Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту
інформації
_____ Іван ПАРХОМЕНКО
«___» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: «Засіб багатофакторної автентифікації до системи управління базами даних»

Виконавець: студент IV курсу, групи КБ-41

_____ **Роман КОГУТ**
(підпис) (ім'я, прізвище)

	Підпис	**Ім'я ПРІЗВИЩЕ**
Керівник		Іван ПАРХОМЕНКО
Нормоконтроль		Лариса МИРУТЕНКО

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студенту **КБ-41** (група) **Когуту Роману Дмитровичу** (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Засіб багатофакторної автентифікації до системи управління базами даних

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Засоби та методи забезпечення захисту інформації в системах управління базами даних

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитись із сучасним станом захисту від комп'ютерних злочинів, розробити рекомендації щодо протидії кіберзагрозам у системах управління базами даних

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність реалізація засобу підвищення безпеки систем управління базами даних, який може бути впроваджений для зменшення ризиків витоку інформації та несанкціонованого доступу

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

Завдання видав _____ (підпис) Іван ПАРХОМЕНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Роман КОГУТ (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 06.12.2024	*виконано*
2	Аналіз літератури	07.12.2024 – 30.12.2024	*виконано*
3	Обґрунтування вибору методів дослідження	16.01.2025 – 30.01.2025	*виконано*
4	Аналіз загроз безпеці баз даних	31.01.2025 – 15.02.2025	*виконано*
5	Дослідження методів захисту систем управління базами даних	16.02.2025 – 28.02.2025	*виконано*
6	Розробка рекомендацій щодо забезпечення безпеки систем управління базами даних	01.03.2025 – 31.03.2025	*виконано*
7	Написання тексту атестаційної роботи	01.04.2025 – 15.05.2025	*виконано*
8	Оформлення пояснювальної записки	16.05.2025 – 23.05.2025	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	24.05.2025 – 10.06.2025	*виконано*

Завдання видав _____ (підпис) Іван ПАРХОМЕНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Роман КОГУТ (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 13 червня 2025 року

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, має 67 сторінок основного тексту, 2 таблиці та 24 рисунки. Список використаних джерел містить 24 найменування і займає 3 сторінки.

Метою роботи є впровадження засобу багатофакторної автентифікації до систем управління базами даних та забезпечення підвищення їх захисту.

Для досягнення зазначеної мети необхідно виконати наступні завдання:

- проаналізувати сучасні підходи до організації баз даних та їх управління;
- дослідити основні загрози безпеці баз даних та методи їх усунення;
- розробити програмний модуль багатофакторної автентифікації до СУБД;

Об'єктом дослідження є процес захисту системи управління базами даних від кіберзагроз інформаційної безпеки.

Предметом дослідження є методи автентифікації, контролю доступу та шифрування даних у системі управління базами даних.

Методи дослідження, використані при підготовці кваліфікаційної роботи: аналіз сучасних загроз для баз даних та методів їх запобігання, порівняння методів автентифікації та контролю доступу, моделювання процесів безпеки у системі управління базами даних, тестування ефективності розроблених рекомендацій.

Практична цінність отриманих результатів полягає у реалізації засобу підвищення безпеки систем управління базами даних, який може бути впроваджений для зменшення ризиків витоку інформації та несанкціонованого доступу.

Ключові слова: база даних, система, безпека, автентифікація, контроль доступу, конфіденційність, захист, аудит, інформаційна безпека, JSON Web Token (JWT).

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API	–	Application Programming Interface
DDoS	–	Distributed Denial of Service
DLP	–	Data Loss Prevention
DoS	–	Denial of Service
ERP	–	Enterprise resource planning
HR	–	Human Recruiter
IBM	–	International Business Machines Corporation
IP	–	Internet Protocol
JS	–	Java Script
JSON	–	Java Script Object Notation
JWT	–	JSON Web Token
SQL	–	Simple Query Language
VPN	–	Virtual Private Network
АС	–	Автоматизована система
БД	–	База даних
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення
СУБД	–	Система управління базами даних

ВСТУП

Актуальність роботи зумовлюється тим, що у сучасному світі обсяг даних, які обробляються системами управління базами даних (СУБД), зростає з неймовірною швидкістю. Нижче наведено графік (див. рис. 1), що демонструє зростання обсягів цифрових даних у світі з 2010 до 2024 року [1]. Як видно, обсяг даних збільшується експоненційно – з 1,2 ZB у 2010 році до понад 96 ZB у 2024-му. ZB (зеттабайт) — це одиниця вимірювання обсягу цифрової інформації, що дорівнює 10^{21} байт.

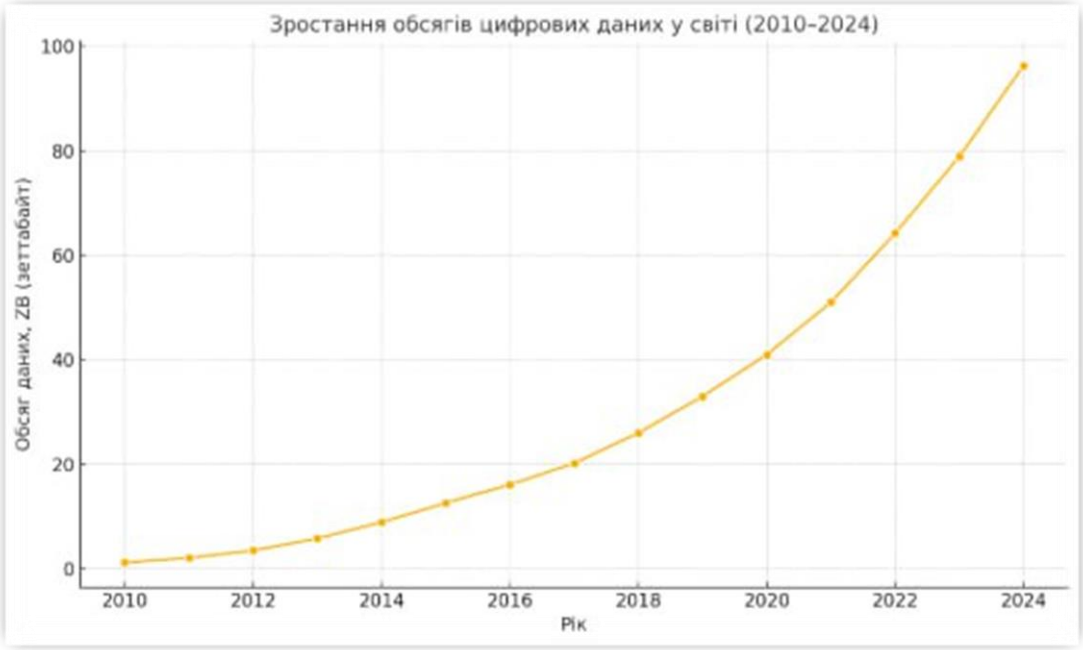


Рисунок 1 – Зростання обсягів цифрових даних

Однак поряд із цим зростає кількість загроз, пов’язаних із несанкціонованим доступом, атаками на СУБД, витоками інформації та кібератаками.

Забезпечення безпеки СУБД є критично важливим завданням, оскільки витік чи компрометація інформації можуть завдати значних фінансових втрат, репутаційного збитку та загрожувати правам користувачів. Сучасні методи захисту інформації вимагають комплексного підходу, який включає автентифікацію, контроль доступу, шифрування, моніторинг активності та

запобігання атакам. Вивчення цих аспектів та розробка ефективних заходів безпеки є надзвичайно актуальним питанням.

Бази даних є невід'ємною частиною цифрової інфраструктури підприємств, державних установ, медичних та освітніх закладів, банківських систем та інших сфер [2]. Нижче наведено діаграму (див. рис. 2), яка показує використання баз даних у різних галузях.

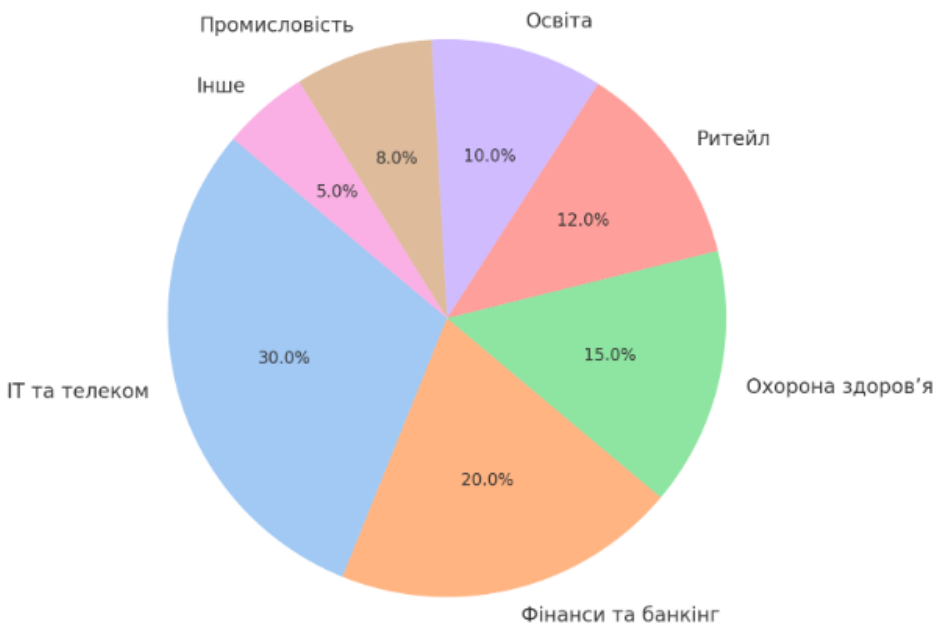


Рисунок 2 – Використання баз даних у різних галузях

Методи дослідження, використані при підготовці кваліфікаційної роботи:

- аналіз наукової літератури та нормативних документів у сфері інформаційної безпеки;
- порівняльний аналіз методів захисту даних;
- моделювання процесів автентифікації та контролю доступу в СУБД;
- тестування ефективності запропонованих заходів захисту.

Дослідження спрямоване на реалізацію засобу підвищення безпеки систем управління базами даних, який може бути впроваджений для зменшення ризиків витоку інформації та несанкціонованого доступу.

РОЗДІЛ 1
АНАЛІЗ ОРГАНІЗАЦІЇ ТА УПРАВЛІННЯ БАЗАМИ ДАНИХ

1.1 Основні концепції баз даних: що таке база даних, її роль та призначення

База даних (*англ.* database) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування [3]. Зазвичай БД містить таблиці, схеми, подання, збережені процедури та інші об'єкти. Дані в БД організовані відповідно до моделі організації даних. Таким чином, сучасна БД містить опис даних, окрім самих даних, а також може містити інструменти для їхньої обробки.

Базу даних також можна описати як набір даних, пов'язаних спільними ознаками або властивостями й розташованих, наприклад, в алфавітному порядку [4]. Великі обсяги даних можуть бути згруповані в одній БД, що дозволяє використовувати безліч варіацій групування інформації.

Основною перевагою баз даних є швидкість, з якою можна вводити і використовувати необхідну інформацію. Завдяки спеціальним алгоритмам, що використовуються в базі даних, необхідну інформацію можна легко знайти всього за кілька секунд. Крім того, дані в БД мають певні взаємозв'язки, так що зміна в одному рядку може спричинити зміну в іншому рядку.

Бази даних використовуються майже у всіх сферах сучасного життя, від бізнесу та медицини до розваг і соціальних мереж. Розглянемо практичні приклади застосування баз даних у різних галузях.

– Банківська справа та фінанси.

Основні завдання БД у фінансовому секторі: облік клієнтів та їхніх рахунків, обробка банківських транзакцій у реальному часі, управління кредитами та депозитами, запобігання шахрайству тощо;

– Електронна комерція (e-commerce, онлайн-магазини): управління каталогами товарів, обробка замовлень та платежів управління запасами та складами, аналітика продажів та рекомендації товарів тощо.

– Соціальні мережі: збереження профілів користувачів, взаємодія між користувачами (лайки, коментарі, повідомлення), стрічка новин та рекомендаційний алгоритм, аналітика та реклама.

– Охорона здоров'я та медицина: електронні медичні картки пацієнтів, управління лікарськими препаратами та рецептами, запис пацієнтів на прийом, аналіз медичних даних для досліджень тощо.

– Логістика та транспорт: відстеження вантажів і посилок, управління маршрутами транспорту, оптимізація логістичних ланцюгів, контроль за складськими запасами.

– Розважальна індустрія (фільми, музика, ігри): каталоги фільмів, музики, ігор, рекомендаційні алгоритми, стрімінг контенту.

– Державні установи та електронне урядування.

Для того щоб створити БД використовують спеціальне програмне забезпечення, а для встановлення контролю за ними існує система управління базою даних. СУБД - це програмно-апаратні комплекси, які забезпечують визначення, створення, функціонування, контроль, управління та використання баз даних [3]. Застосунки, які використовуються для роботи з БД можуть бути як і частиною СУБД, так і автономними. Найбільш популярними СУБД на сьогоднішній день є PostgreSQL, MySQL, Microsoft SQL Server та Oracle Database [5]. СУБД дозволяють ефективно працювати з базами даних, обсяг яких робить неможливим їх ручне опрацювання.

Сучасні організації та підприємства активно використовують бази даних для управління інформацією, адже вони дозволяють ефективно зберігати, обробляти, аналізувати та управляти великими обсягами даних. Основні ролі та призначення баз даних [6]:

1. Централізоване зберігання даних: БД забезпечують єдине місце для зберігання всієї інформації, що значно полегшує управління даними. Це

особливо важливо для великих організацій, де інформація використовується багатьма відділами одночасно. У банку всі дані про клієнтів, їхні рахунки та транзакції зберігаються в одній базі даних, що забезпечує швидкий доступ і цілісність інформації.

2. Швидкий доступ та ефективне управління інформацією: СУБД дозволяють швидко знаходити, оновлювати, видаляти чи додавати записи, що значно перевершує традиційні файлові системи.

3. Забезпечення цілісності та узгодженості даних: Цілісність означає, що дані залишаються коректними та узгодженими навіть при одночасному доступі багатьох користувачів. У системі бронювання авіаквитків однакове місце в літаку не може бути продано двом різним пасажирам.

4. Використання великих обсягів даних: Сучасні бази даних здатні обробляти терабайти або навіть петабайти інформації, що критично для великих організацій, таких як Google, Facebook або Amazon. Соціальні мережі зберігають величезні масиви даних про користувачів, їхні дописи, лайки та коментарі.

5. Безпека та контроль доступу: БД надають можливість визначати, хто і які дані може переглядати, змінювати або видаляти. Вони підтримують механізми автентифікації та шифрування. Лікар може бачити всі медичні записи пацієнта, але медсестра – лише необхідну для роботи інформацію.

6. Запобігання дублювання даних та оптимізація зберігання: Правильне структурування бази зменшує надмірність та запобігає дублюванню, що економить місце для зберігання даних.

7. Підтримка багатокористувацького доступу: Багато користувачів можуть одночасно працювати з базою даних, не створюючи конфліктів і збоїв у системі. У компанії відділ продажів, бухгалтерія та HR можуть працювати з однією базою, отримуючи необхідні їм дані.

8. Автоматизація бізнес-процесів: БД використовуються для автоматизації обліку, ведення фінансової звітності, управління товарними запасами тощо. В ERP-системі (наприклад, SAP) база даних автоматично оновлює залишки товарів після здійснення покупки.

9. Підтримка аналітики та прийняття рішень: Бази даних можуть використовуватися для аналізу даних, побудови звітності та прогнозування. Система бізнес-аналітики аналізує продажі за рік і прогнозує попит на майбутні періоди.

10. Гнучкість та масштабованість: Бази даних можуть розширюватися та змінюватися відповідно до потреб організації, підтримуючи нові технології та пристрої. Хмарні бази даних (Amazon, Google, BigQuery) дозволяють компаніям масштабувати зберігання та обробку даних відповідно до навантаження.

1.2 Системи управління базами даних. Огляд функціоналу та ключових компонентів

Система управління базами даних (СУБД) - це ПЗ для створення та управління базами даних. СУБД дозволяє кінцевим користувачам створювати, захищати, читати, оновлювати та видаляти дані в базі даних. Вона також керує безпекою, цілісністю даних і паралельністю БД.

Найпоширеніший тип платформи управління даними, СУБД, по суті, слугує інтерфейсом між базами даних і користувачами або прикладними програмами, гарантуючи, що дані послідовно організовані і залишаються легкодоступними [7].

Механізм БД забезпечує доступ до даних, їх блокування та модифікацію, а схема БД визначає логічну структуру. Ці три основні елементи даних допомагають забезпечити безпеку, цілісність та уніфіковані процедури адміністрування даних.

Нижче наведено загальні функції, які виконує СУБД [8]:

– Завдання адміністрування. СУБД підтримує багато типових завдань адміністрування баз даних, включаючи управління змінами, моніторинг та налаштування продуктивності, безпеку, резервне копіювання та відновлення. Більшість систем керування базами даних також відповідають за автоматичний

відкат і перезапуск, а також ведення журналів і аудит активності в базах даних і додатках, які отримують до них доступ.

– Зберігання. СУБД забезпечує ефективне зберігання та пошук даних, гарантуючи, що дані зберігаються в таблицях, рядках і стовпцях.

– Контроль паралелізму. У середовищах, де кілька користувачів одночасно отримують доступ до бази даних і змінюють її, СУБД гарантує контрольоване виконання транзакцій, щоб запобігти пошкодженню або неузгодженості даних.

– Централізоване представлення. СУБД забезпечує централізоване представлення даних, до яких багато користувачів можуть отримати доступ з різних місць у контрольований спосіб. СУБД може обмежувати, які дані бачать кінцеві користувачі і як вони їх бачать, забезпечуючи багато представлень однієї схеми бази даних. Кінцевим користувачам і програмам не потрібно розуміти, де фізично знаходяться дані або на якому типі носія вони зберігаються, оскільки СУБД обробляє всі запити.

– Маніпулювання даними. СУБД забезпечує цілісність та узгодженість даних, дозволяючи користувачам вставляти, оновлювати, видаляти та модифікувати дані всередині бази даних.

– Незалежність даних. СУБД забезпечує як логічну, так і фізичну незалежність даних, щоб захистити користувачів і програми від необхідності знати, де зберігаються дані, або турбуватися про зміни у фізичній структурі даних. Поки програми використовують інтерфейс прикладного програмування (API) для бази даних, який надає СУБД, розробникам не доведеться модифікувати програми тільки тому, що в базу даних були внесені зміни.

– Резервне копіювання та відновлення**.** СУБД полегшує резервне копіювання та відновлення даних, створюючи резервні копії, щоб дані можна було відновити до узгодженого стану. Це захищає від втрати даних через апаратні збої, програмні помилки або інші непередбачувані події. У реляційних системах керування базами даних (СУБД) - найпоширенішому типі СУБД - API

- це мова структурованих запитів (SQL), стандартна мова програмування для визначення, захисту та доступу до даних.

СУБД - це складне системне програмне забезпечення, що складається з декількох інтегрованих компонентів (див рис 1.1), які забезпечують узгоджене, кероване середовище для створення, доступу та модифікації даних у БД. Ці компоненти включають наступне [8]:

– Механізм зберігання даних. Цей базовий елемент СУБД використовується для зберігання даних. Для зберігання даних СУБД повинна взаємодіяти з файловою системою на рівні операційної системи (ОС). Вона може використовувати додаткові компоненти для зберігання даних або взаємодіяти з власне даними на рівні файлової системи.

– Каталог метаданих. Іноді його називають системним каталогом або словником бази даних, каталог метаданих функціонує як сховище для всіх створених об'єктів бази даних. Коли створюються бази даних та інші об'єкти, СУБД автоматично реєструє інформацію про них у каталозі метаданих. СУБД використовує цей каталог для перевірки запитів користувачів на отримання даних, а користувачі можуть звертатися до каталогу за інформацією про структури баз даних, які існують в СУБД. Каталог метаданих може містити інформацію про об'єкти бази даних, схеми, програми, безпеку, продуктивність, зв'язок та інші відомості про навколишнє середовище баз даних, якими вона керує.

– Мова доступу до бази даних. СУБД повинна також надавати API для доступу до даних, як правило, у вигляді мови доступу до бази даних, яку можна використовувати для модифікації даних, а також для створення об'єктів бази даних, захисту та авторизації доступу до даних. SQL є прикладом мови доступу до бази даних і охоплює кілька наборів команд, включаючи мову управління даними для авторизації доступу до даних, мову визначення даних для визначення структур бази даних і мову маніпулювання даними для читання і модифікації даних.

– Механізм оптимізації. СУБД також може надавати механізм оптимізації, який використовується для розбору запитів на мові доступу до бази даних і перетворення їх на команди для доступу до даних та їхньої модифікації.

– Обробник запитів. Після того, як запит оптимізовано, СУБД повинна забезпечити спосіб виконання запиту та повернення результатів.

– Менеджер блокувань. Цей важливий компонент СУБД керує одночасним доступом до одних і тих самих даних. Блокування потрібні для того, щоб гарантувати, що кілька користувачів не намагаються одночасно змінити одні й ті самі дані.

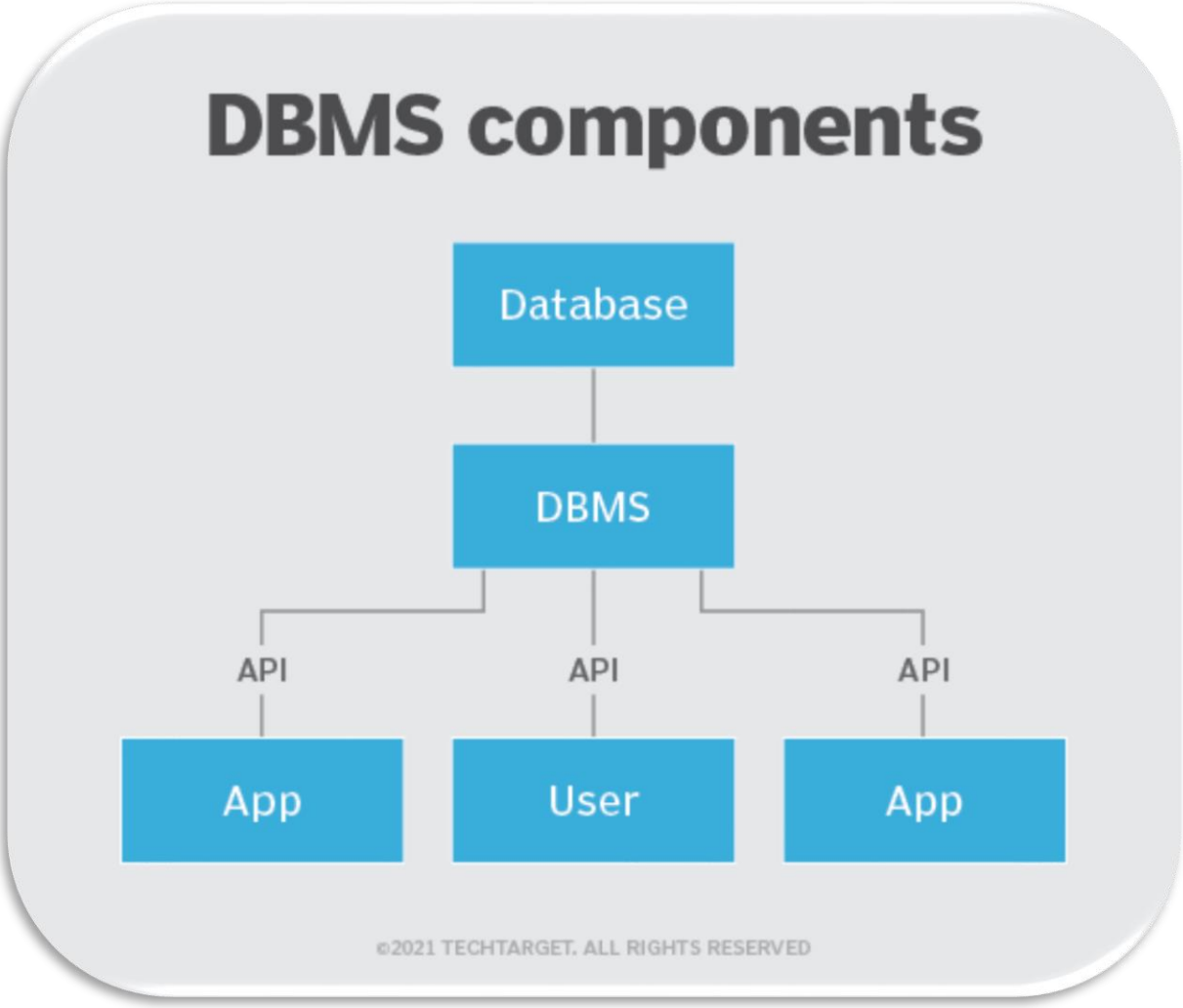


Рисунок 1.1 – Структура СУБД включає в себе кілька компонентів

Таблиця 1.1.

Типи СУБД

Тип СУБД	**Опис**	**Приклади**
Реляційні (SQL)	Дані у вигляді таблиць	MySQL, PostgreSQL, Oracle, SQL Server
Документоорієнтовані	Дані у форматі JSON/BSON	MongoDB, CouchDB
Графові	Використовують вузли та зв’язки	Neo4j, ArangoDB
Ключ-значення	Зберігають дані у форматі "ключ – значення"	Redis, Amazon DynamoDB
Колонкові	Дані організовані за стовпцями	Apache Cassandra, HBase

1.3 Архітектура СУБД

База даних зберігає багато критично важливої інформації, доступ до якої має бути швидким і безпечним. Тому важливо вибрати правильну архітектуру для ефективного управління даними. Архітектура системи управління базами даних (СУБД) має вирішальне значення для ефективного управління даними та продуктивності системи. Вона допомагає користувачам виконувати свої запити під час підключення до бази даних. Вона зосереджується на тому, як база даних спроєктована, побудована та підтримується, визначаючи, як користувачі отримують доступ до неї та взаємодіють з нею.

Існує декілька типів архітектури СУБД, які використовуються у відповідності до вимог. Типи архітектури СУБД наведені нижче:

- 1-рівнева архітектура;
- 2-рівнева архітектура;
- 3-рівнева архітектура;

В однорівневій архітектурі БД (див рис. 1.2) доступна безпосередньо користувачеві, користувач може безпосередньо сидіти в СУБД і використовувати її, тобто клієнт, сервер і база даних знаходяться на одній машині [10]. Таке налаштування просте і часто використовується в персональних або автономних додатках, де користувач безпосередньо взаємодіє з базою даних.

Наприклад: Електронна таблиця Microsoft Excel є чудовим прикладом однорівневої архітектури.

Все - користувацький інтерфейс, логіка програми та дані - обробляються в одній системі. Користувач безпосередньо взаємодіє з додатком, виконує такі операції, як обчислення або введення даних, і зберігає дані локально на тому ж комп'ютері.

Ця архітектура проста і добре працює для персональних, автономних додатків, де не потрібен зовнішній сервер або мережеве з'єднання.

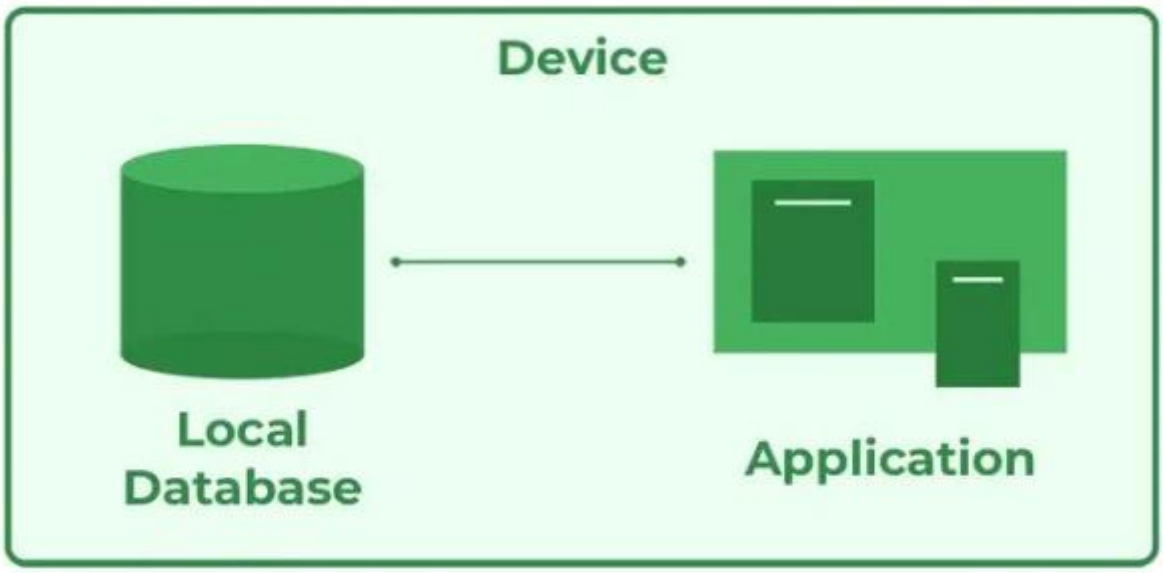


Рисунок 1.2 – 1-рівнева архітектура СУБД

До переваг можна віднести: *простоту архітектури:* 1-рівнева архітектура є найпростішою в налаштуванні, оскільки для її підтримки потрібна лише одна машина, *економічну ефективність*: для реалізації 1-рівневої архітектури не потрібне додаткове обладнання, що робить її економічно вигідною, *простота впровадження*: 1-рівнева архітектура легко розгортається, і тому вона здебільшого використовується в невеличких проектах.

Дворівнева архітектура (див. рис. 1.3) схожа на базову модель клієнт-сервер. Додаток на стороні клієнта безпосередньо взаємодіє з базою даних на стороні сервера. Серверна частина відповідає за обробку запитів і управління транзакціями. На стороні клієнта виконуються користувацькі інтерфейси та прикладні програми. Додаток на стороні клієнта встановлює з'єднання з серверною частиною для зв'язку з СУБД.

До прикладу: Система управління бібліотекою, що використовується в школах або невеликих організаціях, є класичним прикладом дворівневої архітектури.

- Клієнтський рівень (рівень 1): Це користувацький інтерфейс, з яким взаємодіє персонал бібліотеки або користувачі. Наприклад, вони можуть використовувати десктопний додаток для пошуку книг, їх видачі або перевірки термінів повернення.
- Рівень бази даних (Рівень 2): Сервер бази даних зберігає всі бібліотечні записи, такі як дані про книги, інформацію про користувачів і журнали транзакцій.

Клієнтський рівень надсилає запит (наприклад, на пошук книги) до рівня бази даних, який обробляє його і надсилає результат. Таке розділення дозволяє клієнтові зосередитися на користувацькому інтерфейсі, в той час як сервер займається зберіганням і пошуком даних.

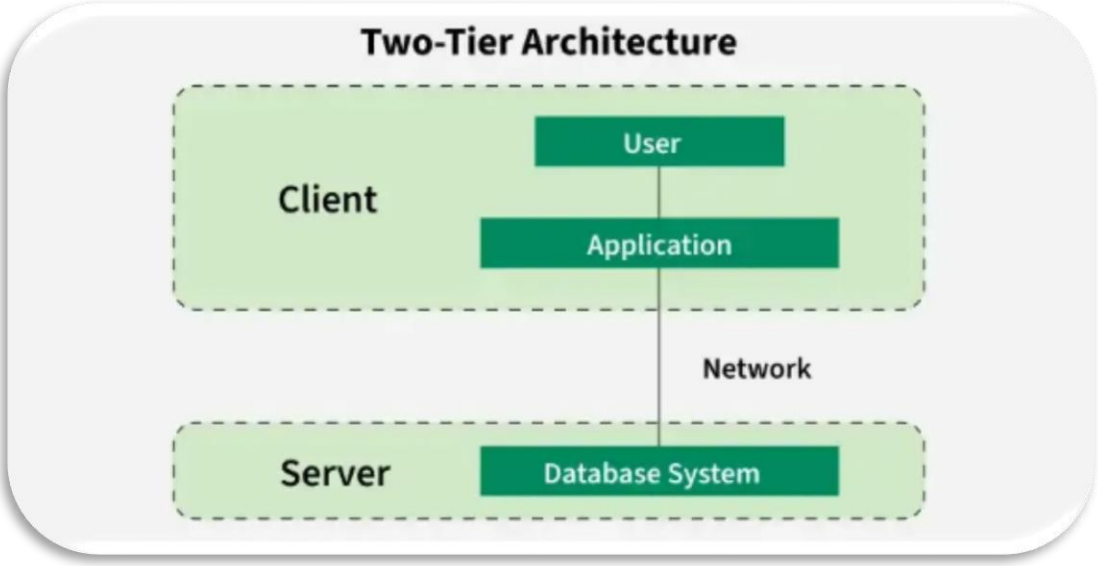


Рисунок 1.3 – 2-рівнева архітектура СУБД

Переваги: *легкий доступ*: 2-рівнева архітектура полегшує доступ до бази даних, що забезпечує швидкий пошук, *масштабованість*: доволі легко масштабувати базу даних, додаючи клієнтів або модернізуючи обладнання. *низька вартість*: 2-рівнева архітектура дешевша за 3-рівневу та багаторівневу архітектуру. *легке розгортання*: 2-рівневу архітектуру значно легше розгортати, ніж 3-рівневу.

У 3-рівневій архітектурі (див. рис. 1.4) між клієнтом і сервером є ще один рівень. Клієнт не взаємодіє безпосередньо з сервером. Замість цього він взаємодіє з сервером додатків, який далі взаємодіє з системою баз даних, а потім відбувається обробка запитів і управління транзакціями. Цей проміжний рівень діє як середовище для обміну частково обробленими даними між сервером і клієнтом. Цей тип архітектури використовується у випадку великих веб-додатків.

Простий приклад: Інтернет-магазин

- Користувач: Відвідує інтернет-магазин, шукає товар і додає його в кошик.
- Обробка: Система перевіряє наявність товару на складі, розраховує загальну ціну і застосовує будь-які знижки.
- База даних: Дані про товар, кошик та історія замовлень зберігаються в базі даних для подальшого використання.

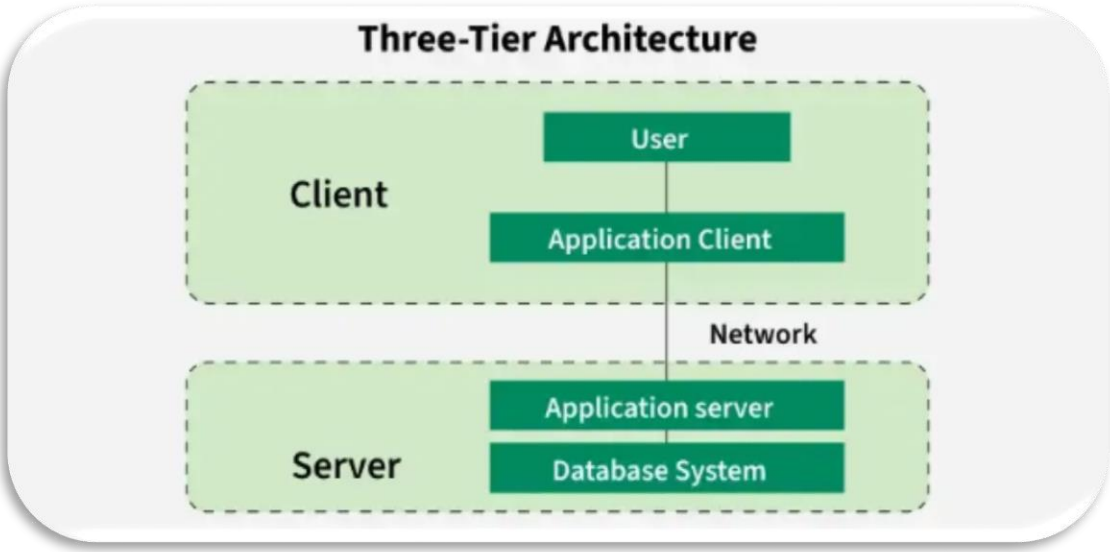


Рисунок 1.4 – 3-рівнева архітектура СУБД

Переваги 3-рівневої архітектури:

– *Покращена масштабованість*. Масштабованість підвищується завдяки розподіленому розгортанню серверів додатків. Тепер не потрібно встановлювати індивідуальні з'єднання між клієнтом і сервером.

– *Цілісність даних.* 3-рівнева архітектура підтримує цілісність даних. Оскільки між клієнтом і сервером є середній рівень, пошкодження даних можна уникнути або усунути.

– *Безпека.* 3-рівнева архітектура покращує безпеку. Цей тип моделі запобігає прямій взаємодії клієнта з сервером, тим самим зменшуючи доступ до несанкціонованих даних.

Недоліки 3-рівневої архітектури:

– *Більш складна.* 3-рівнева архітектура є більш складною в порівнянні з 2-рівневою архітектурою. Точки зв'язку також подвоюються в 3-рівневій архітектурі.

– *Складність взаємодії.* взаємодія такого роду ускладнюється через наявність проміжних рівнів.

Вибір правильної архітектури СУБД залежить від потреб системи в складності та масштабованості. Трирівнева структура є чудовим вибором, оскільки вона розділяє завдання на окремі ролі, роблячи систему більш організованою та легшою в управлінні. Здатність працювати з великими базами даних, підтримувати багато користувачів і забезпечувати безперебійну взаємодію між компонентами робить її ідеальною для сучасних систем, що розвиваються.

1.4 Моделі баз даних

Модель БД показує логічну структуру бази даних, включаючи взаємозв'язки та обмеження, які визначають, як можна зберігати дані та отримувати до них доступ [9]. Окремі моделі БД розробляються на основі правил

і концепцій тієї ширшої моделі даних, яку приймають розробники. Більшість моделей даних можна представити за допомогою супровідної схеми БД.

Існує багато типів моделей даних. Деякі з найпоширеніших включають:

– Ієрархічна модель;

– Реляційна модель;

– Мережева модель;

– Об'єктно-орієнтована модель;

– Модель «сутність-зв'язок»;

– Модель документів;

– Модель «сутність-атрибут-значення»;

– Зіркова схема;

– Об'єктно-реляційна модель, яка поєднує в собі дві моделі, що складають її назву;

Можна вибрати для опису БД будь-який з цих способів, залежно від кількох факторів. Найважливішим фактором є те, чи підтримує система управління базами даних певну модель. Більшість СУБД створено з урахуванням певної моделі даних і вимагають від користувачів прийняття цієї моделі, хоча деякі з них підтримують декілька моделей.

Крім того, різні моделі застосовуються на різних етапах процесу проектування бази даних. Високорівневі концептуальні моделі даних найкраще підходять для відображення взаємозв'язків між даними так, як їх сприймають люди. Логічні моделі на основі записів, з іншого боку, більш точно відображають способи зберігання даних на сервері.

Вибір моделі даних - це також питання узгодження пріоритетів щодо БД з сильними сторонами конкретної моделі, незалежно від того, чи включають ці пріоритети швидкість, зниження витрат, зручність використання або щось інше.

Розглянемо деякі з найпоширеніших моделей БД:

1) Реляційна модель (див. рис. 1.5), яка є найпоширенішою, сортує дані в таблиці, також відомі як відношення, кожна з яких складається зі стовпців і рядків. У кожному стовпчику перелічено атрибут відповідного об'єкта,

наприклад, ціна, поштовий індекс або дата народження. Разом атрибути у відношенні називаються доменом. Певний атрибут або комбінація атрибутів обирається як первинний ключ, на який можна посилатися в інших таблицях, і називається зовнішнім ключем.

Кожен рядок, який також називається кортежем, містить дані про конкретний екземпляр сутності, про яку йдеться, наприклад, про конкретного працівника.

Модель також враховує типи зв'язків між цими таблицями, включаючи зв'язки «один-до-одного», «один-до-багатьох» і «багато-до-багатьох».

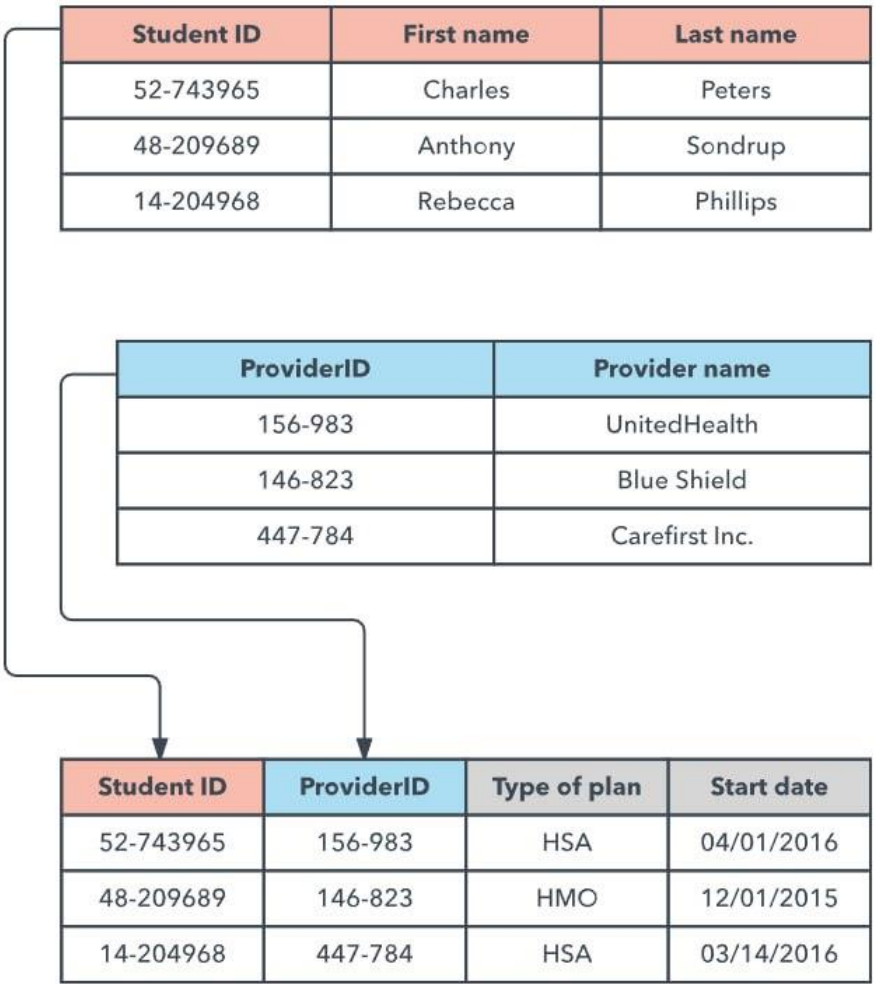


Рисунок 1.5 – Приклад реляційної моделі БД

У БД таблиці можуть бути нормалізовані або приведені у відповідність до правил нормалізації, які роблять базу даних гнучкою, адаптивною та

масштабованою. При нормалізації кожен фрагмент даних є атомарним, або розбитим на найменші корисні частини.

Реляційні бази даних зазвичай пишуться на мові структурованих запитів (SQL). Ця модель була представлена Е.Ф. Коддом у 1970 році.

2) Ієрархічна модель (див. рис. 1.6) організовує дані у деревоподібну структуру, де кожен запис має єдиного батька або корінь. Споріднені записи сортуються в певному порядку. Цей порядок використовується як фізичний порядок для зберігання бази даних. Ця модель добре підходить для опису багатьох реальних відносин.

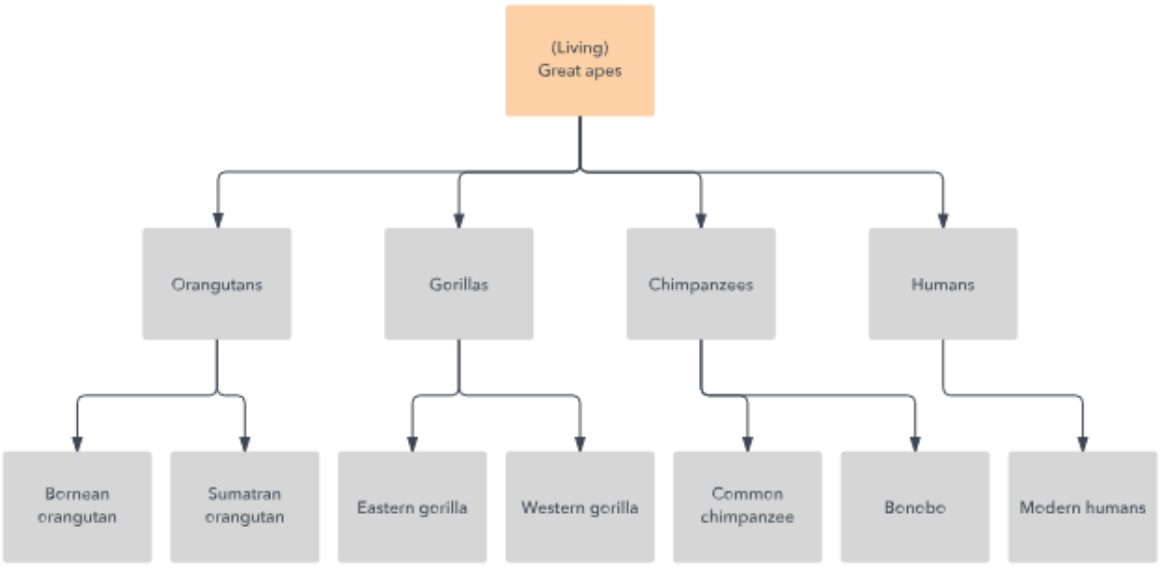


Рисунок 1.6 – Приклад ієрархічної моделі БД

Ця модель в основному використовувалася в системах управління інформацією IBM в 60-х і 70-х роках, але сьогодні вони рідко зустрічаються через певну операційну неефективність.

3) Мережева модель (див. рис. 1.7) будується на основі ієрархічної моделі, дозволяючи зв'язки «багато до багатьох» між пов'язаними записами, що передбачає наявність декількох батьківських записів. Заснована на математичній теорії множин, модель будується за допомогою наборів пов'язаних записів. Кожен набір складається з одного запису-власника або батьківського запису та одного або більше записів-членів або дочірніх записів. Запис може бути членом

або дочірнім у декількох множинах, що дозволяє цій моделі передавати складні взаємозв'язки.

Найбільшої популярності вона набула в 70-х роках після того, як була офіційно визначена Конференцією з мов систем даних (CODASYL).

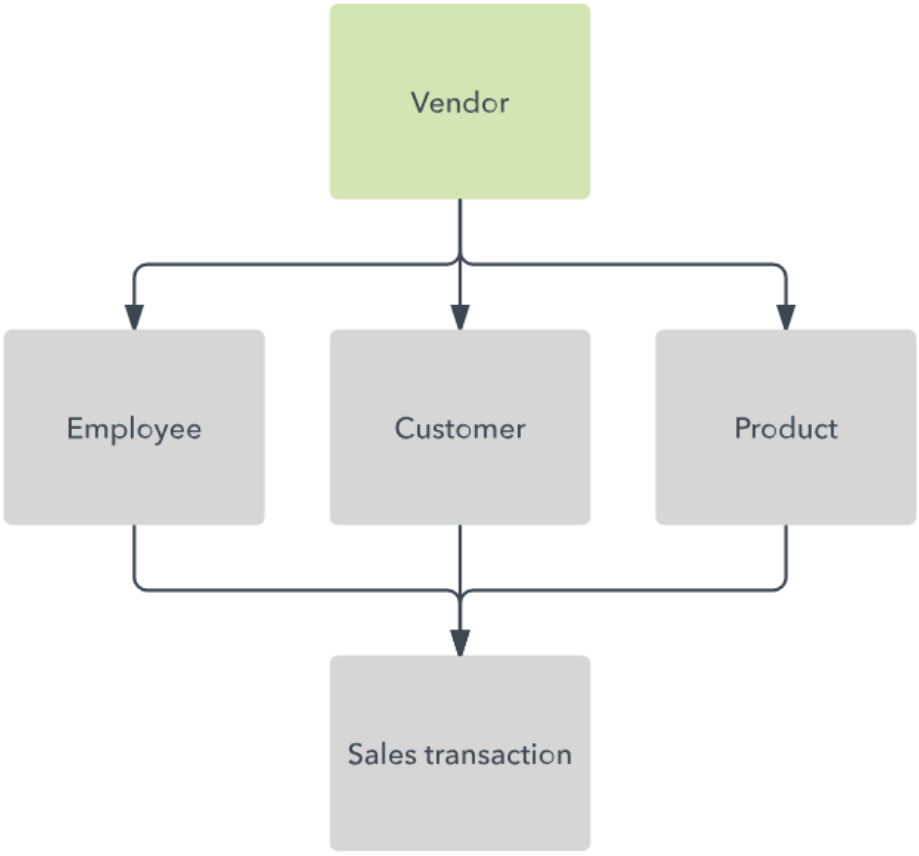


Рисунок 1.7 – Приклад мережевої моделі БД

4) Об'єктно-орієнтована модель (див. рис. 1.8) визначає базу даних як сукупність об'єктів або програмних елементів багаторазового використання з пов'язаними з ними функціями та методами. Існує кілька типів об'єктно-орієнтованих баз даних:

Мультимедійна БД включає в себе мультимедійні дані, такі як зображення, які не можуть зберігатися в реляційній базі даних.

Гіпертекстова БД дозволяє будь-якому об'єкту посилатися на будь-який інший об'єкт. Це корисно для організації великої кількості розрізнених даних, але не ідеально підходить для числового аналізу.

Об'єктно-орієнтована модель БД є найвідомішою моделлю пост-реляційної БД, оскільки вона включає таблиці, але не обмежується ними. Такі моделі також відомі як гібридні моделі баз даних.

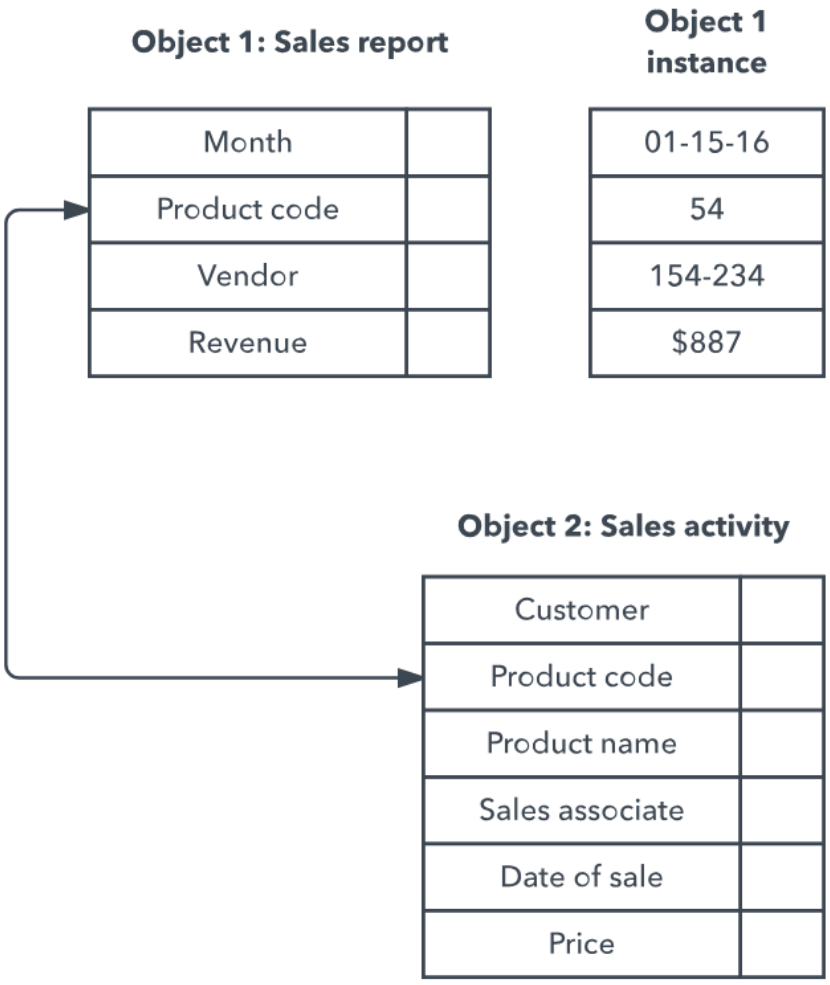


Рисунок 1.8 – Приклад об'єктно-орієнтованої моделі БД

5) Об'єктно-реляційна модель – це гібридна модель бази даних, що поєднує в собі простоту реляційної моделі з деякими розширеними функціональними можливостями об'єктно-орієнтованої моделі бази даних. По суті, вона дозволяє дизайнерам вбудовувати об'єкти у звичну структуру таблиць.

Мови та інтерфейси виклику включають SQL3, мови постачальників, ODBC, JDBC та власні інтерфейси виклику, які є розширенням мов та інтерфейсів, що використовуються в реляційній моделі.

6) Модель «сутність-зв'язок»: Ця модель (див. рис. 1.9) відображає взаємозв'язки між реальними сутностями так само, як і мережева модель, але вона не так безпосередньо пов'язана з фізичною структурою бази даних. Замість цього вона часто використовується для концептуального проектування бази даних.

Тут люди, місця та речі, про які зберігаються дані, називаються сутностями, кожна з яких має певні атрибути, що разом складають їхню предметну область. Кардинальність, або зв'язки між сутностями, також відображаються на карті.

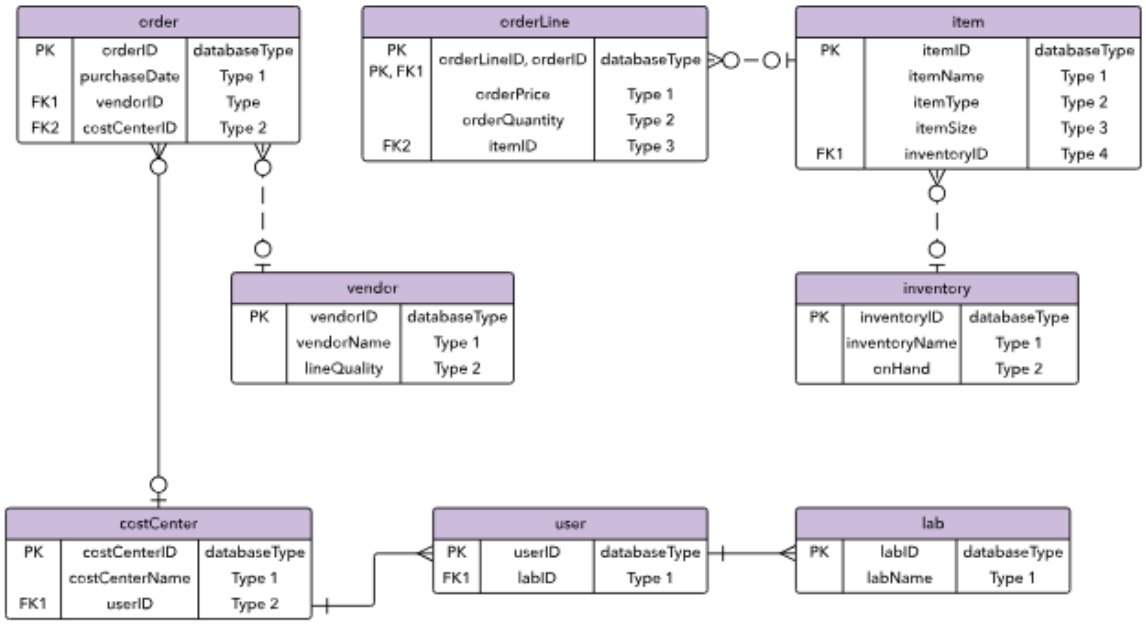


Рисунок 1.9 – Приклад моделі «сутність-зв'язок»

7) Моделі NoSQL: На додаток до об'єктної моделі бази даних та на противагу реляційній моделі з'явилися інші не-SQL моделі:

– Графова модель бази даних, яка є навіть більш гнучкою, ніж мережева модель, дозволяючи будь-якому вузлу з'єднуватися з будь-яким іншим;

– Багатозначна модель, яка відрізняється від реляційної моделі тим, що дозволяє атрибутам містити список даних, а не одну точку даних.

– Документна модель, яка призначена для зберігання та управління документами або напівструктурованими даними, а не атомарними даними.

Висновки за розділом 1

У першому розділі було розглянуто основні концепції баз даних, їхні роль і призначення, класифікацію та основні принципи роботи систем управління базами даних. Було визначено, що база даних є впорядкованою структурою даних, яка використовується для зберігання, обробки та управління інформацією.

Розглянуто ключові сфери застосування баз даних, включаючи банківську сферу, електронну комерцію, соціальні мережі, медицину, логістику, державні установи та інші галузі. Це підтверджує, що бази даних є невід'ємною частиною сучасного інформаційного простору та відіграють критично важливу роль у функціонуванні різних організацій.

Також було проаналізовано основні види СУБД, їхні компоненти, архітектуру та моделі баз даних. Особливу увагу приділено трирівневій архітектурі, яка забезпечує ефективне управління даними, безпеку та масштабованість.

Отже, бази даних та СУБД є ключовими елементами інформаційних систем, що дозволяють організаціям ефективно керувати великими обсягами даних, забезпечувати їхню безпеку, узгодженість та доступність. Подальші дослідження зосередяться на методах захисту баз даних, що є критично важливим аспектом для збереження конфіденційності та цілісності інформації.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ОСНОВНИХ ПІДХОДІВ ДО ЗАХИСТУ СИСТЕМ УПРАВЛІННЯ БАЗАМИ ДАНИХ

2.1 Класифікація та аналіз загроз

Безпека СУБД - це комплекс інструментів, засобів контролю та заходів, призначених для створення та збереження конфіденційності, цілісності та доступності даних.

Безпека СУБД повинна враховувати та захищати наступне [11]:

– Дані, які зберігаються в базах даних.

– Будь-які пов'язані з нею додатки.

– Фізичний сервер БД або віртуальний сервер БД та основне обладнання.

– Обчислювальну або мережеву інфраструктуру, яка використовується для доступу до СУБД.

Забезпечити надійний захист СУБД - це складна і відповідальна справа, яка охоплює всі аспекти технологій і практик інформаційної безпеки. Вона також нерозривно пов'язана зі зручністю використання СУБД. Чим більш доступною і зручною є СУБД, тим більш вразливою вона є до загроз безпеці; чим більш невразливою є СУБД до загроз, тим складніше до неї отримати доступ і користуватися нею. Цей парадокс іноді називають правилом Андерсона [12].

2.1.1 Важливість забезпечення інформаційної безпеки СУБД

За визначенням, витік даних - це порушення конфіденційності інформації у БД. Наскільки велика шкода від витоку даних може бути завдана підприємству, залежить від різних наслідків або факторів:

– Порушення прав інтелектуальної власності: Будь-яка інтелектуальна власність - комерційні таємниці, винаходи, запатентовані методи - може мати вирішальне значення для збереження конкурентних переваг на ринку. Якщо цю

інтелектуальну власність викрадено або викрито, то конкурентну перевагу може бути важко або ж неможливо зберегти та відновити.

– Пошкодження репутації бренду: Клієнти або партнери можуть не захотіти купувати товари чи послуги (або вести бізнес з компанією), якщо вони не впевнені, що можуть довіряти у захисті своїх даних.

– Доступність бізнесу: Деякі компанії не можуть продовжувати працювати, доки не буде усунуто порушення.

– Штрафи або покарання за недотримання вимог: Фінансові наслідки недотримання глобальних нормативних актів, таких як Стандарт безпеки даних індустрії платіжних карток (PCI DSS) [13], галузевих нормативних актів щодо захисту даних, таких як HIPAA [14], або регіональних нормативних актів щодо захисту даних, таких як Європейський загальний регламент захисту даних (GDPR) [15], можуть бути руйнівними, а штрафи в найгірших випадках перевищувати кілька мільйонів доларів за одне порушення.

– Витрати на усунення порушень та інформування клієнтів: На додаток до витрат на повідомлення клієнта про порушення, організація, що зазнала витоку даних, повинна оплатити судово-медичну експертизу та розслідування, кризовий менеджмент, сортування, ремонт пошкоджених систем та багато іншого.

2.1.2 Поширені загрози та виклики

Людська природа підказує, що перед тим, як залізти на дерево, ми скоріше візьмемо низько висячий плід. Оскільки суб'єкти загроз (врешті-решт) люди, це стосується і їх. У зв’язку з цим СУБД особливо вразливі до багатьох атак нижчого рівня. Від неправильної конфігурації до крадіжки облікових даних - ці сховища конфіденційної інформації можуть стати здобиччю навіть для кіберзлочинців-початківців.

Однак ця Ахіллесова п'ята також дозволяє легко захистити їх - якщо розуміти, що їм загрожує. Нижче наведені найбільш поширені типи та причини атак на СУБД [11, 16]:

Людські помилки - нещасні випадки, слабкі паролі, спільне використання паролів та інші нерозумні або необізнані дії користувачів залишаються причиною майже половини (49%) всіх витоків даних, про які повідомляється.

Інсайдерські (внутрішні) загрози (див. рис. 2.1) - це загрози безпеці від будь-кого з привілейованим доступом до бази даних. Як правило, виділяють три типи інсайдерів:

– Недбалий інсайдер (Negligent), який робить помилки, що роблять базу даних вразливою до атак.

– Зловмисний інсайдер (Malicious), який має намір завдати шкоди.

– Скомпрометований інсайдер (Compromised) - це співробітник, чиїм обліковим записом електронної пошти заволодів зловмисник за допомогою збору облікових даних, соціальної інженерії, фішингових електронних листів або шкідливого програмного забезпечення з метою викрадення інформації або здійснення шахрайських фінансових транзакцій.

Внутрішні загрози є однією з найпоширеніших причин порушень безпеки баз даних і часто є результатом того, що занадто багато співробітників мають доступ до привілейованих облікових даних користувачів.

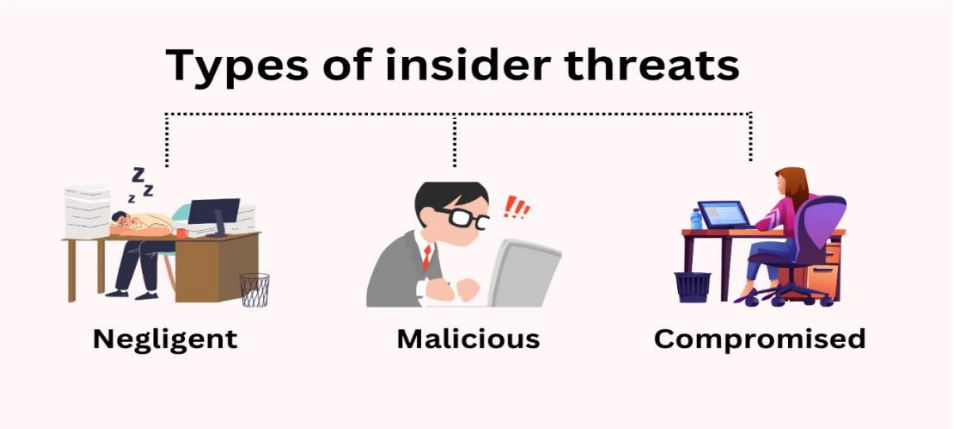


Рисунок 2.1 – Види інсайдерів

Атаки SQL-ін'єкцій. SQL-ін'єкції є однією із найпоширеніших загроз безпеці БД. Ця атака здійснюється шляхом введення запиту в SQL-форму, і якщо база даних інтерпретує результат як «істину», вона дозволяє доступ до бази

даних. Ці атаки зазвичай спрямовані на реляційні системи управління базами даних (СУБД), засновані на мові програмування SQL.

Бази даних, не засновані на SQL (NoSQL), не схильні до таких атак. Натомість, бази даних NoSQL стають мішенню для запитів, що надсилаються кінцевим користувачем, який використовує команди для виконання шкідливого програмного забезпечення.

Обидва методи однаково небезпечні, оскільки дозволяють обійти системи верифікації шляхом отримання облікових даних, а потім розкрити структуру та вміст бази даних. Успішна атака дає зловмиснику повну владу над усім, що міститься в базі даних.

Шкідливе програмне забезпечення (ШПЗ). Тип атаки із використанням ШПЗ призначений для реалізації вразливостей в мережі, надання доступу до бази даних або заподіяння їй шкоди. Ці вразливості пов'язані з незахищеними кінцевими точками в мережі, які можуть бути використані за допомогою різних атак.

Для ІТ-команди, щоб захиститися від атак ШПЗ, важливо визначити поверхню атаки мережі. Під поверхнею атаки мається на увазі кількість вразливостей в мережі, які можуть бути використані кіберзлочинцями.

Атаки на відмову в обслуговуванні (DoS/DDoS). Атака на безпеку бази даних з метою відмови в обслуговуванні (DoS) відбувається, коли сервер бази даних отримує більше запитів, ніж може обробити, що призводить до нестабільної роботи системи або її аварійного завершення. Ці помилкові запити можуть бути створені зловмисником і спрямовані на певну ціль. Обсяг фальшивих запитів перевантажує систему, що призводить до простою для жертви.

Розподілена атака на відмову в обслуговуванні (DDoS) (див. рис. 2.2) використовує ботнет (дуже велику мережу комп'ютерів) для створення величезного обсягу трафіку, який навіть найсучасніші системи безпеки не зможуть запобігти. Найкращим захистом від таких атак є використання хмарного

сервісу захисту від DDoS-атак, який може допомогти обмежити великий і підозрілий трафік.

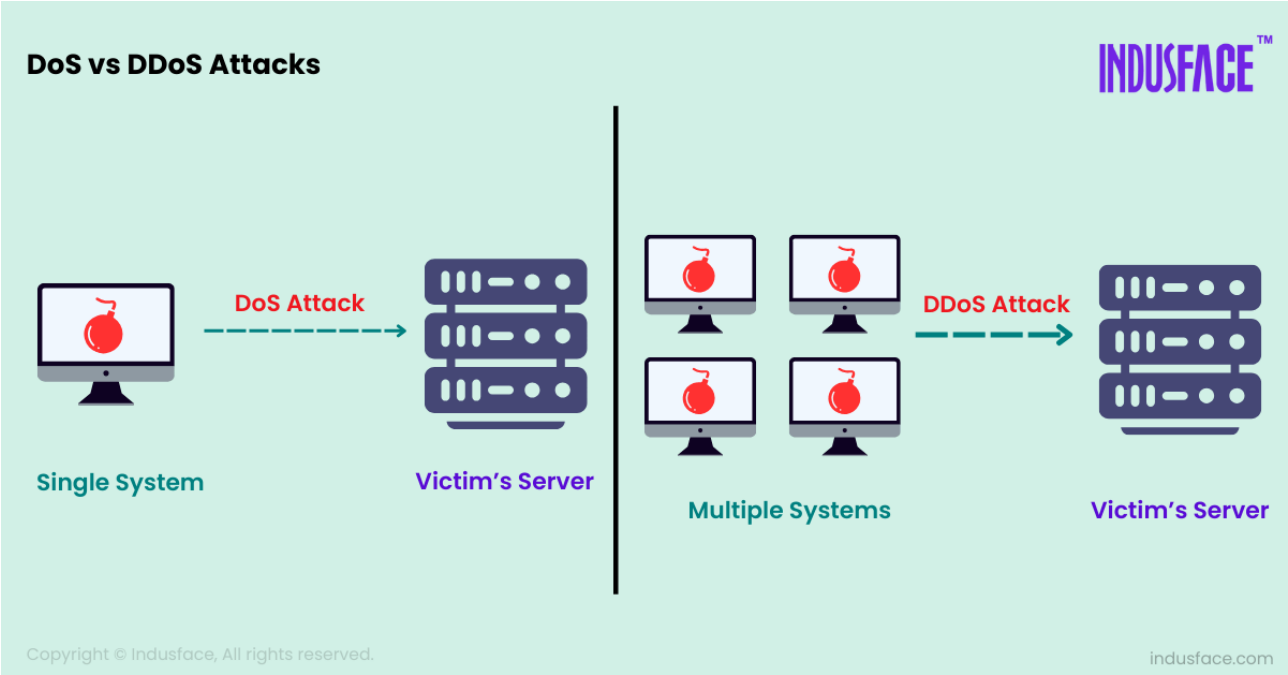


Рисунок 2.2 – Dos/DDos атаки

Вразливості резервних копій баз даних**.** Очевидно, що рекомендується регулярно створювати резервні копії БД, але часто багато з них залишаються незахищеними, що робить їх поширеною мішенню для зловмисників. Захист резервних копій особливо важливий для галузей, які зберігають важливу інформацію про клієнтів, таких як медичні заклади, банки та фінансові установи.

Щоб запобігти цій поширеній загрозі безпеці баз даних і поставити організацію в найкраще можливе становище, слід (окрім автоматизації регулярного резервного копіювання баз даних):

- Зашифрувати БД і всі зроблені резервні копії.
- Проводити регулярний аудит БД та їхніх резервних копій, щоб зафіксувати, хто мав доступ до цих даних.

Погане управління дозволами. Багато організацій не змінюють стандартні налаштування безпеки, встановлені під час первинного встановлення сервера БД. Лише кілька років тому 20% компаній навіть не змінювали стандартні паролі для привілейованих облікових записів [16]. Це робить їх вразливими до атак

зловмисників, які знають налаштування за замовчуванням і, що важливіше, як їх можна використати.

Злочинці можуть отримати дані для входу в привілейовані акаунти під час доступу до БД. Неактивні облікові записи також можуть становити ризик, якщо зловмисник знає про їхнє існування. Ось чому управління дозволами має бути на першому плані при розробці кібербезпеки підприємства в цілому, використовуючи протоколи нульової довіри для запобігання несанкціонованому доступу.

Іноді користувач може випадково отримати права доступу до БД, до яких він не повинен мати доступу. Це дає можливість хакерам атакувати таких користувачів за допомогою фішингових атак або інших тактик, які намагаються запустити ШПЗ на їхніх пристроях.

Кіберзлочинці також можуть спробувати захопити контроль над системою управління даними організації, змінюючи привілеї таким чином, щоб отримати доступ до бази даних у будь-який час. Рішення для запобігання втраті даних (DLP) можуть зробити багато для запобігання подібним випадкам.

Помилки в процесі аудиту. Неякісний аудит може стати чудовою можливістю для кіберзлочинців, зробивши БД невідповідною правилам безпеки даних. Організації зобов'язані реєструвати всі події, що відбуваються на сервері БД, і проводити регулярний аудит. Звичайно, такий аудит найкраще проводити за допомогою автоматизованих систем (АС).

Відсутність ефективних процедур аудиту збільшує шанси на успішну кібератаку. Однак також важливо, щоб будь-яке ПЗ для автоматизованого аудиту не впливало на загальну продуктивність БД.

Облікові дані (Credentials). Атаки соціальної інженерії, такі як фішинг або клікабельна реклама, можуть бути використані для отримання облікових даних для входу в систему, які зловмисник може використати для доступу до мережі та БД.

Згідно з останнім звітом Google «Горизонт загроз» [17], слабкі облікові дані (разом з неправильними конфігураціями) в хмарі були причиною трьох

чвертей вторгнень в мережу протягом першої половини 2024 року. Посилення вимог до облікових даних - це просте рішення, яке може запобігти більшості випадків компрометації даних, як пов'язаних з базами даних, так і не пов'язаних з ними.

Незашифровані дані. Шифрування даних є фундаментальним і найважливішим компонентом будь-якої політики кібербезпеки, особливо коли йдеться про захист фінансової інформації. Усі рахунки та фінансові дані, які зберігаються у фінансовій установі, повинні бути зашифровані. Таким чином, навіть якщо якісь дані будуть викрадені, шифрування гарантує, що вони стануть непридатними для використання. Насправді, принаймні один закон про кібербезпеку [18] вимагає шифрування даних для дотримання стандарту.

Багато різних загроз безпеці баз даних можуть становити значний ризик для конфіденційної інформації підприємства. Найпоширенішою загрозою для баз даних є SQL-ін'єкції, але такі атаки, як відмова в обслуговуванні та шкідливе програмне забезпечення, є не менш небезпечними. Навчання співробітників, використання шифрування та управління привілеями користувачів - одні з найкращих способів захистити базу даних від кібератак.

2.2 Ефективні стратегії запобігання загрозам інформаційної безпеки СУБД

Механізми забезпечення безпеки СУБД - це сукупність методів і технологій, що використовуються для захисту від зловмисних кібератак і несанкціонованого використання. Це комплексне завдання, яке поєднує в собі кілька дисциплін інформаційної безпеки - безпеку додатків, безпеку даних і безпеку кінцевих точок.

Метою безпеки СУБД є захист від неправомірного використання, пошкодження і вторгнення даних. Іншим аспектом безпеки є захист і зміцнення фізичного або віртуального сервера, на якому розміщена база даних, а також навколишнього комп'ютерного та мережевого середовища.

2.2.1 Передові методи захисту СУБД

Оскільки СУБД є мережевими, будь-яка загроза безпеці будь-якого компонента або частини мережевої інфраструктури також є загрозою для бази СУБД, і будь-яка атака на пристрій або робочу станцію користувача може загрожувати СУБД. Таким чином, безпека СУБД повинна виходити далеко за межі самої бази даних.

Оцінюючи безпеку СУБД у середовищі організації, щоб визначити головні пріоритети, слід розглянути кожну з наведених нижче сфер:

– Фізична безпека: Незалежно від того, чи знаходиться сервер БД локально, чи в хмарному дата-центрі, він має бути розміщений у безпечному середовищі з контрольованим кліматом. Якщо ж сервер БД знаходиться в хмарному дата-центрі, про це подбає хмарний провайдер.

– Адміністративний та мережевий контроль доступу: Практично мінімальна кількість користувачів повинна мати доступ до СУБД, а їхні права повинні бути обмежені до мінімального рівня, необхідного для виконання своїх обов'язків. Аналогічно, доступ до мережі повинен бути обмежений мінімально необхідним рівнем дозволів.

– Надійна автентифікація: Автентифікація СУБД - це процес підтвердження того, що користувачі або службові облікові записи, які намагаються підключитися до СУБД, є тими, за кого себе видають. Пов'язаним з цим процесом є авторизація, яка визначає, на основі підтвердженої ідентичності, які дозволи повинен мати обліковий запис на доступ до СУБД. Оскільки СУБД майже завжди є критично важливими системами, вони повинні мати надійну автентифікацію. Якщо підприємство володіє фінансовими та технічними можливостями, слід використовувати двофакторну автентифікацію, наприклад, поєднуючи пароль або PIN-код з чимось, що належить користувачеві, наприклад, токеном безпеки або номером мобільного телефону.

– Обліковий запис користувача та безпека пристрою: Потрібно завжди знати, хто має доступ до СУБД, а також коли і як використовуються дані. Рішення для моніторингу даних можуть попередити, якщо дії з даними є незвичними або виглядають ризикованими. Усі користувацькі пристрої, що підключаються до мережі, в якій знаходиться БД, повинні бути фізично захищеними (лише в руках належного користувача) і постійно підлягати контролю безпеки.

– Шифрування: Усі дані, включаючи дані в БД та облікові дані, повинні бути захищені найкращим у своєму класі шифруванням як у стані спокою, так і під час передачі. З усіма ключами шифрування слід поводитися відповідно до найкращих практик.

– Безпека програмного забезпечення СУБД: Використання останньої версії програмного забезпечення для управління базами даних та встановлення всіх оновлень, коли вони виходять.

– Безпека додатків та веб-серверів: Будь-який додаток або веб-сервер, який взаємодіє з БД, може бути каналом для атаки, тому його слід постійно тестувати на безпеку та керувати ним відповідно до найкращих практик.

– Безпека резервних копій: Всі резервні копії, копії або образи БД повинні підлягати такому ж (або не менш суворому) контролю безпеки, як і сама база даних.

– Аудит: Запис всіх входів до сервера БД та операційної системи, а також реєстрація всіх операції, які виконуються з конфіденційними даними. Регулярне проведення аудиту стандартів безпеки БД.

2.2.2 Контроль та політики

На додаток до впровадження багаторівневих засобів контролю безпеки у всьому мережевому середовищі, безпека СУБД вимагає встановлення правильних засобів контролю і політик для доступу до самої СУБД. До них відносяться

- Адміністративні засоби для керування встановленням, змінами та конфігурацією СУБД.
- Профілактичні засоби керування доступом, шифруванням, токенізацією та маскуванням.
- Засоби виявлення для моніторингу активності в СУБД та засоби запобігання втраті даних. Ці рішення дають змогу виявляти аномальні або підозрілі дії та сповіщати про них.

Політики безпеки СУБД повинні бути інтегровані у відповідністю із загальними бізнес-цілями, такими як захист критично важливої інтелектуальної власності, а також з політикою кібербезпеки та політикою безпеки хмарних технологій, і підтримувати їх. Варто бути переконаним, що призначено високо кваліфікованих працівників з питань аудиту та підтримки засобів контролю безпеки в організації та що політики доповнюють політики постачальника хмарних послуг в угодах про спільну відповідальність. Засоби контролю безпеки, програми навчання та підвищення обізнаності з питань безпеки, а також стратегії тестування на проникнення та оцінки вразливостей мають бути паралельно розроблені на підтримку офіційних політик безпеки.

2.2.3 Інструменти та платформи для захисту даних

Сьогодні широкий спектр вендорів пропонує інструменти та платформи для захисту даних. Повномасштабне рішення повинно включати в себе всі перераховані нижче можливості:

– Виявлення: Інструмент, який може сканувати і класифікувати вразливості у БД - незалежно від того, розміщені вони в хмарі або локально - і пропонувати рекомендації щодо усунення будь-яких виявлених вразливостей. Функції виявлення вразливостей часто потрібні для дотримання нормативних вимог.

– Моніторинг активності даних: Рішення повинно мати можливість моніторингу та аудиту всіх дій з даними у всіх базах даних, незалежно від того,

чи є розгортання локальним, хмарним або контейнерним. Воно повинно сповіщати про підозрілі дії в режимі реального часу, щоб мати змогу швидше реагувати на загрози. Також знадобиться рішення, яке може забезпечувати дотримання правил, політик і розподіл обов'язків, а також надавати інформацію про стан даних за допомогою комплексного і уніфікованого інтерфейсу користувача. Необхідно бути переконаним, що будь-яке обране рішення може генерувати звіти, необхідні для дотримання нормативних вимог.

– Можливості шифрування та токенізації: Шифрування забезпечує останню лінію захисту від компрометації даних у разі їхнього витоку. Будь-який інструмент, повинен мати гнучкі можливості шифрування, що дозволяють захистити дані в локальних, хмарних, гібридних або мультихмарних середовищах. Слід шукати інструмент з можливостями шифрування файлів, томів і додатків, які відповідають вимогам відповідності галузі підприємства, що може вимагати токенізації (маскування даних) або розширених можливостей управління ключами безпеки.

Оптимізація безпеки даних та аналіз ризиків: Інструмент, який може генерувати контекстну інформацію, поєднуючи інформацію про безпеку даних з розширеною аналітикою, дозволить легко здійснювати оптимізацію, аналіз ризиків та звітність. Потрібно надавати перевагу рішенню, яке може зберігати і синтезувати велику кількість історичних і нещодавніх даних про стан і безпеку СУБД, а також пропонувати можливості дослідження даних, аудиту та звітності за допомогою комплексної, але зручної для користувача інформаційної панелі самообслуговування.

2.3 Актуальні проблеми при організації захисту СУБД

Провівши аналіз засобів забезпечення безпеки даних, реалізованих у СУБД, архітектуру сховищ, інтерфейси систем, відомі вразливості та інциденти безпеки, можна виділити основні проблеми забезпечення надійного захисту системи:

– На високому рівні боротьби із загрозами інформації займаються тільки провідні корпорації, виробники промислових, великих СУБД;

– При створенні програмних продуктів розробники намагаються скористатися лише стандартними засобами захисту, що надаються за замовчуванням СУБД;

– Різновид масштабів та клас інформації, що зберігається, часто потребують індивідуальних підходів до безпеки;

– Переважно кожна СУБД використовує різні лінгвістичні конструкції для доступу до даних, що організовані на основі однієї моделі. [20]

Стрімкий розвиток ІТ-технологій призводить до змін у парадигмах захисту інформації. Як наслідок, це призводить до втрати актуальності як підходів, так і реалізації захисту. Еволюція засобів, методів і форм автоматизації процесів обробки інформації, а також масове застосування персональних комп'ютерів роблять інформацію набагато вразливішою.

Нижче наведено перелік основних чинників вразливості інформації [19]:

– ускладнення режимів роботи технічних засобів обчислювальних систем;

– обмін інформацією в локальних та глобальних мережах;

– розширення кола користувачів, що мають безпосередній доступ до ресурсів обчислювальної системи та масивів даних;

– збільшення обсягів та видів інформації, що обробляється за допомогою комп'ютерів, її збереження в електронному вигляді;

– накопичення інформації у сховищах та базах даних різнорідного призначення.

В свою чергу основними загрозами для інформації, яка зберігається, обробляється або ж передається в базах даних підприємства є [19]:

– використання прав доступу іншої особи;

– несанкціонована модифікація даних;

– зміна (підміна) програмного забезпечення;

– непродумані методики і процедури, що допускають змішування відкритої інформації та з обмеженим доступом в одному документі чи місці зберігання;

– підключення до кабельних мереж без вживання заходів захисту;

– введення кіберзлочинцями некоректних даних;

– шантаж;

– створення “лазівок” у системі;

– викрадення інформації, ПЗ та обладнання;

– відмова систем захисту;

– недостатній рівень компетенції та недотримання правил безпеки персоналом;

– надання доступу до засекречених даних третім особам;

– електронні перешкоди і радіація;

– руйнування даних у результаті відключення або перенапруження в мережі електроживлення;

– пожежі, повені, диверсії;

– фізичне пошкодження обладнання та елементів інфраструктури.

Також слід зазначити, що загрози бувають комбінованими та часто призводять до таких наслідків [19]:

– викрадення і фальсифікація даних;

– фінансовими збитками;

– порушення недоторканності особистих даних;

– втрата конфіденційності;

– втрата цілісності і доступності даних, що загрожує підприємству.

Проаналізувавши основні проблеми, що виникають при забезпеченні безпеки систем управління базами даних (СУБД), виявлено, що, попри наявність засобів захисту, рівень безпеки сучасних СУБД значною мірою залежить від індивідуальних підходів підприємств, а також від технологічного прогресу інформаційних технологій.

Основними факторами вразливості є ускладнення режимів роботи обчислювальних систем, збільшення обсягів інформації, її централізоване збереження та широке розповсюдження через локальні та глобальні мережі. Це робить інформацію вразливою до кібератак, несанкціонованого доступу, внутрішніх загроз та фізичних ризиків.

Крім того, виявлено ключові загрози, серед яких: викрадення облікових даних, несанкціоновані зміни в базах, зловмисні модифікації програмного забезпечення, недостатній рівень компетенції персоналу, електронні перешкоди, фізичні пошкодження інфраструктури та природні катаклізми.

Наслідками таких загроз можуть бути витік та фальсифікація даних, фінансові збитки, порушення конфіденційності інформації та втрата цілісності даних, що може призвести до серйозних загроз для підприємств.

З цього випливає, що питання безпеки баз даних потребує комплексного підходу, що включає технічні, організаційні та процедурні заходи для мінімізації ризиків і запобігання можливим атакам та втратам інформації.

Висновки за розділом 2

У другому розділі було проведено аналіз основних підходів до захисту систем управління базами даних, розглянуто потенційні загрози та вразливості, а також досліджено методи та засоби забезпечення безпеки.

Було визначено, що безпека СУБД є багатогранним процесом, який охоплює як технічні, так і організаційні заходи. Зокрема, розглянуто проблеми людських помилок, інсайдерських загроз, атак SQL-ін'єкцій, шкідливого програмного забезпечення та атак на відмову в обслуговуванні (DoS/DDoS). Крім того, особливу увагу приділено питанням управління доступом, автентифікації, шифрування та регулярного аудиту.

Проаналізовано сучасні методи забезпечення безпеки, такі як контроль доступу, моніторинг активності користувачів, застосування двофакторної

автентифікації, використання криптографічних механізмів та реалізація принципу "нульової довіри".

Також було розглянуто нормативно-правову базу України щодо захисту баз даних. Визначено, що законодавство забезпечує базові принципи кібербезпеки, захисту персональних даних та регулювання інформаційної безпеки, однак потребує постійного оновлення для відповідності сучасним викликам.

Таким чином, для ефективного захисту СУБД необхідно впроваджувати комплексний підхід, що включає регулярний моніторинг, обмеження доступу, використання передових технологій захисту, а також дотримання актуальних правових норм.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ БАГАТОФАКТОРНОЇ АВТЕНТИФІКАЦІЇ ДО СИСТЕМИ УПРАВЛІННЯ БАЗАМИ ДАНИХ

Мета розробки даного програмного модуля – забезпечити надійність процесу автентифікації користувачів до системи управління базами даних організації, адже отримання зловмисником несанкціонованого доступу – це суттєвий крок до порушення конфіденційності, цілісності та доступності інформації, яка зберігається в базі даних.

3.1 Автентифікація користувачів у СУБД: методи, особливості та порівняння

СУБД безперечно є дуже важливим об’єктом захисту практично на будь-якому підприємстві, оскільки в такій системі обробляється значна кількість критично важливої інформації. Саме тому варто розглянути наступні питання: Яким чином користувач отримує доступ до роботи з даними? Як відбувається процес автентифікації користувача в СУБД? Як СУБД реагує на НСД до своїх ресурсів?

Ґрунтуючись на праці «Види біометричної автентифікації та методи їх оцінки» [21], було розглянуто існуючі методи автентифікації при порівнянні за своїми властивостями, а саме:

– Слабка автентифікація;
– Двофакторна автентифікація;
– Біометрична автентифікація.

Щоб отримати доступ до будь-якої інформації в захищених системах необхідно пройти ідентифікацію та автентифікацію. Представлення користувача системі зазвичай виглядає як вказування свого імені (логіну) та паролю, який є підтвердженням, що користувач насправді є тим за кого себе видає.

Найпростіший спосіб автентифікації полягає у тому, що система порівнює передані користувачем дані у вигляді логіна та пароля з даними, які містяться в базі. Якщо дані співпадають – користувач отримує права відповідні до його ролі та рівню доступу в системі. Така автентифікація вважається слабкою і уразливою перед перехопленням та методом підбору паролю [21].

Більш складним методом автентифікації вважається спосіб за допомогою додаткового значення, окрім логіна та паролю. Прикладом такої автентифікації є додаткова перевірка IP адреси користувача, який намагається пройти автентифікацію. У такій ситуації IP адреса виступає додатковим значенням перевірки користувача, оскільки кожен комп'ютер в мережі має унікальний мережевий ідентифікатор. Але якщо користувачу доведеться повторно підключитися до мережі, пристрій отримає новий мережевий ідентифікатор, а це означає, що сервер вже буде мати розбіжність у даних, отриманих від користувача. Хоча логін і пароль буде співпадати, однак мережевий ідентифікатор буде відрізнятись від значення, яке отримав сервер при автентифікації користувача. В результаті система зареєструє таку спробу входу в систему потенційно небезпечною. Як наслідок таку незначну, на перший погляд, ситуацію потрібно буде розглянути спеціалістом з безпеки підприємства, що своєю чергою може відвернути від можливо справжньої спроби отримання НСД. [21]

Розглянемо двофакторну автентифікацію. Суть цього методу полягає у додатковому підтвердженні особистості користувача, який намагається отримати доступ до системи. Наприклад, після того як було введено логін та пароль, користувачу буде надісланий код на номер його мобільного телефону, який, зазвичай, складається с 4-6 символів. Як правило ці 4 або 6 символів є цифрами, після введення яких користувач повністю підтверджує свою особистість, отримує токени доступу та заходить до системи. Безперечно, ймовірність обходу такого методу автентифікації зводиться до нуля. Але якщо кіберзлодій викраде токени доступу, то зловмиснику вже не потрібно буде проходити автентифікацію. Система визначить таку автентифікацію як безпечну,

оскільки користувач повністю пройшов автентифікацію і вважається безпечним для системи. Через деякий час, коли система знову починає перевіряти токени доступу та при умові, що вони являються валідованими, надає користувачу новий токен доступу для продовження роботи в системі.

Найбільш фінансово витратним, однак своєю чергою найбільш ефективним методом автентифікації для підприємства є біометрична система ідентифікації користувача. Безперечно, як показує статистика такий метод є безпечнішим на 86-93% ніж розглянуті раніше, оскільки характеристики такої системи вражають. Ймовірність відмови у доступі такої системи дорівнює 2-6%. А ймовірність помилкової ідентифікації користувача дорівнює 0,0001% [21].

Може виникнути питання: А яким чином уникнути такої ситуації, коли спеціаліст та система безпеки «шукають не там». Звичайно, одразу на думку приходить зробити так, щоб під час перевірки системою токенів доступу, до уваги був піднесений додатковий параметр пристрою користувача. Якщо користуватись лише методом двофакторної автентифікації, то це означає, що користувачу потрібно буде кожен певний період отримувати код на мобільний для проходження автентифікації. Такий спосіб є безпечним, але не є зручним для самого користувача. Варто приділити цьому питанню увагу.

Узагальнене порівняння згаданих вище методів автентифікації представлену у таблиці 3.1.

Таблиця 3.1.

Порівняння методів автентифікації

Метод автентифікації	**Характеристика методу та рівень захисту**	**Приклади застосування**
Багаторазові паролі	Потрібно ввести правильний пароль, який використовується протягом певного періоду часу. Рівень захисту: низький	WEB-сайти, мережеві служби, смартфони, ПК, електронні замки.
Одноразові паролі	Потрібно ввести правильний пароль, який використовується тільки для одного входу. Рівень захисту: середній	Електронний квиток, відновлення облікового запису

Продовження табл. 3.1

Двофакторна автентифікація	Окрім того, що користувачу потрібно ввести правильний пароль, йому ще потрібно підтвердити, що це саме він намагається виконати вхід, надавши інформацію, яка йому відома(код на мобільному телефоні або в електронній пошті). Рівень захисту: високий	Вхід до мобільного застосунку банку, Вхід до облікового запису соцмереж тощо.
Біометрична автентифікація	Окрім того, що користувачу потрібно ввести правильний пароль, йому ще потрібно підтвердити, що це саме він намагається виконати вхід, підтвердивши це особливістю свого фізичного тіла (відбиток пальця, сканер ока, детектор голосу). Рівень захисту: високий	Розблокування мобільного телефону (TouchID, FaceID).

Проаналізувавши методи простої автентифікації, потрібно відзначити, що у кожного з яких є як і переваги, так і недоліки. Слід розуміти, що ці недоліки можуть негативно вплинути на забезпечення безпеки інформації, зокрема на вразливість системи до процесу автентифікації. Щоб уникнути надання зловмисникові потенційної можливості для порушення конфіденційності, доступності та цілісності в системі, слід дотримуватися рекомендацій по створенню і зберіганню паролів та використовувати схеми більш надійної автентифікації.

3.2 Використання JSON Web Token при автентифікації у СУБД

JSON Web Token (JWT) - це відкритий стандарт (RFC 7519 [22]), який визначає компактний і автономний спосіб безпечної передачі інформації між сторонами у вигляді об'єкта JSON [23]. Для використання JWT достатньо базового розуміння мови програмування Java Script.

Нижче наведено дві основні сфери, в яких веб-токени JSON можуть бути корисними:

- Авторизація: Це найпоширеніший сценарій використання JWT. Після того, як користувач увійшов в систему, кожен наступний запит буде включати JWT, дозволяючи користувачеві отримати доступ до маршрутів, сервісів і ресурсів, які дозволені з цим токеном. Єдиний вхід - це функція, яка широко використовує JWT сьогодні через її невеликі накладні витрати і можливість легко використовувати в різних доменах.
- Обмін інформацією: JWT - це хороший спосіб безпечної передачі інформації між сторонами. Оскільки JWT можна підписувати - наприклад, за допомогою пар відкритих/закритих ключів – то можна бути впевненим, що відправники є тими, за кого себе видають. Крім того, оскільки підпис обчислюється за допомогою заголовка і корисного навантаження, також легко можна перевірити, що вміст не був підроблений.

Розглянемо структуру JWT:

У своєму компактному вигляді JSON Web Token складається з трьох частин, розділених крапками (.) [24], а саме:

- Заголовок (Header);
- Корисне навантаження (Payload);
- Підпис (Signature).

Отже, JWT зазвичай виглядає наступним чином (див. рис. 3.1).

```
xxxxx.yyyyy.zzzzz
```

Рисунок 3.1 – Приклад JWT

Усі JSON веб-токени є зашифровані та представляють собою засіб автентифікації для кожного запиту від клієнта до серверу. Вони генеруються, базуючись на секретному ключі, який в свою чергу зберігається на сервері. Токен доступу зберігається у клієнта та використовується при необхідності автентифікації будь-якого запиту.

При спробі підробити дані, які містить в собі зашифрований токен, кіберзлочинець зіткнеться з тим, що такий токен стане не валідним, оскільки його сигнатура не буде відповідати початковим значенням. Можливість згенерувати нову сигнатуру у зловмисника відсутня, тому що секретний ключ шифрування знаходиться на сервері. Також варто зазначити, що токен зберігається не в локальній пам’яті пристрою клієнта, а в пам’яті клієнтського додатку.

При автентифікації, коли користувач успішно увійшов в систему, використовуючи свої облікові дані, буде повернуто JWT у форматі JSON. Оскільки токени є обліковими даними, необхідно бути дуже обережними, щоб запобігти проблемам безпеки. Загалом, не слід зберігати токени довше, ніж потрібно.

Кожного разу, коли користувач хоче отримати доступ до захищеного маршруту або ресурсу, агент користувача повинен відправити JWT, як правило, в заголовку авторизації, використовуючи схему пред'явника.

JWT поділяються на два види та мають обмеження в часі існування:

– Токени доступу, які використовуються для автентифікації запитів та зберігання додаткової інформації про користувача.

– Токени повтору, які видаються сервером у результаті успішної автентифікації і використовуються у подальшому для створення нових токенів.

Такі токени слід зберігати в БД, тому що вони використовуються для обліку доступу та інвалідації викрадених токенів. Таким чином серверу буде відомо яким користувачам дозволено автентифікуватись.

Все це працює наступним чином:

1. Користувач ідентифікує себе, передаючи свій логін, пароль та унікальний ідентифікатор пристрою, з якого здійснює вхід у систему.

2. Сервер перевіряє коректність введеного логіна та паролю.

3. Якщо введені дані вірні, то сервер записує сесію в БД та створює токен доступу.

4. Сервер відправляє клієнту відповідні токени, створені на основі даних користувача.

5. Клієнт зберігає отримані від сервера токени.

Варто зауважити, що процес запису сесії до СУБД має мати свої заходи безпеки. Сервер має перевірити скільки повторних сесій є у користувача. Наприклад, якщо у користувача більше ніж 1 сесія, на це варто звернути увагу. При спробі встановити ще одну нову сесію, сервер повинен видаляти попередню. Таким чином можна уникнути підозрілих дій зі сторони користувача, оскільки система обнулить всі сесії окрім останньої.

Запис унікального ідентифікатора пристрою також має велике значення. При ситуації коли кіберзлочинець отримав токен доступу, сервер перевіряє час існування такого токена. Якщо час існування вичерпаний, сервер звертається до токена повтору, який надсилає зловмисник разом зі своїм ідентифікатором пристрою. Звісно, цей ідентифікатор буде відрізнятись з ідентифікатором, який був переданий користувачем при створенні токенів, тому СУБД видалить таку сесію та зафіксує спробу несанкціонованого оновлення токенів. Коли справжній користувач зробить спробу отримати доступ до СУБД, то через відсутність токену, система відхилить таку спробу та направить такого користувача на повторну автентифікацію.

Такий метод автентифікації має переваги над іншими, тому що унікальний ідентифікатор кожного пристрою не може повторюватись. У порівнянні з автентифікацією, де унікальним значенням виступає ідентифікатор мережевого рівня, тобто IP адреса, використання ідентифікатору пристрою має переваги в тому, що при необхідності використання VPN сервісів, користувачу СУБД не потрібно буде знову автентифікуватись в системі та проходити відповідний етап. Повторна автентифікація відбудеться тільки при спробі зловмисником перехопити токен доступу або ж при використанні користувачем нового пристрою, що призведе до зміни його ідентифікатору. Така ситуація трапляється на багато рідше ніж зміна IP адреси пристрою, яким орудує користувач. Простим прикладом є ситуація, коли підприємство змінює інтернет провайдера. Тоді всім

користувачам системи необхідно буде знову автентифікуватися, що може призвести до значних затримок у функціонуванні підприємства.

Перевагою JWT є можливість використання такого способу підприємствами низького рівня, оскільки процедура не є затратною і не потребує у своєму використанні високого рівня спеціаліста.

3.3 Розробка програмного модуля

Програмний модуль базується на перевагах технології JSON Web Token, в наслідок чого процедура створення модуля є корисною та відносно не складною.

Для вирішення поставленої задачі було обрана мова JavaScript, яка містить усі необхідні функції.

Оскільки JS є мовою, що інтерпретується, дуже часто вона позиціонується як мова сценаріїв, а не як мова програмування, при цьому мається на увазі, що мови сценаріїв простіші і більшою мірою орієнтовані не на програмістів, а на звичайних користувачів. Справді, за відсутності контролю типів JS можна допускати не критичні помилки, які допускають недосвідчені програмісти.

Спочатку JS розроблявся з метою на вбудовування в будь-які додатки і надання можливості виконувати сценарії. З найперших днів вебсервери компанії Netscape включали в себе інтерпретатор JS, що дозволяло виконувати JS сценарії на стороні сервера. Аналогічним чином на додаток до Internet Explorer корпорація Microsoft використовує інтерпретатор JS в своєму веб-сервері IIS і в продукті Windows Scripting Host. Компанія Adobe задіяла похідну від JS моу для управління своїм програвачем Flash-файлів. Компанія Sun також вбудувала інтерпретатор JS дистрибутив Java 6.0, що істотно полегшує можливість вбудовування сценаріїв в будь-який Java-додаток.

Почати слід із визначення алгоритму розробки, представленого у вигляді блок-схеми (див. рис. 3.2).

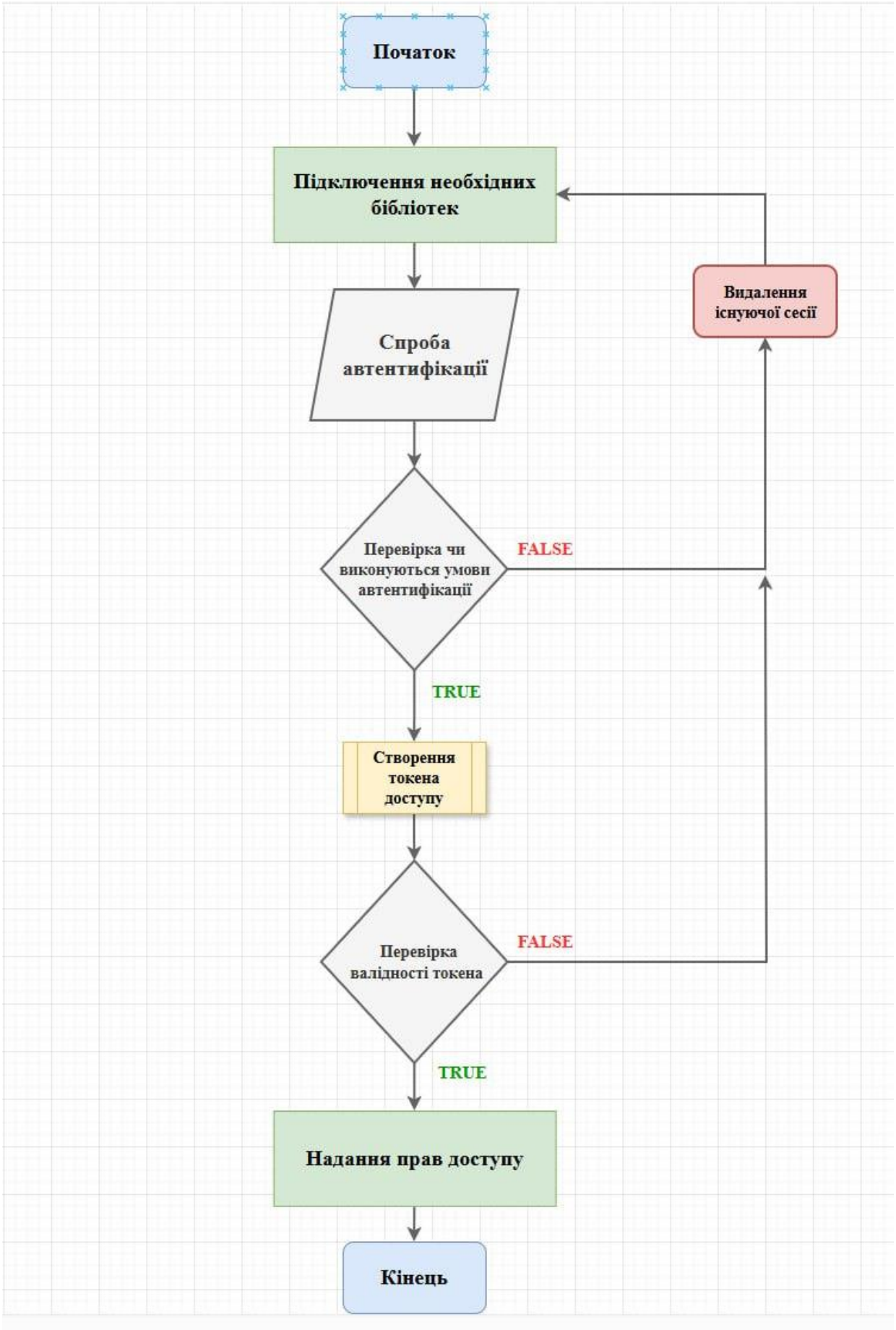


Рисунок 3.2 – Алгоритм розробки

Програма працює за наступним алгоритмом:

1. Система отримує від користувача логін, пароль та ідентифікатор пристрою:
2. Сервер перевіряє коректність отриманих даних:
   a. Якщо дані не коректні, система повертає користувача до етапу автентифікації;
   b. Якщо дані коректні, система створює токен доступу.
3. Сервер перевіряє валідність токену доступу через деякий час:
   a. Якщо токен не валідний, система повертає користувача до етапу автентифікації;
   b. Якщо токен валідний, система надає доступ.

Користувач представляє себе в системі за допомогою логіна, пароля та унікального ідентифікатора пристрою (див. рис. 3.3).

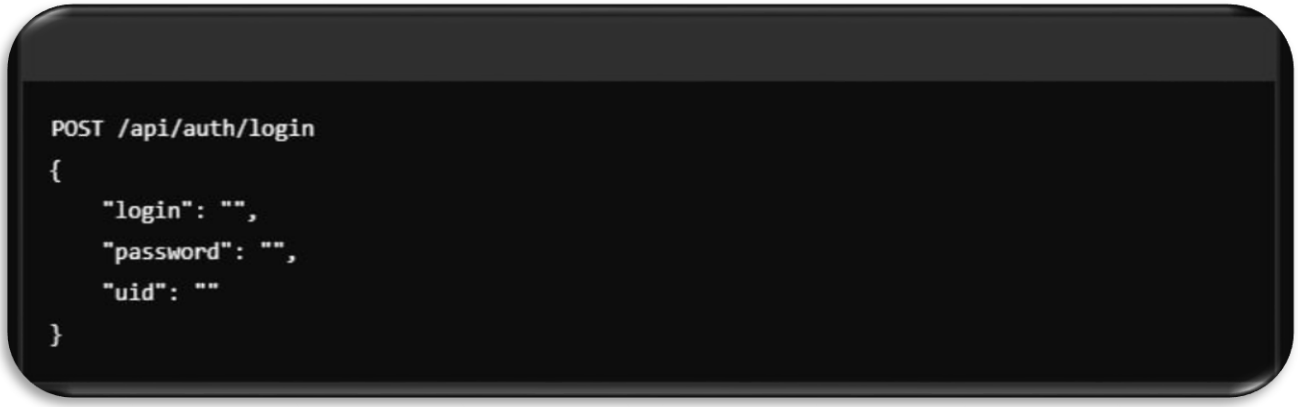


Рисунок 3.3 – Елементи автентифікації користувача

Після визначення основних елементів автентифікації користувача потрібно задати формат даних для входу (див. рис. 3.4).

```
static get validationRules () {
    return {
        body: {
            password: new RequestRule(AuthModel.schema.password, { required: true }),
            login: new RequestRule(AuthModel.schema.login, { required: true }),
            uid: new RequestRule(AuthModel.schema.uid, { required: true })
        }
    }
}
```

Рисунок 3.4 – Функція задання формату даних входу

Коли користувач вводить логін та пароль, необхідно перевірити коректність введених даних (див. рис. 3.5).

```
function checkPassword(password, hash) {
    assert.string(password, { notEmpty: true })
    assert.string(hash, { notEmpty: true })

    return new Promise((resolve, reject) => {
        bcrypt.compare(password, hash, (error, result) => {
            if (error) return reject(new AppError(error))
            if (!result) return reject(new AppError({ ...errorCodes.INVALID_PASSWORD }))
            return resolve(result)
        })
    })
}
```

Рисунок 3.5 – Фрагмент коду - перевірка введених даних

Далі буде відбуватися формування токену доступу (див. рис. 3.6).

```
const newRefreshSession = new RefreshSessionEntity({
    userId: user.id,
    ip: ctx.ip,
    ua: ctx.headers['User-Agent'],
    uid: ctx.body.uid,
    expiresIn: refTokenExpiresInMilliseconds
})
```

Рисунок 3.6 – Фрагмент коду - формування даних для створення токену

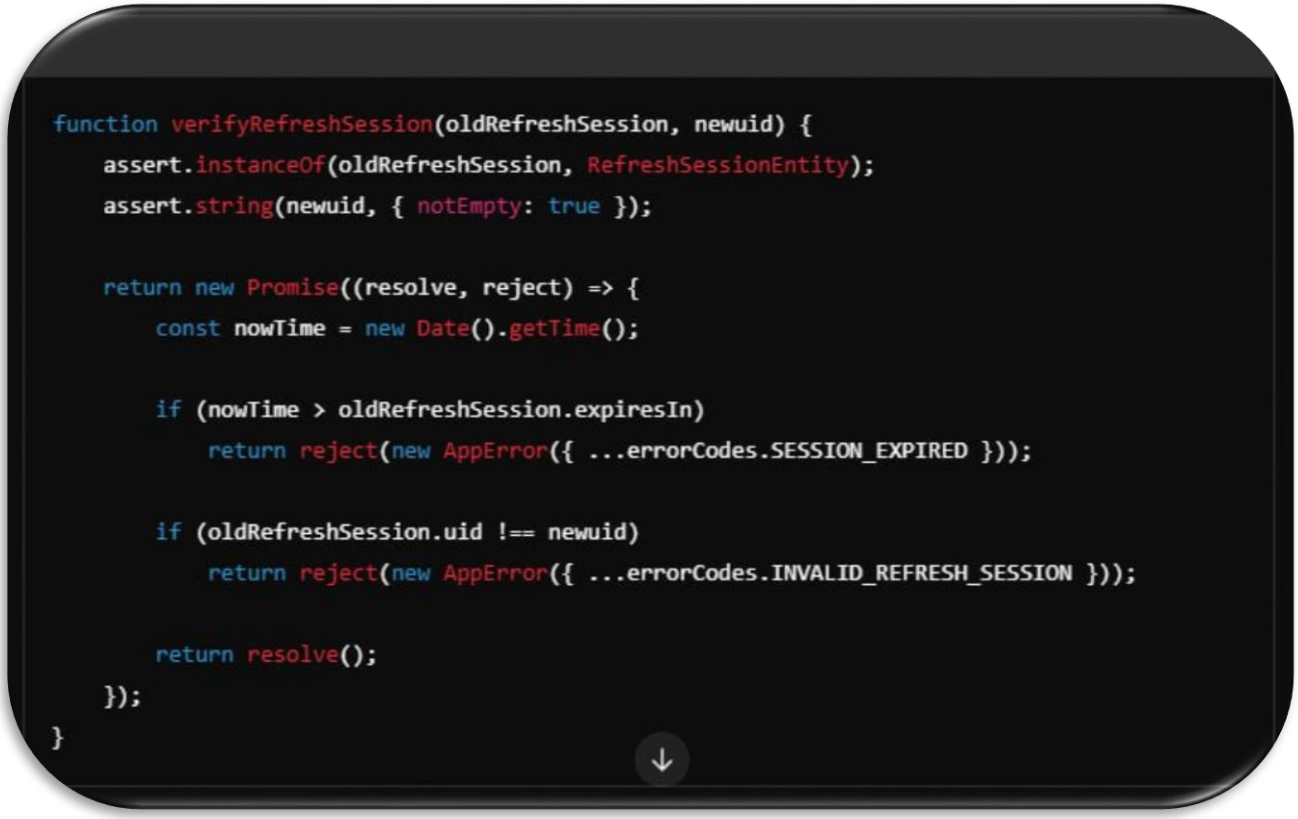
Після цього, на основі отриманих даних, формується сам токен доступу (див. рис. 3.7).

```
function makeAccessToken(userEntity) {
    assert.object(userEntity, { required: true });

    let config = {
        payload: {
            tokenType: type,
            username: userEntity.name,
            userRole: userEntity.role,
            email: userEntity.email,
            iss
        },
        options: {
            algorithm: 'HS512',
            subject: userEntity.id,
            expiresIn
        }
    };
}
```

Рисунок 3.7 – Фрагмент коду - формування токену доступу

Під час цього процесу сервер перевіряє максимальну кількість сесій. Як було зазначено раніше, якщо таких сесій більше ніж одна, то сервер має видалити попередній токен доступу(якщо такий є) та створити новий (див. рис. 3.8).



```
function verifyRefreshSession(oldRefreshSession, newuid) {
    assert.instanceOf(oldRefreshSession, RefreshSessionEntity);
    assert.string(newuid, { notEmpty: true });

    return new Promise((resolve, reject) => {
        const nowTime = new Date().getTime();

        if (nowTime > oldRefreshSession.expiresIn)
            return reject(new AppError({ ...errorCodes.SESSION_EXPIRED }));

        if (oldRefreshSession.uid !== newuid)
            return reject(new AppError({ ...errorCodes.INVALID_REFRESH_SESSION }));

        return resolve();
    });
}
```

Рисунок 3.8 – Фрагмент коду - перевірка на наявність існуючих сесій та унікального цифрового ідентифікатора пристрою

Для досягнення безпеки максимальна кількість існуючих сесій користувача має бути не більше ніж одна. Інакше попередні сесії будуть видалені. У разі успіху користувач зберігає на своєму пристрої токен доступу, який виглядає в наступному форматі (див. рис. 3.9).

```
Set-Cookie: refreshToken='c84f18a2-c6c7-4850-be15-93f9cbaef3b3';

{
    body: {
        accessToken: 'eyJhbGciOiJIUzUxMiIsI...',
        refreshToken: 'c84f18a2-c6c7-4850-be15-93f9cbaef3b3'
    }
}
```

Рисунок 3.9 – Загальний вигляд токенів доступу

На основі отриманих даних від користувача, було створено токен доступу. Це робиться для того, щоб користувач системи не проходив етап автентифікації повторно, коли система через деякий час дасть запит на перевірку валідності цього токена.

Використання токена повтору: якщо користувач автентифікувався в системі і вже має токен доступу, то такий токен необхідно перевірити на валідність. Сервер запитує токен повтору, який зберігає користувач. Потім видаляє старий сеанс та перевіряє початковий унікальний ідентифікатор пристрою користувача.

Для початку потрібно перевірити існуючі сесій, кількість яких не повинна перевищувати одиниці (див. рис. 3.10).

```
async function _isValidSessionsCount(userId) {
    assert.validate(userId, UserModel.schema.id, { required: true })

    const existingSessionsCount = await RefreshSessionDAO.baseGetCount({ userId })
    return existingSessionsCount < MAX_REFRESH_SESSIONS_COUNT
}
```

Рисунок 3.10 – Фрагмент коду - перевірка кількості активних сесій користувача

Це важливий момент для вирішення поставленого завдання, оскільки саме тут система вирішує продовжити надання доступу користувачу або ж при ситуації, коли першопочатковий ідентифікатор не співпадає з ідентифікатором, отриманим повторно через деякий час, обмежити доступ до системи та створити запит на повторну автентифікацію користувача.

У випадку успішного співпадіння ідентифікатора, створюється, за раніше описаним способом, новий токен повтору, який є таким самим, як при створенні першого разу. Важливо щоб створений токен співпадав з першостворeним токеном повтору. Інакше знову ж таки система зафіксує таку спробу автентифікації як потенційно небезпечною.

Суть захисту, який представляє ця розробка, полягає в наступному:

- Зловмисник отримав токени доступу.
- Закінчився час існування такого токену.
- Зловмисник відправляє системі свій ідентифікатор пристрою (uid) для отримання повторного токену доступу.
- Сервер перевіряє наданий користувачем ідентифікатор. Звісно, співпадіння не виявляється.
- Сервер видаляє всі сесії користувача та створює нову сесії, в якій потрібно автентифікуватись

3.4 Тестування програмного модуля в імітованому середовищі

Розглянемо роботу розробленого модуля на прикладі. Існує СУБД, в якій необхідно провести процес автентифікації у відповідному вікні. Виглядає воно наступним чином. Існує два поля. Одне з них – це логін. Друге – пароль.

Після введення коректного логіна і пароля СУБД надає користувачу доступ до інформації. Токен доступу має обмежений час існування. Через 30 хвилин користування СУБД потрібно буде отримати оновлений токен для подальшого користування системою. Якщо користувач не буде змінювати пристрій з якого

працює, то це буде виглядати як швидке оновлення сторінки з можливістю продовжити роботу в СУБД.

Тепер необхідно розглянути ситуацію, коли зловмисник перехопив токен доступу та намагається отримати доступ. Складність системи може зіграти на користь, тому що за час, поки кіберзлодій буде шукати інформацію в БД, сплине час дії токену та сервер буде потребувати його оновлення. Сервер перевірить, що сесії користувача існує вже дві, а не одна. Та зі спробою отримання унікального ідентифікатора буде виявлено, що друга сесія має не коректне значення. Після чого відбудеться оновлення сторінки на пристрої зловмисника і відповідно буде перехід на сторінку з автентифікацією. А зі сторони справжнього користувача буде таке ж швидке оновлення сторінки з можливістю подальшої роботи в БД.

Для тестування було використане ПЗ Fiddler, за допомогою чого було перехоплено трафік та викрадено токен доступу. Та розширення GetThisCookies для спроби підмінити викрадений токен вже на іншому пристрої.

При спробі підміни, було отримано відмову у вигляді помилки (див. рис. 3.11).

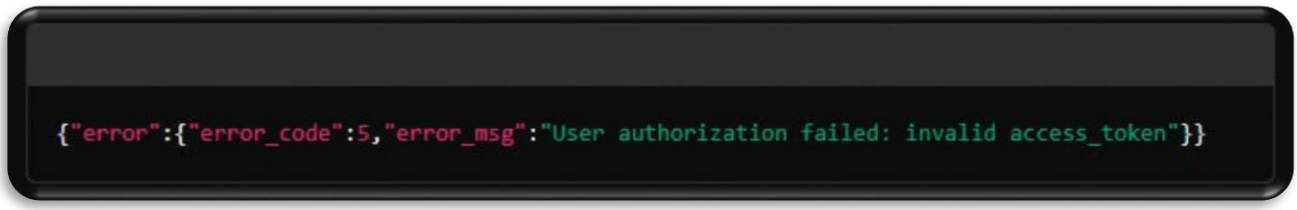


Рисунок 3.11 – Зображення помилки невалідної автентифікації

Це означає, що токен, який був представлений користувачем системі, не є валідним, що підтвердило дієвість розробленого модуля.

3.5 Переваги та рекомендації щодо використання розробленого модуля забезпечення захисту

Основна перевага такої розробки є простота в своєму використанні та в незначних затратах підприємства в підтримці розробленого модуля. Також варто

відмітити випадок, коли зловмисник розуміє, що в системі існує перевірка ідентифікатору пристрою. Та навіть знаючи це, шанс на проникнення в систему не збільшується. Вгадати або підібрати такий ідентифікатор буде практично неможливим або займе у зловмисника значного обсягу часу.

При розробці модуля була використана динамічна, об’єктно-орієнтована мова, що набирає популярність в усіх сферах інноваційних технологій, Java Script. Тому спеціалісти, які можуть у подальшому використовувати цей програмний модуль, зможуть з легкістю модифікувати його в своїх цілях. Оскільки обрана мова програмування є однією з найпопулярніших мов, яка в наш час неухильно розвивається. Її використання у вирішені завдання є цілком коректним.

Якщо порівняти наведений модуль захисту СУБД з методом двофакторної автентифікації, то такий спосіб є менш затратний за часом, тому що користувач пропускає момент з відправленням коду на номер мобільного телефону та час на введення коду у відповідне вікно програми. Також варто відмітити, що при спробі обходу двофакторної автентифікації, зловмисник може отримати номер мобільного телефону користувача. У свою чергу це може загрожувати підприємству тим, що кіберзлодій може виявитись досить вмілим соціальним інженером. У подальшому за допомогою номера телефону користувача зловмисник шляхом соціальних навичок може отримати конфіденційну інформацію про підприємство. При умові, що зловмисник дізнається про унікальний ідентифікатор пристрою, яким користується лише один робітник підприємства, це йому вцілому нічого не дає. Доступ до такого пристрою є і має бути лише у визначеного користувача, ідентифікатор якого ніде не записується та не оголошується.

Раніше було зазначено, що представлений модуль забезпечення безпеки не потребує від користувача високих навичок у сфері інформаційних технологій. Оскільки була обрана мова Java Script, розроблений програмний модуль встановлюється на сервер як додатковий сценарій.

За необхідністю можливо зменшити час існування токену доступу користувача, що дасть змогу здійснити більший контроль над користувачами в системі. Зменшивши час існування токену, спеціаліст збільшить кількість звертань від сервера до клієнта, що в свою чергу призведе до збільшення навантаження на систему. Тому варто обрати такий час, при якому навантаження на сервер буде мінімальним, але за котрий зловмисник не зможе знайти важливу для підприємства інформацію.

Поетапне використання модуля:

- Встановити сценарій на сервер.
- Встановити необхідні обмеження автентифікації користувача.

Висновки за розділом 3

У цьому розділі було проведено аналіз методів автентифікації користувачів у системах управління базами даних та розглянуто переваги й недоліки різних підходів, зокрема слабкої автентифікації, двофакторної автентифікації та біометричних методів. Було встановлено, що використання JSON Web Token у процесі автентифікації забезпечує надійний рівень захисту даних, оскільки дозволяє перевіряти дійсність токенів та унеможливлює їхнє несанкціоноване використання.

Розроблений програмний модуль, заснований на технології JWT, дозволяє ефективно контролювати процес автентифікації користувачів у СУБД. Основною перевагою цього підходу є додаткова перевірка унікального ідентифікатора пристрою, що значно ускладнює доступ зловмисникам навіть у разі викрадення токену доступу.

У ході тестування модуля було продемонстровано його ефективність в умовах імітованого середовища. Зловмисники не змогли обійти систему безпеки навіть при спробі перехоплення токену доступу, що підтверджує дієвість запропонованого рішення.

Запропонований модуль є простим у використанні, не вимагає значних фінансових витрат на впровадження та може бути легко адаптований до потреб підприємства. Водночас він забезпечує високу швидкість автентифікації та мінімізує ризики несанкціонованого доступу до СУБД.

Таким чином, використання JWT у поєднанні з перевіркою ідентифікатора пристрою є ефективним методом автентифікації, який поєднує зручність для користувачів із високим рівнем безпеки.

ВИСНОВКИ

У кваліфікаційній роботі було проведено комплексне дослідження проблеми забезпечення захисту системи управління базами даних підприємства.

У першому розділі було проаналізовано основні концепції баз даних, їхньої класифікації, принципів роботи та архітектуру СУБД. Визначено ключові переваги використання баз даних у різних сферах, включаючи банківську справу, електронну комерцію, медицину, логістику та соціальні мережі. Особливу увагу було приділено реляційним, об'єктно-орієнтованим та NoSQL базам даних, а також їхнім моделям. Було доведено, що ефективне управління даними потребує сучасних підходів до їх організації, доступу та безпеки.

У другому розділі було проведено аналіз загроз безпеці СУБД та методів їхнього захисту. Було розглянуто основні вразливості СУБД, такі як атаки SQL-ін'єкціями, шкідливе програмне забезпечення, атаки на відмову в обслуговуванні (DoS/DDoS), злом облікових записів та проблеми з неналежним управлінням доступом. Запропоновано ефективні стратегії для підвищення рівня безпеки, серед яких використання багаторівневого контролю доступу, шифрування даних, двофакторна автентифікація та регулярний аудит системи. Було підкреслено важливість нормативно-правового регулювання у сфері захисту інформації, зокрема відповідність сучасним стандартам безпеки, таким як GDPR, PCI DSS та інші.

У третьому розділі було розглянуто практичну реалізацію програмного модуля автентифікації користувачів у СУБД на основі JSON Web Token (JWT). Запропонований підхід забезпечує надійність автентифікації, дозволяючи використовувати унікальні ідентифікатори пристроїв для додаткового рівня захисту. Було проведено тестування модуля, яке підтвердило його ефективність у запобіганні несанкціонованому доступу навіть у випадках викрадення токенів доступу. Запропонований метод є менш ресурсозатратним, ніж двофакторна автентифікація, але водночас забезпечує високий рівень безпеки.

Загалом, у роботі було розглянуто ключові аспекти організації, захисту та автентифікації у СУБД. Отримані результати можуть бути використані для підвищення рівня безпеки системи управління базами даних на підприємствах та в організаціях, що працюють із конфіденційною інформацією. Подальші дослідження можуть бути зосереджені на розробці додаткових механізмів захисту, зокрема на основі штучного інтелекту та машинного навчання для виявлення аномалій у доступі до БД.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Digitization of the World from Edge to Core.pdf. URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf (дата звернення: 03.03.2025).

2. Application of DBMS. GeekforGeeks. URL: https://www.geeksforgeeks.org/application-of-dbms/ (дата звернення: 03.03.2025).

3. ДСТУ ISO/IEC 2382:2017. Інформаційні технології. Словник термінів. Чинний від 2019-01-01. Вид. офіц. Київ : УкрНДНЦ, 2019.

4. ДСТУ 7448:2013. Інформація та документація. Бібліотечно-інформаційна діяльність. Терміни та визначення понять. Чинний від 2014-07-01. Вид. офіц. Київ : Мінекономрозвитку України, 2014.

5. 5 найпопулярніших систем управління базами даних 2020 року. URL: https://www.management.com.ua/partners/2021/02/21/5-najpopulyarnishih-sistem-upravlinnya-bazami-danih-2020-roku/ (дата звернення: 26.02.2025).

6. Elmasri R., Navathe S. Fundamentals of Database Systems. Addison-Wesley, 2011. 1273 p.

7. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. McGraw-Hill, 2019. 1376 p.

8. Database management system. TechTarget Network. URL: https://www.techtarget.com/searchdatamanagement/definition/database-management-system (дата звернення: 27.02.2025).

9. What is a Database Model? LucidChart. URL: https://www.lucidchart.com/pages/tutorial/database-models (дата звернення: 28.02.2025).

10. DBMS Architecture 1-level, 2-Level, 3-Level. geeksforgeeks. URL: https://www.geeksforgeeks.org/dbms-architecture-2-level-3-level/ (дата звернення: 01.03.2025)

11. What is database security? Ibm.com. URL: https://www.ibm.com/think/topics/database-security (дата звернення: 03.03.2025).

12. Anderson's rule (computer science). Wikipedia. URL: https://en.wikipedia.org/wiki/Anderson%27s_rule_(computer_science) (дата звернення: 03.03.2025).

13. Payment Card Industry Data Security Standard v4.0. Requirements and Testing Procedures. Чинний від 2008-10-01.

14. HIPAA. Wikipedia. URL: https://uk.wikipedia.org/wiki/HIPAA (дата звернення: 03.03.2025).

15. Про захист фізичних осіб у зв'язку з опрацюванням персональних даних і про вільний рух таких даних : Регламент Європейського Парламенту і Ради (ЄС) від 27.04.2016 № 2016/679 : станом на 27.02.2025 URL: https://zakon.rada.gov.ua/laws/show/984_008-16#Text (дата звернення 27.02.2025)

16. Major Database Security Threat. Tripwire Integrity Management URL: https://www.tripwire.com/state-of-security/major-database-security-threats-prevent (дата звернення: 03.03.2025).

17. Threat_horizons_report_h2_2024.pdf. Google Cloud Security URL: https://services.google.com/fh/files/misc/threat_horizons_report_h2_2024.pdf (дата звернення: 03.03.2025).

18. CYBERSECURITY REQUIREMENTS FOR FINANCIAL SERVICES COMPANIES.pdf. NEW YORK STATE DEPARTMENT OF FINANCIAL SERVICES URL: https://www.dfs.ny.gov/system/files/documents/2019/02/dfsrf500txt.pdf (дата звернення: 03.03.2025).

19. Кавун С.В., Носов В.В., Манжай О.В. Інформаційна безпека. Навчальний посібник. Харків : Вид. ХНЕУ, 2008. 352 с.

20. Дудикевич В.Б., Хорошко В.О., Яремчук Ю.є. Основи інформаційної безпеки. Навчальний посібник. Вінниця : Вид. ВНТУ, 2018. 316 с.

21. Колесніков К.В., Ободовський Б.П., Види біометричної автентифікації та методи їх оцінки. Штучний інтелект. Черкаси : Вид. ЧДТУ, 2017.

22. Request for Comments: 7519. JSON Web Token: ISSN: 2070-1721. Чинний від 2015-05-01.

23. Payment Card Industry Data Security Standard v4.0. Requirements and Testing Procedures. Чинний від 2008-10-01.

24. Introduction to JSON Web Tokens. jwt.io. URL: https://jwt.io/introduction (дата звернення: 28.02.2025).