер генерує відповіді, які містять посилання на інші ресурси для навігації додатків, пов'язаних ресурсів. Такі відповіді містять вбудовані посилання. Аналогічна при запиті родового або контейнерного ресурсу типовий RESTful-відповідь може містити посилання на попередників родового елемента або на підлеглі ресурси, щоб зберігати зв'язок з ними.

Сервер генерує відповіді, які містять інформацію про те, чи підлягають вони кешуванню з метою підвищення продуктивності за рахунок зменшення кількості запитів дубльованих ресурсів і повної відмови від деяких запитів.

Клієнтський додаток відправляє повні запити, які можуть оброблятися незалежно від інших запитів. Це вимагає від клієнтського додатка використання в повному обсязі HTTP-заголовків, визначених інтерфейсом Web-сервісу, і відправлення повних уявлень ресурсів в тілі запиту. Клієнтський додаток відправляє запити, які практично нічого не знають про попередні запити, про існування сеансу на сервері, про здатність сервера додавати контекст запиту про стан програми, що зберігається між запитами.

Спільна робота клієнтського додатка і сервісу є важливою для стану відмови від збереження в Web-сервісах RESTful. Результатом є зростання продуктивності за рахунок зменшення трафіку і мінімізації стану серверного додатка.

Подання системних ресурсів через RESTful API - це гнучкий спосіб забезпечення різних додатків даними в стандартному форматі. Це допомагає виконати вимоги до інтеграції, що мають вирішальне значення для створення систем, що дозволяють легко комбінувати дані, а також для побудови на основі набору базових RESTful-сервісів більших конструкцій. Інформація, що широко публікується з цього питання підштовхує до продовження вивчення цієї теми. З розгляду принципів проектування RESTful-інтерфейсу, XML поверх HTTP є потужним інтерфейсом, що дозволяє внутрішнім додаткам (наприклад, призначеним для користувача інтерфейсів, заснованим на технології Ajax

(Asynchronous JavaScript + XML)), легко підключатися і звертатися до ресурсів і споживати їх. Фактично саме хороша сумісність Ajax і REST стала причиною сьогоднішньої популярності REST.

3.2.4 Що таке XML-RPC

3.2.4.1 RPC і мережеві архітектури [36].

Розподілена програма використовує для взаємодії певний протокол, який також можна розглядати як інтерфейс виклику процедур віддалено (RPC — remote procedure call). Фактично, інтерфейс RPC реалізує прикладний рівень моделі OSI, але для його підтримки також необхідно в тій чи іншій мірі реалізувати протоколи рівня представлення та сеансового рівня. Рівень представлення вирішує задачу передачі даних в рамках гетерогенної (тобто складеної з різних компонентів) мережі в "зрозумілій" формі. Для цього потрібно враховувати такі аспекти, як старшинство байт (endianness), кодування для текстових даних, подання композитних даних (колекцій, структур) і т.д. Ще одним завданням RPC-рівня є знаходження сервісів (service discovery). Реалізація RPC може бути заснована на власному (ad hoc) або ж якомусь із стандартних протоколів прикладного рівня і представлення. Наприклад, реалізація RPC по методолгіі REST використовує стандартні протоколи HTTP в якості транспортного і JSON/XML для представлення (сериалізації). XML/RPC або JSON/RPC - це ad hoc RPC, які використовує XML або JSON для представлення даних. Протоколи ASN.1 і Thrift - це бінарні протоколи, які визначають реалізацію всіх 3-х рівнів. У форматів серіалізації існує 2 дихотомії: бінарні і текстові формати, а також статичні (що використовують схему) і динамічні (без схеми, schema-less). Поширені формати серіалізації включають: JSON - текстовий динамічний формат XML - тестовий формат з опціональною схемою Protocol Buffers - бінарний статичний формат MessagePack - заснований на JSON бінарний формат Avro - заснований на JSON формат зі схемою EDN (extensible data notation) - текстовий динамічний формат Мережеві програми можуть бути реалізовані у вигляді різних мережевих архітектур. Ключовим параметром для кожної архітектури є рівень

централізації: від повністю централізованих — клієнт-сервер — до повністю децентралізованих - peer-to-peer/P2P. Важливими моделями між цими двома крайнощами є модель сервісно-орієнтованої архітектури (SOA), а також модель клієнт-черга-клієнт.

3.2.4.2 RPC - віддалений виклик процедур за допомогою XML [30].

Сама методика віддаленого виклику процедури відома давно і використовується в таких технологіях, як DCOM, SOAP, CORBA. RPC призначений для побудови розподілених клієнт-серверних додатків. Це дає можливість будувати додатки, які працюють в гетерогенних мережах. [30].

Згідно з цими технологіями одна програма може викликати будь-якої метод в іншій програмі, запущеної на віддаленому комп'ютері. Скорочено це називається віддалений виклик процедур RPC (Remote Procedure Calling). Однак формат обміну даними при класичній моделі RPC (DCOM, CORBA) бінарний [24].

А отже, працювати з даними складніше і він не дуже підходить, якщо треба організувати роботу розподіленої системи, де між окремими ділянками мережі стоять firewall / проксі-сервери. Технологія DCOM реалізована тільки для Windows-систем, CORBA же функціонує на різних платформах, але найбільш повноцінна її реалізація на J2EE. Недоліком CORBA є те, що вона працює через якісь свої мережеві протоколи замість простого HTTP, який проходить через firewall. Значить, завжди знайдеться така конфігурація мережі / платформ, що реалізація розподіленої системи буде дуже важким.

Концепція WEB-сервісу пов'язана зі створенням такого RPC, який можна включити в HTTP пакети. Таким чином почалася розробка стандарту, що визначає процес взаємодії додатків по протоколу HTTP, щоб додатки могли функціонувати на різних апаратно-програмних платформах і використовувати різні типи технологій і мови розробки. З появою WEB-сервісів почався розвиток сервіс-орієнтованої архітектури веб-додатків SOA. Найбільшого поширення набули такі протоколи реалізації WEB-сервісів: XML-RPC XML-виклик віддалених процедур; XML-RPC

Протокол WEB-сервісу, розроблений Дейвом Вінером з компанії «UserLand Software» у співпраці з Майкрософт в 1998 році. В ньому використовується в якості транспорту механізму протокол HTTP і як формат переданих даних XML. Це знімає обмеження, що накладаються як на конфігурацію мережі, так і на маршрут проходження пакетів. Виклики XML-RPC є простий тип даних text / xml і вільно проходять крізь шлюзи всюди, де допускається ретрансляція http-трафіку [24].

Віддалений виклик процедур означає, що додаток (неважливо, скрипт на сервері або звичайна програма на клієнтському комп'ютері) може прозоро використовувати метод, який фізично реалізований і виповнюється на іншому комп'ютері. XML тут застосовується для забезпечення універсального формату опису переданих даних. Як транспорт, для передачі повідомлень застосовується протокол HTTP, що дозволяє безперешкодно обмінюватися даними через будь-які мережеві пристрої - маршрутизатори, фаєрволи, проксі-сервера [38].

Тобто для використання треба мати: сервер XML-RPC, який надає один або кілька методів, клієнт XML-RPC, який може формувати коректний запит і обробляти відповідь сервера, а також знати необхідні для успішної роботи параметри сервера - адреса, назва методу і параметри, шо передаються.

Вся робота з XML-RPC відбувається в режимі "запит-відповідь", в цьому і є одна з відмінностей технології від стандарту SOAP, де є і поняття транзакцій, і можливість робити відкладені виклики (коли сервер зберігає запит і відповідає на нього в певний час в майбутньому). Ці додаткові можливості більше знадобляться для потужних корпоративних сервісів, вони значно ускладнюють розробку і підтримку серверів, і ставлять додаткові вимоги до розробників клієнтських рішень.

Повідомлення XML-RPC передаються методом POST протоколу HTTP і бувають трьох типів: запит, відповідь і повідомлення про помилку.

3.2.4.3 XML-RPC - відмінний інструмент для встановлення широкого спектру з'єднань між комп'ютерами. Якщо необхідно інтегрувати кілька

обчислювальних середовищ, але не потрібно безпосередньо ділитися складними структурами даних, то XML-RPC дозволяє швидко і легко встановлювати такий зв'язок. Працюючи в одному середовищі, XML-RPC дозволяє легко підключати програми, що мають різні моделі даних, і забезпечувати легкий доступ до логіки багаторазового використання [20].

Найбільш очевидним областю застосування XML-RPC є установка зв'язків між різними обчислювальним середах, для створення простих веб-сервісів [38].

3.3 SOAP в порівнянні з XML-RPC

3.3.1 XML-RPC чи SOAP [30] є вибір, якщо для реалізації віддаленого виклику використовується XML. Деякі характеристики, які визначають відмінності XML-RPC або SOAP показані в таблиці 3.2.

Таблиця 3.2 Відмінності XML-RPC або SOAP

Характеристика	XML-RPC	SOAP
Скалярні типи даних	+	+
Структури	+	+
Масиви	+	+
Іменовані масиви і структури	-	+
Обумовлені розробником кодування	-	+
Обумовлені розробником типи даних	-	+
Деталізація помилок	+	+
Легкість освоєння і практичного застосування	+	+

"Мінус" у стовпці SOAP зустрічається тільки один раз. Майже все реалізовано в ньому. Хоча основні типи даних в обох технологіях однакові, але в XML-RPC відсутня можливість задавати імена для масивів і структур. Та вирішити цю

проблему можна вводячи додаткову строкову змінну з ім'ям масиву або структури. Стосовно "визначення розробником кодуваннями" треба відзначити, що сам механізм такого обмеження не визначений - ні стандарт XML, ні транспортний рівень (протокол HTTP) таких обмежень не мають. Та й прагнення зробити клієнт/сервер XML-RPC як можна більш простим теж не призвело б до виникнення такого обмеження. Хоча, SOAP не дуже підтримує кодування (US-ASCII, UTF-8, UTF-16). Можна обійти всі ці недоліки в обох технологіях - тип даних base64. Щодо пункту "легкість в освоєнні і застосуванні", то під час значних темпів розвитку технологій і стандартів, цей пункт набуває більшої важливості. Буває таке, коли проект починає розроблятися на передовій основі, а в кінці роботи новий стандарт вже не є новим.

SOAP Version 1.2 має великий обсяг і складну документацію. Труднощі виникають навіть на етапі ознайомчого читання, не кажучи вже про розробку. А ось специфікація XML-RPC займає близько трьох сторінок А4 і гранично проста. Жодна з цих технологій не є ідеальною.

Більшість програмістів і розробників специфікацій сходяться на тому, що [30]:

якщо потрібна система для роботи зі складною логікою, якщо передаються великі комплексні структури даних, якщо потрібна повна інформація про клієнта, якщо потребується, щоб запит містив у собі інструкції по його обробці, якщо важливо, щоб за стандартом стояли відомі фірми (Microsoft, IBM, Sun) рекомендовано SOAP;

якщо ж дані є відносно простими, а додатки повинні працювати на безлічі платформ і на різних мовах, якщо важлива швидкість роботи і логіка системи не потребує складних командах рекомендовано використовувати XML-RPC;

якщо XML-RPC - написати програму клієнта/сервера не важко, про вибір ПО можна не турбуватися - хоч Borland Delphi/Kylix, хоч Phyton.

Але не всі завдання будуть вирішуватися відразу, а деякі не будуть вирішуватися взагалі.

Стандарт XML-RPC простий, а застосування XML як основного інструменту для опису даних дозволяє зробити його гнучким. Можна модифікувати протокол під кожну конкретну задачу. В використанні добре зарекомендували себе стандарти на передачу даних HTTP / HTTPS. Реалізації XML-RPC існують для безлічі платформ і мов програмування. Посилання на найбільш цікаві компоненти для: Borland Delphi / Kylix; PHP.

PHP - є кілька варіантів: XMLRPC-EPI, phpRPC, phpxmlrpc; Keith Devens 'XML-RPC (найпростіший); XML-RPC for C / C ++; XML-RPC for Python; XML-RPC.NET for MS.NET environment.

Реалізацій XML-RPC великий список. Крім того, постійно поповнюється список публічних XML-RPC-сервісів. Одними з найбільш відомих проектів, що використовують XML-RPC, є Mozilla і Apache.

3.3.2 XML-RPC чи SOAP [19]

В [19] підкреслено, що в світі XML існує два основних шляхи реалізації віддаленого виклику процедур: XML-RPC та SOAP. На базі XML-RPC-протоколу створено SOAP і виділяється він виключною простотою в застосуванні та використанні. SOAP - це протокол, в той час як REST - це архітектурний стиль. SOAP є добре продуманим протоколом, що використовується в веб індустрії і стандартизується World Wide Web Consortiumom (W3C). XML-RPC RPC-віддалений виклик процедур з допомогою XML. Запит на виконання процедури разом з параметрами записується у вигляді XML-документа і передається по мережі посередництвом HTTP на інший комп'ютер. Після завершення роботи процедури формується відповідь, і вона передається комп'ютеру, що надіслав запит. Але формат обміну даними в класичній моделі RPC залишається бінарним, що означає складність у роботі з ним. Різні платформи мають тонкощі у використанні. Ця проблема передувала створенню похідної технології XML-RPC компанією UserLand Software Inc. Передача даних у них виконується протоколом HTTP, формат даних - XML. Обмеження накладені на конфігурацію мережі і на маршрут слідування пакетів

знімаються. XML-RPC є простим типом даних text/ xml, вільно проходять через шлюзи де допускається ретрансляція HTTP-трафіку.

Хоча протокол є застарілим, але й досі використовується і має певні переваги та недоліки. Порівняльний аналіз протоколів з огляду на те, що SOAP є похідним від XML-RPC і, водночас, виступає основною альтернативою REST, доцільним є окреме з XML-RPC порівняння технології SOAP. Порівняно з SOAP, XML-RPC має простішу архітектуру, він простіший для розуміння, ніж SOAP.

Основні характеристики технології XML-RPC наведенні в таблиці 3.3.

Таблиця 3.3 Основні характеристики технологій XML- RPC

Характеристика	XML- RPC
Скалярні типи даних	Присутні
Структури	Присутні
Масиви	Присутні
Іменовані масиви та структури	Відсутні
Кодування, визначені розробником	Відсутні
Типи даних, визначені розробником	Відсутні
Деталізація помилок	Присутні
Легкість практичного застосування та засвоєння	Легкий для розуміння і практичного використання

Порівняння технології XML-RPC за характеристиками наведені у таблиці 3.4.

Таблиця 3.4 Порівняння SOAP та XML- RPC

Характеристика	SOAP	XML- RPC
Скалярні типи даних	Присутні	Присутні
Структури	Присутні	Присутні
Масиви	Присутні	Присутні
Іменовані масиви та структури	Відсутні	Відсутні
Детальний опис помилок	Відсутні	Присутні
Кодування, визначені розробником	Присутні (USASII, UTF-8, UTF-16)	Відсутні
Типи даних, визначені розробником	Присутні	Відсутні
Можливість вказати отримувача	Присутні	Відсутні
Деталізація помилок	Присутні	Присутні
Потребує «розуміння» клієнтом	Потребує	Не потребує
Легкість практичного застосування та засвоєння	Потребує час для вивчення та розуміння	Легкий для розуміння і практичного використання

3.4 SOAP в порівнянні з REST

3.4.1 В деяких джерелах інформації, наприклад [28], вказано, що SOAP і REST не можна порівнювати, тому що SOAP – це протокол, а REST – архітектурний стиль. Але вважається, що це один з джерел безладу навколо нього, оскільки люди зазвичай називають REST будь-яким HTTP-API, який не є SOAP. При цьому в подальшому автор [28], аналізуючи відому інформацію про відмінності між SOAP і REST як протокол зв'язку веб-сервісу, все ж проводять їхнє порівняння, та вказують найбільші переваги для REST над SOAP:

REST більш динамічний, немає необхідності створювати і оновлювати UDDI;

REST не обмежується форматом XML. Веб-служби REST можуть відправляти простий текст, JSON, а також XML.

Але SOAP більш стандартизований (Ex; безпека).

Під час порівняння виявлено таку відмінність між SOAP і REST - це ступінь взаємодії між реалізаціями клієнта і сервера. Клієнт SOAP працює як настільний додаток користувача, пов'язаний з сервером. Між клієнтом і сервером є жорсткий контракт, який зруйнується, якщо будь-яка сторона щось змінить, або знадобляться оновлення після змін, то краще дотримуватися контракту.

Клієнт REST більше схожий на браузер. Це загальний клієнт, який знає, як використовувати протокол і стандартизовані методи, і додаток повинен відповідати цьому, не порушуючи стандарти протоколу. Для створення додаткових методів використовуються стандартні методи. Якщо REST, то є менше зв’язків і зміни можуть бути оброблені більш акуратно. Клієнт може увійти в службу REST з нульовим знанням API, за винятком точки входу і типу носія. У SOAP клієнту потрібні попередні знання про все, що він буде використовувати, або він не зможе почати взаємодію. А клієнт REST на вимогу може бути розширений кодом, наданим сервером (кодом JavaScript, що служить для управління взаємодією з іншою службою на стороні клієнта).

Основні принципи REST, комунікація клієнт-сервер

Архітектури клієнт-сервер мають дуже чіткий поділ проблем. Всі додатки, створені в стилі RESTful повинні бути клієнт-сервером.

Stateless. Кожен запит на сервер вимагає повного представлення його стану. Сервер повинен зрозуміти запит клієнта без використання стану сервера або стану сеансу сервера.

Cacheable. Можуть бути використані кешовані обмеження, що дозволяє вказувати дані у відповіді як кешувального або не кешувального. Якщо дані помічені у виді кешувального, то можуть повторно використані в якості відповіді на один і той же наступний запит.

Рівномірний інтерфейс. Всі компоненти повинні взаємодіяти через єдиний уніфікований інтерфейс. Як що взаємодія всіх компонентів відбувається через інтерфейс, то взаємодія з різними службами є дуже простою. Якщо інтерфейс той же - це означає, що зміни в реалізації можуть бути зроблені ізольовано. Такі зміни не вплинуть на взаємодію фундаментальних компонентів, тому що рівномірний інтерфейс завжди залишається незмінним.

3.4.2 SOAP чи REST

Проблеми даного протистояння широко описані в науково-технічній літературі.

Будь-яке програмне забезпечення з часом розвивається. Можна зажадати різних версій для всіх істотних змін в додатку. Якщо використовуються версії REST додатки, то це стає однією з найбільш обговорюваних тем серед розробників REST. Існує два загальних способу для управління версіями REST додатків: URI версії; Мультимедіа версії.

В плані використання архітектура REST дуже проста. По вигляду запиту відразу можна визначити, що він робить, не розбираючись в форматах (на відміну від SOAP, XML-RPC). Дані передаються без застосування додаткових шарів, тому REST вважається менш ресурсномістким. Щоб зрозуміти, що в плані використання повинен зробити, не треба переводити з формату на формат дані.

SOAP приносить свій власний протокол і фокусується на викритті частин логіки додатка (а не даних) в якості сервісів. SOAP орієнтований на доступ до іменованих операцій, що реалізує деяку бізнес-логіку через різні інтерфейси. SOAP дуже безпечний [24-41].

На думку ж автора [40] SOAP чи REST коротко про це питання:

SOAP - дані передаються в форматі XML.

Переваги: галузевий стандарт за версією W3C; наявність суворої специфікації; широка підтримка в продуктах Microsoft, однозначність.

Недоліки: складність реалізації; ресурсомісткість парсинга XML-даних.

Щоб визначитися з тим, який спосіб реалізації використовувати необхідно розглядати в контексті реалізованої системи і її обмежень. SOAP використовується в великих корпоративних системах зі складною логікою, коли потрібні чіткі стандарти, підкріплені часом. Якщо розробляється публічне API і логіка взаємодії багато в чому покривається четвіркою методів CRUD - треба вибирайте REST. Він найбільш популярний в WEB. Яндекс, Google та інші використовують саме його для свого API.

SOAP проти REST [41, 42] - коротко можна виділити наступне:

SOAP більш застосовний в складних архітектурах, де взаємодія з об'єктами виходить за рамки теорії CRUD, в тих додатках, які не покидають межі цієї теорії, цілком прийнятним може REST через простоту і прозорості. Якщо для будь-яких об'єктів сервісу не потрібні більш складні взаємини, крім: «Створити», «Прочитати», «Змінити», «Видалити» (в 99% випадків цього достатньо), можливо REST стане правильним вибором. REST в порівнянні з SOAP, може виявитися і більш продуктивним, він не вимагає витрат на розбір складних XML команд на сервері (виконуються звичайні HTTP запити - PUT, GET, POST, DELETE). Хоча SOAP, в свою чергу, більш надійний і безпечний.

SOAP є головнішою частиною технології Web-сервісів. SOAP це базова однонаправлена модель з’єднання, що забезпечує узгоджену передачу повідомлення, допускає наявність посередників, які можуть обробляти частину повідомлення або додавати до нього додаткові елементи. SOAP дозволяє

відправнику і одержувачу XML-документів підтримувати загальний протокол передачі даних. Найбільшу перешкоду для користувачів при використанні SOAP складає механізм безпеки брандмауеру.Він блокує майже всі порти, крім HTTP-порт 80 і порт HTTP, що використовується SOAP завдяки обходу брандмауера.

3.5 Визначення відмінностей у техніках видаленого доступу протоколів взаємодії за результатами проведених досліджень.

3.5.1 Відомості про те, як SOAP відрізняється від REST

Під час виконання дипломної роботи запропонована інформація з порівняння між REST і SOAP веб сервісами була узагальнена (рис. 3.2) і є наглядною.

Рисунок 3.2 REST відрізняється від SOAP

Виявленні відмінності між REST і SOAP в показниках указані в табл. 3.5.

Таблиця 3.5. Порівняння REST і SOAP

SOAP	REST
Рекомендовано для: корпоративних програм з високим рівнем безпеки, з розподіленою середою, фінансових послуг, платіжних, телекомунікаційних послуг.	Рекомендовано для: Публічних API для веб-сервісів, мобільних сервісів, соціальних мереж.
Це протокол простого доступу до об'єктів. SOAP - це протокол, що розроблений зі специфікацією. До складу SOAP входить файл WSDL, в якому є інформація про роботу веб-служби та місце розташування веб-служби	Це архітектурний стиль перенесення репрезентативного стану (представницький державний трансферт). В цьому архітектурному стилі веб-сервіс може розглядатися як сервіс RESTful тільки в тому випадку, якщо він відповідає обмеженням: клієнтський сервер, Stateless, Cacheable, багаторівнева система, єдиний інтерфейс
Стандартизований протокол із заздалегідь визначеними правилами. Вимагає строго дотримуватися стандартів, які він визначає. Чітко визначений механізм опису інтерфейсу, наприклад WSDL + XSD, WS-Policy.	Архітектурний стиль з вільним керівництвом і рекомендаціями, не визначає занадто багато стандартів, таких як SOAP. Формальні описи стандартів не набули широкого поширення.
Протокол SOAP, не може використовувати архитектурный паттерн REST.	може використовувати веб-служби SOAP (як базовий протокол), і може використовувати будь-який протокол, (HTTP, SOAP). REST- всього лише архитектурний паттерн.
Підтримує тільки XML формат даних.	Підтримує формати **повідомлення** різні: text, JSON,XML,HTML,**YAML**, та є найкращим форматом для передачі даних є JSON.
Працює з різними протоколами передачі HTTP,SMTP,UDP, SMTP, HTTP є чинними протоколами прикладного рівня, які використовуються в якості транспорту.	Працює тільки за HTTP (S). Пов'язаний з транспортною моделлю HTTP.
Працює з операціями, які реалізують яку-небудь бізнес логіку за допомогою інтерфейсів. Використовує сервісні інтерфейси для розкриття бізнес-логіки	Може працювати з ресурсами. Кожен URL це представлення якого або ресурсу. Використовує URL для розкриття бізнес-логіки.
Для надання своїх функцій клієнтським застосуванням SOAP використовує сервісні інтерфейси. У SOAP файл WSDL надає клієнту необхідну інформацію, яку можна використовувати для розуміння того, які послуги може пропонувати веб-служба.	Використовує локатори Uniform Service для доступу до компонентів на апаратному пристрої.

Продовження Таблиця 3.5. Порівняння REST і SOAP

SOAP	REST
Клієнт і Сервер пов'язані контрактом WSDL	Між клієнтом і сервером немає договору.
Необхідний мінімальний набір інструментів / проміжне програмне забезпечення. Потрібно лише підтримка HTTP (в основному це HTTP)	URL зазвичай посилається на ресурс, до якого здійснюється доступ / видалений / оновлений
Функціонально керований - дані доступні як сервіси («getUser»)	Керований даними, які доступні як ресурси («користувач»).
У Java JAX-WS використовується Java-API для веб-сервісів SOAP.	В Java JAX-RS - це API-інтерфейс java для веб-служб RESTful.
За замовчуванням без збереження стану, але можна зробити API-інтерфейс SOAP зі збереженням стану.	Без збереження стану - без сеансів на стороні сервера.
На основі читання не може бути поміщений в кеш, виклики API не можуть кешованими.	Може бути закешованим, тобто виклики API можуть бути кешованими.
Підтримує SSL і WS-security, Вбудована сумісність. Безпека SOAP добре стандартизована за допомогою WS-SECURITY	Підтримує тільки SSL та **HTTPS.** Безпека забезпечується на рівні протоколу HTTP, така, як базова аутентифікація і шифрування зв'язку через TLS.
Визначає свою власну безпеку.	Служби RESTful успадковують заходи безпеки від основного транспорту. Rest успадковує свою безпеку від нижчого транспортного рівня.
Підтримує ACID (Atomisity, Consistency, Isolations, Durability)	Підтримує транзакції, але не один з ACID не сумісний з двома фазовим коммитом
Вбудована обробка помилок.	Немає обробки помилок, хоча можна використовувати коди помилок HTTP
Вимагає більшої пропускної здатності та обчислювальної потужності.	Вимагає менше ресурсів.
Вимагає більше смуги пропускання і ресурсу, ніж REST, тому що SOAP вимагає більшої пропускної здатності для його використання	Вимагає менше пропускної здатності і ресурсу, ніж SOAP. Повідомлення REST в основному складаються з повідомлень JSON. тому REST не вимагає великої смуги пропускання при відправці запитів на сервер.
Корисне навантаження відповідає схеміSOAP	Немає обмежень на корисне навантаження.
Має високий рівень безпеки, є стандартизованим, має розширюваність.	Має масштабованість, зручність браузера, кращу продуктивність, гнучкість.
Веб-сервіси SOAP складно підтримувати, тому що у разі будь-які змін в WSDL, потрібно знову створити заглушку клієнта	Веб-сервіси REST зазвичай прості в обслуговуванні.
Веб-сервіси SOAP важко освоїти, так як потрібно розуміти WSDL, клієнтську заглушку.	Веб-сервіси REST легко зрозуміти, потрібно анотувати Java-клас анотаціями JAX-RS для використання різних HTTP-методів.
Має низьку продуктивність, велику складність, меншу гнучкість.	Має менше безпеки, не підходить для розподілених середовищ.

Вищезазначені концепції є визначальними для характеристик REST і відрізняють архітектуру REST від інших архітектур, таких як веб-служби. служба

REST є веб-службою, але веб-служба не обов'язково є службою REST. Один з недоліків полягає в тому, що з REST неможна змінити інтерфейс. REST оптимізований для Інтернету, тому REST по HTTP є популярним.

Відомості про те, як SOAP відрізняється від REST є цікавими з практичної точки зору, тому ця інформація під час виконання дипломної роботи була узагальнена надані порівняння між REST і SOAP веб сервісами. Існує багато відмінностей між веб-службами SOAP і REST, на рисунку 3.2 виділено найважливіші.

3.5.2 SOAP в порівнянні з іншими техніками видаленого доступу

3.5.2.1У разі необхідності реалізації віддаленого виклику, треба зробити вибір серед існуючих технологій для отримання бажаного результату. Для здійснення такого вибору активно вивчаються ці технології.

За результатами проведеного аналізу в цій галузі виявлено, що дуже активно досліджуються відомі техніки видаленого доступу, порівнюються її позитивні і негативні боки, характеристики та опис найбільш відповідних ефективних випадків використання.

Є багато інформації про порівняння протоколів взаємодії веб-сервісів з подальшим виявленням переваг та недоліків кожного з протоколів, зосереджено увагу як на будові протоколів, так і на порівнянні з іншими протоколами для ефективного аналізу.

На сьогоднішній день найбільш широке розповсюдження отримали SOAP, XML-RPC та REST. Вони відрізняються як контекстом, так і використанням. Дуже багато праць надають рекомендації з вибору потрібної технології, тому що жодна з цих технологій не є ідеальною і не може забезпечити потрібну повноту вирішення проблем.

Порівняння SOAP і іншими техніками видаленого доступу (CORBRA або DCOM) надані в таблиці 3.6.

Таблиця 3.6. Порівняння SOAP і іншими техніками видаленого доступу (CORBRA або DCOM)

SOAP	Інші техніки видаленого доступу (CORBRA або DCOM)
Не симетричний -проще у використанні	Симетричний ніби CORBRA або DCOM
Є незалежнішим від платформи і мови	Є залежнішим від платформи і мови CORBRA або DCOM
Використовує HTTP в якості транспортного протоколу і дані зберігаються у форматі XML, який може бути прочитаний оператором	CORBRA або DCOM мають свої власні бінарні формати, які використовуються для траннспортировки даних складним чином
Ідентифікує об'єкт, відмінний від кінцевого URL. Об'єкти SOAP є незалежними і їх складно підтримувати.	У випадку інших методів видаленого доступу робота в цьому випадку може бути простіше

3.5.2.2 Різниця між SOAP і CORBA, DCOM і Java RMI

До появи SOAP і REST такі методи віддаленого доступу, як методи RPC, широко використовувалися.

Різні методи віддаленого доступу, які були доступні, це такі:

CORBA - система була створена для того, щоб програми, створені на різних платформах, могли спілкуватися.

Основним недоліком цього методу те, що він розробляється окремою мовою, мовою визначення інтерфейсу, і потрібне представляти додаткову мову, яку повинен був вивчити розробник, щоб використовувати систему CORBA.

DCOM - фірмова технологія Microsoft для клієнтів доступу віддалених компонентів. Найбільша проблема з цим механізмом в тому, що клієнтський додаток міг звільняти ресурси, коли вони більше не потрібні.

По-друге, коли клієнт відправив запит, він повинен переконатися, що запит був упакований вірно, щоб веб-служба могла зрозуміти відправлений запит. Інша проблема в тому, що якщо клієнтська програма -це додаток на основі Java, що повинно працювати з DCOM (технологія Microsoft), потрібно додаткове кодування, щоб додатки, створені на інших мовах програмування, могли працювати з веб-службами на основі DCOM.

У порівнянні з класичним клієнт-серверних підходом, використання технології CORBA для розробки розподілених додатків має такі переваги: використання IDL для опису інтерфейсів дозволяє розробляти програмні компоненти незалежно від мови програмування і базової операційної системи; підтримка багатою інфраструктурою розподілених об'єктів; прозорість виклику віддалених об'єктів. Програмні рішення на базі технології CORBA майже не виходять за рамки окремих підприємств. Розробка систем на базі технології CORBA поєднана з такими труднощами: недобра сумісність різних реалізацій технології CORBA від різних постачальників; проблеми взаємодії вузлів CORBA через Інтернет; неузгодженість багатьох архітектурних рішень CORBA.

ВИСНОВКИ

В третім розділі дипломної роботи показані практичні результати проведених досліджень. Запропонована узагальнена інформація з порівняння між REST і SOAP веб сервісами, Проведені порівняння характеристик між REST і SOAP, здійснено порівняння SOAP з іншими техніками видаленого доступу, виявленні відмінності між SOAP і CORBA, DCOM і Java RMI.

3 Вибір протоколів взаємодії потребує значної уваги. Розвиток проекту в залежить від вимог до архітектури додатку. Кожна з технологій має переваги та недоліки. Розглянуті протоколи функціонально відрізняються, але є корисними і залежать від контексту використання XML-RPC, SOAP, REST. Використовуються і в невеликих системах з простою логікою, і в масштабних корпоративних додатках, та в комбінуванні технологій для досягнення оптимальної продуктивності і для задоволення потреб як розробників, котрі його використовують, так і інших систем, що взаємодіють з веб-сервісом.
Вибір технології залежить від цілей та задач, котрі реалізуються,
довгостроковості роботи.

ЗАГАЛЬНІ ВИСНОВКИ

1 Аналіз науково-технічної літератури показав, що відсутній єдиний підхід до визначень у клієнт-серверній системи, це є перешкодою під час вибору важливих архітектурних рішень та вибору протоколів взаємодії для ефективного використання із – за плутанини в визначеннях. Найчастіше помилково проводиться відмінність або використовуються взаємозамінні терміни.

2 Спостерігається тенденція в дослідженнях та в проведенні аналізів протоколів взаємодії з подальшим виявленням переваг та недоліків кожного з них для будь яких випадків використання, та зосереджуються не тільки на будові окремого протоколу, а і на їхнім порівнянні. Це підкреслює актуальність теми і необхідність продовження досліджень в цьому напряму.

3 Єдиного підходу для вибору протоколу взаємодії, не існує.

Для вибору відповідного протоколу варто ураховувати умови та область використання технологічних ланок системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Закон України про Національну програму інформатизації

2. Закон України про телекомунікації

3. Технології «клієнт-сервер» і Internet Розподілені бази даних технології в ГІС,
https://geoknigi.com/ book_view.php?id-644

4. О модели взаимодействия клиент-сервер простыми словами. Архитектура «клиент-сервер» с примерами. Сервера и протоколы. 28.07.2016
https://zametkinapolyah.ru/servera-i-protokoly/

5. Інженерія програмного забезпечення усіх форм навчання. Міністерство освіти і науки України ЗНТУ Кафедра програмних засобів "Технології розробки мережевих додатків" Конспект лекцій для студентів. Запоріжжя 2016
http://eir.zntu.edu.ua/bitstream/123456789/2877/1/Stepanenko_The_summary.pdf

6. Клиент-сервер - о технологии простыми словами 30 Декабря 2019
https://galtsystems.com/blog/start/klient_server_o_tekhnologii_prostymi_slovami/

7. Морозов Ю.В., Пастернак І.І., Класифікація засобів модульної взаємодії між клієнтом і сервером. 2011. Національний університет “Львівська політехніка”, кафедра електронних обчислювальних машин. http://ena.lp.edu.ua

8. Загальна будова глобальних мереж 07.12.2017 Київський національний уіверситет імені Тараса Шевченка, кафедра математичної фізики http://www.matfiz.univ.kiev.ua/userfiles/files/Pres25.pdf

9. Петроченко А.О. Технология «Клиент – сервер». Реферат. https://works.doklad.ru/view/iLGEMa4lxXA.html

10. Що таке взаємоді клієнт-сервер? https://registrio.com.ua/shcho-take-vzaiemodiia-kliient-server.html

11. Клієнт-серверні системи https://stud.com.ua/97304/informatika/kliyent_serverni_sistemi

12. Коржов В. Многоуровневые системы клиент-сервер. Издательство Открытые системы (17 июня 1997). https://www.osp.ru/nets/1997/06/142618/

13 Основи клієнт-серверних технологій Протокол HTTP і способи передачі даних на сервер технологій http://www.lib.mdpu.org.ua/e-book/web/lec2.htm
14. Maryana Rozhankivska Інфрмаційні технології в інфраструктурі ринку. Архітектура клієнт-сервер. Лекція № 8 2008 http://educational.mariroz.com/InformTechVInfrastrRynkulect/lect8.pdf
15. Технология «Клиент – сервер» https://works.doklad.ru/view/b
16. Основи технології клієнт/сервер https://studopedia.com.ua/1_332969_osnovi-tehnologii-kliientserver.html
17. Поняття "відкрита система" і проблеми стандартизації http://matveev.kiev.ua/ archnet/glava1/003.htm
18. Протоколи Інтернету https://buklib.net/books/23157/
19. Кравчук Є. С. Аналіз протоколів взаємодії у клієнт- серверній архітектурі International Scientific Journal № 6, т. 1, 201620file:///C:/Users/USER/Downloads/mnj_2016_6(1)__11.pdf
20. Вікіпедія https://ru.wikipedia.org/wiki/
21.Лекція 8. Основи технології клієнт – сервер http://ito.vspu.net/ENK/2015-2016/Programming_SQL/lections/Lectoin-8.pdf
22. Програмні канали https://studopedia.com.ua/1_58316_programni-kanali.html
23. Зайцев В.Г., Дробязко І.П. Операційні системи. Київ «КПІ ім. Ігоря Сікорського» 2019 https://ela.kpi.ua/bitstream/123456789/29600/1/
24. Описание WEB-сервисов, XML-RPC http://java-online.ru/web-service.xhtml
25. Веб-сервисы как средство интеграции приложений в WWW http://kit.znu.edu.ua/iLec/6sem/PMES/Lecs/PCIS/Lec7-8-WebServices-RESTFul.htm
26. **Технологія клієнт – сервер. Моделі реалізації цієї технології.** https://lubbook.org/book_499_glava_58_131._Tekhnolog%D1%96ja
27. Архитектура REST http://habrahabr.ru/post/38730/
28. SOAP vs REST (различия) http://utyatnishna.ru/info/2543
29. Что такое веб сервисы? https://jsehelper.blogspot.com/2016/04/

30. Лозовюк А., XML-RPC: вызов процедур посредством XML, Издательский Дом "КОМИЗДАТ". http://citforum.ck.ua/internet/xml/xml_rpc/
31. Использование HTTP методов для создания RESTful сервисов http://www.restapitutorial.ru/lessons/httpmethods.html
32. Руководство для начинающих по HTTP и REST https://code.tutsplus.com/ru/tutorials/a-beginners-guide-to-http-and-rest--net-16340
33. Родригес А Web-сервисы RESTful: основы. Опубликовано 16.09.2015 https://www.ibm.com/developerworks/ru/library/ws-restfu/
34. Фастовский Э.Г Сервис-ориентированные технологии интеграции информации. Лекция 5. Стандарт SOAP – протокол взаимодействия сервисов. 1 Сервіс-орієнтовані технології інтеграції інформації 2011 г. http://khpi-iip.mipk.kharkiv.edu/library/sotii/lectures/Lecture5.pdf
35. Бергер А., Горбач И. Меломед Э., Щербинин В., Степаненко В. Mscrosoft ®SL Server 2005 Analysis Services. OLAP и многомерный анализ данных. Санкт-Петербург. БХВ - Санкт-Петербург. 2007. https://books.google.com.ua/books?idy
36. Дьомкін В. Лекція №9. Взаємодія з мережею Операційні системи (CC-BY-NC). 2015, https://medium.com/@IvanZmerzlyi/
37. Операційна система UNIX (Конспект лекцій) http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=48369
38. Лозовик А.12.11.2004 Введение в XML-RPC http://hostinfo.ru/articles/450
39. Тарнавський Ю. А., Кузьменко І. М. Організація комп'ютерних мереж. Підручник. Київ КПІ ім. Ігоря Сікорського 2018. https://ela.kpi.ua/bitstream/123456789/25156/1/Tarnavsky_Kuzmenko_Org_Komp_merej.pdf
40. Рельсы веб-интеграции. REST и SOAP https://www.intervolga.ru/blog/projects/relsy-veb-integratsii-rest-i-soap/
41. Черняк Л. SOAP и REST, вместе или порознь? 18.09.2003 www.citforum.ru
42. Які відмінності між SOAP і REST? https://www.quality-assurance-group.com/soap-web-servis-api-testing-interview/

43. Розробка інформаційних ресурсів та систем Лекція 3.1. Технології реалізації та інтеграції потоків завдань http://www.its.kpi.ua/itm/tkot/Students/Lec3_1-Wf-Implementation.pdf