

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи захисту баз даних
користувачів у розподіленій серверній інфраструктурі”

Виконав здобувач вищої освіти
IV курсу, групи КІ-23-МБ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Красюк І.М.
«___» _____ 2026 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«___» _____ 2026 р.

Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань ._12 _____“Інформаційні технології”
Спеціальність 123 “Комп’ютерна інженерія”
Освітньо-професійна (освітньо-наукова) програма “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 27 » лютого 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Красюку Івану Миколайовичу

(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи захисту баз даних користувачів у розподіленій серверній інфраструктурі*

2. Керівник роботи *Якименко Наталія Миколаївна, кандидат фізико-математичних наук, доцент*

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 126-02 від 27.02.2026 року

3. Строк подання студентом роботи до захисту 23.05.2026 р.

4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи захисту баз даних користувачів у розподіленій серверній інфраструктурі*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи 1 аркуш

Функціональна схема системи 1 аркуш

Діаграма процесів 1 аркуш

Блок-схема алгоритму роботи додатку 2 аркуша

7. Дата видачі завдання « 27 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника

Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача

Красюк І.М.
(прізвище та ініціали)

АНОТАЦІЯ

Красюк І.М. Програмне забезпечення системи захисту баз даних користувачів у розподіленій серверній інфраструктурі. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи захисту баз даних користувачів у розподіленій серверній інфраструктурі.

Метою розробки є програмне забезпечення системи захисту баз даних користувачів у розподіленій серверній інфраструктурі.

Результат роботи – програмна реалізація системи захисту баз даних користувачів у розподіленій серверній інфраструктурі.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі TypeScript.

Ключові слова: комп'ютерна інженерія, захист баз даних користувачів, розподілена серверна інфраструктура

ABSTRACT

Krasiuk I.M. Software for the user database protection system in a distributed server infrastructure. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the user database protection system in a distributed server infrastructure.

The purpose of the development is software for the user database protection system in a distributed server infrastructure.

The result of the work is the software implementation of the user database protection system in a distributed server infrastructure.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with Windows 10/11.

The program is developed in the TypeScript environment.

Keywords: computer engineering, user database protection, distributed server infrastructure

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП.....	4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	6
1.1 Призначення системи.....	6
1.2 Область застосування.....	7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.....	9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування.....	14
2.3 Розгорнута постановка завдання	17
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	18
3.1 Опис функціонування системи	18
3.2 Розробка структурної схеми.....	33
3.3 Розробка функціональної схеми	40
3.4 Розробка діаграми процесів.....	43
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	45
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	45
4.2 Захист розробленого програмного забезпечення.....	68
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	71
6 ОСНОВНІ ВИСНОВКИ.....	77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	79

					ВКРБ-123.26.0033.00.00.ПЗ			
Вим.	Арк.	№ докум.	Підп.	Дата	<i>Програмне забезпечення системи захисту баз даних користувачів у розподіленій серверній інфраструктурі</i>	Літ.	Аркуш	Аркушів
<i>Розроб.</i>	<i>Красюк І.М.</i>					Б	1	86
<i>Перев.</i>	<i>Якименко Н.М.</i>							
<i>Н.контр.</i>	<i>Коваленко А.С.</i>					ЦНТУ КІ-23-МБ		
<i>Затв.</i>	<i>Смірнов О.А.</i>							

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ACID	—	описує вимоги до транзакційної системи, що забезпечують найбільш надійний і передбачуваний її роботу
DB	—	Data Base
IT	—	інформаційні технології
Base64Url	—	стандарт кодування двійкових даних за допомогою тільки 64 символів ASCII
JSON	—	текстовий формат обміну даними, заснований на JavaScript
HMAC	—	один з механізмів перевірки цілісності інформації, що дозволяє гарантувати те, що передача даних або які зберігаються в ненадійною середовищі, не внесено жодних змін сторонніми особами
ПЗ	—	програмне забезпечення
SHA256	—	сімейство криптографічних алгоритмів
RSA	—	криптографічний алгоритм з відкритим ключем, який базується на обчислювальній складності задачі факторизації великих цілих чисел
OAuth	—	відкритий протокол авторизації, який дозволяє надати третій стороні обмежений доступ до захищених ресурсів користувача без необхідності передавати їй логін і пароль
POST	—	один з багатьох методів запиту, підтримуваних HTTP протоколом, використовуваним у Всесвітній павутині. Метод запиту POST призначений для запиту, при якому веб-сервер приймає дані, ув'язнені в тіло повідомлення, для зберігання
HTTP	—	HyperText Transfer Protocol

API	–	Application Programming Interface
CSRF	–	вид атак на відвідувачів веб-сайтів, який використовує недоліки протоколу HTTP
REST	–	Representational State Transfer

К6ПЗ-2026

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

ВСТУП

Актуальність теми. У сучасному середовищі, керованому даними, стартапи та команди розробників продуктів рухаються зі швидкістю блискавки. Але швидке зростання часто залишає поза увагою критично важливий компонент: надійну безпеку бази даних. Багато команд розглядають безпеку як перешкоду, складну та дорогую роботу, яка уповільнює інновації. Такий спосіб мислення не лише ризикований, це втрачена можливість. Нехтування захистом даних призводить до технічного боргу, порушень відповідності вимогам та втрати довіри клієнтів, що може паралізувати бізнес, що розвивається.

Гарна новина полягає в тому, що впровадження надійної системи безпеки не обов'язково має бути перешкодою. Застосовуючи сучасний, проактивний підхід, ви можете перетворити безпеку з захисного заходу на ключову функцію, яка будує довіру та забезпечує масштабоване зростання. У даній роботі розглянуто найважливіші передові практики безпеки баз даних, розроблені для швидкозмінних команд. Ми вийдемо за рамки загальних порад і надамо практичні, пріоритетні кроки для створення безпечної основи.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи захисту баз даних користувачів у розподіленій серверній інфраструктурі.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

- Огляд існуючих систем захисту баз даних користувачів у розподіленій серверній інфраструктурі.
- Дослідження системи захисту баз даних користувачів у розподіленій серверній інфраструктурі.
- Програмна реалізація системи захисту баз даних користувачів у розподіленій серверній інфраструктурі.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі захисту баз даних користувачів у розподіленій серверній інфраструктурі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи захисту баз даних користувачів у розподіленій серверній інфраструктурі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

К6ПЗ_2026

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Перш ніж захищатися від чогось, ми повинні зрозуміти, від чого ми захищаємося. Сьогодні, коли StackOverflow скочується у прірву, логічно припустити, що загрози, пов'язані зі штучним інтелектом, будуть на вершині списку, тому отримаємо кілька пунктів для роботи:

1. Більшість успішних компрометацій баз даних у 2026 році все ще починаються зі зловживання дійсними обліковими даними: найімовірнішою загрозою є заповнення облікових даних.

2. Компрометації баз даних та атаки на програми загалом будуть значною мірою сприятися штучним інтелектом: вважаємо що зловмисники використовуватимуть штучний інтелект для розробки експлойтів, швидшого виявлення таблиць з високим значенням, генерування реалістичних шаблонів запитів, щоб уникнути виявлення брандмауерами та іншими пристроями, а також налаштовуватимуть свої методи, щоб залишатися в межах порогів сповіщення, якщо брандмауери неможливо обійти. Іншими словами, зловмисники значно підвищать свою ефективність, використовуючи штучний інтелект для автоматизації завдань, які виконувалися б вручну.

3. Неправильні конфігурації баз даних та автентифікації залишатимуться малоімовірною проблемою ще багато років: неправильно налаштовані бази даних та пов'язані з ними конфігурації широко використовувалися до 2026 року, і ця загроза, ймовірно, продовжуватиметься в майбутньому.

4. Ризики для ланцюгів поставок та залежностей зростають: оскільки бази даних рідко працюють самостійно, зловмисники частіше звертаються до їхніх «помічників» (таких як проксі-сервери баз даних, такі як ProxySQL), незахищених агентів моніторингу та резервного копіювання або навіть сторонніх

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

SaaS- пристроїв, що полегшують доступ до бази даних.

5. Викрадання даних буде більш непомітним, і буде менше «відвертих» загроз, таких як пошкодження веб-сайтів: зловмисники у 2026 році надаватимуть перевагу невеликим, але частим запитам, що атакують певну кінцеву точку, вибіркового вилученню певних стовпців у базі даних та поєднанню операцій з викраданням даних зі звичайними робочими навантаженнями. Пошкодження веб-сайтів, ймовірно, все ще будуть актуальними, але, якщо це не операції, що спонсоруються державою, вони трапляються все рідше.

6. Тиск на дотримання вимог збільшує радіус вибуху: правила не зменшили кількість порушень, але різко посилили їхній вплив.

Можна зробити висновок, що ландшафт загроз у 2026 році включає підробку облікових даних, операції за допомогою штучного інтелекту, загрози неправильної конфігурації автентифікації та загрози, пов'язані з ризиками залежностей. Зі зростанням витоку даних, ускладненням виявлення порушень та підвищенням ставок через нормативні зобов'язання, наші бази даних та програми повідомляють про необхідність вжити наступних заходів.

1.2 Область застосування

Найгіршою загрозою для ваших баз даних і даних у них є зловживання дійсними обліковими даними, або, іншими словами, «введення облікових даних». Введення облікових даних відбувається, коли зловмисник отримує набір дійсних облікових даних внаслідок витоку даних, а потім «вставляє» ці дані у форму входу, що належить непов'язаному веб-сайту. Спрощено кажучи, все виглядає так: список дійсних облікових даних» у дев'яти випадках з десяти стосується витоку даних внаслідок порушення безпеки даних. Тепер помножте ці 6 логінів на кілька мільярдів разів (існують тисячі випадків витоку даних, і це навіть не враховуючи ті, що знаходяться у приватному володінні хакерів), і, якщо всі ваші користувачі не використовують менеджер паролів із паролями, що складаються з

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

30 або більше символів (чесно кажучи: ми всі знаємо, що це не так), ви можете бути впевнені, що кілька з них спрацюють і відкриють двері до серверної частини вашої програми. Таблиця користувачів вашої бази даних займається, і цикл повторюється.

На щастя, захист вашої бази даних і програми від перехоплення облікових даних не такий вже й складний: все, що вам потрібно зробити, це застосувати заходи запобігання методом грубої сили до всіх ваших форм входу. Це може бути так просто, як дозволити не більше X запитів від одного користувача протягом Y секунд або додати 2FA, де це можливо.

Звичайно, ви не можете зробити це лише через базу даних, оскільки вам знадобиться певний бекенд-код, що працює разом із вашою базою даних, але насправді це так просто, як додати кілька стовпців у таблицю users, а потім обробити їх за допомогою бекенд-мов програмування, щоб перевірити, чи є в них значення та які вони:

Такі заходи захисту є ефективною зброєю проти заповнення облікових даних, оскільки вони усувають основну концепцію атаки: швидкий «спам» запитів більше не є ефективним через «охолодження» ззовні (користувачі не можуть входити в систему більше одного разу протягом певної кількості секунд), і тому спроба виконати мільярди запитів одночасно більше неможлива – хіба що ви хочете чекати місяць чи два.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи захисту баз даних користувачів у розподіленій серверній інфраструктурі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Бібліотека Helmet

Helmet.js – це модуль Node.js, який допомагає в захисті HTTP-заголовків. Він реалізований в express-додатках. Тому можна сказати, що helmet.js допомагає в захисті експрес-додатків. Він встановлює різні HTTP-заголовки для запобігання таких атак, як Cross-Site-Scripting (XSS), clickjacking і т.д.

Чому важлива безпека HTTP-заголовків: Іноді розробники ігнорують HTTP-заголовки. Оскільки HTTP-заголовки можуть передавати конфіденційну інформацію про програму, тому важливо використовувати заголовки безпечним чином.

Helmet.js поставляється з великою кількістю вбудованих модулів для підвищення безпеки додатка:

- Content-Security-Policy: Встановлює політику безпеки.
- Expect-CT: Використовується для обробки прозорості сертифікатів.
- X-DNS-Prefetch-Control: Використовується для управління вибіркою DNS браузера.
- X-Frame-Options: Використовується для запобігання ClickJacking.
- X-Powered-By: Використовується для видалення заголовка X-Powered-By. Тема X-Powered-By розкриває версію сервера і його виробника.
- Public-Key-Pins: Використовується для прив'язки відкритого ключа HTTP.
- Strict-Transport-Security: Використовується для політики HTTP Strict Transport.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

- X-Download-Options: Обмежує різні опції завантаження.
- Управління кешем: Використовується для відключення кешування на стороні клієнта.
- X-Content-Type-Options: Використовується для запобігання атак Sniffing.
- Referrer-Policy: Використовується для приховування заголовка referrer.
- X-XSS-Protection: Використовується для додавання захисту від XSS-атак.

При розробці Web-додатків використовуються дві архітектури:

- монолітна архітектура;
- мікросервісна архітектура.

Блокування загроз базам даних за допомогою штучного інтелекту

Хоча боротьба з підробкою облікових даних є досить простою, протидія загрозам, що виникають за допомогою штучного інтелекту, може бути й не такою. Бачите, винуватцем тут є зловмисник, який використовує штучний інтелект для покращення своїх можливостей, а це означає, що штучний інтелект буде використано для пришвидшення чогось – замість того, щоб, наприклад, зловмисник отримав ступінь магістра ділового адміністрування в Sentry та сіяв хаос у вашій формі входу, щоб зруйнувати вашу базу даних. Оскільки немає чіткого вектора атаки, від них важко захиститися.

Водночас, загрози за допомогою штучного інтелекту – це саме такі загрози: загрози за допомогою штучного інтелекту. Це означає, що зловмисник здебільшого використовуватиме власні знання про ваш застосунок, а потім звернеться до штучного інтелекту, якщо щось піде не так. Іншими словами, все, що вам потрібно зробити, це дотримуватися правил безпеки, що застосовуються до вашого випадку використання, і все буде добре. Більшість цих правил безпеки стосуватимуться топ-10 OWASP та пов'язаних із ним загроз. Для вашої бази даних це означає наступне:

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		10

Топ-10 OWASP

Вам потрібно розширити свої знання про загрози, викладені в Топ-10 OWASP, тому що, простіше кажучи, це найгірші загрози безпеці – кошмар для спільноти безпеки.

Можливо, його не часто оновлюють (попередня версія OWASP була випущена у 2021 році, а до цього – у 2017 році), але знання десятки найкращих – це чудовий спосіб зробити вашу програму стійкою до численних загроз, які можуть завдати шкоди в майбутньому, особливо коли, по суті, список не так сильно змінюється.

Останні 3 списки включали зламаний контроль доступу та неправильну конфігурацію безпеки, а також ін'єкції та незахищений дизайн, що вказує на те, що приблизно чверть проблем, ймовірно, залишаться на кілька років.

З огляду на це, розробка стратегії щодо того, як зробити ваш застосунок і базу даних стійкими до загроз, описаних у OWASP Top 10, є надзвичайно важливою. Це також не так складно зробити: для початку, впровадження брандмауера веб-застосунків (WAF) та дотримання порад на графіку вище цілком підійде.

Удосконалення заходів, пов'язаних з моніторингом та веденням обліку

Моніторинг та ведення журналу є частиною кожної програми та бази даних. Водночас, коли ви востаннє переглядали журнали?

Я маю на увазі журнали, такі як error.log у MySQL / MariaDB, загальний журнал SQL-запитів у general.log, журнали доступу в access.log тощо. Коли ви востаннє їх перевіряли? Це не так складно зробити: наприклад, `grep "2026-01-05" access.log` відобразить усі записи, пов'язані з 5 січня 2026 року.

Перевірте їх по одному, можливо, завантажте та перевірте за допомогою Notepad++ або іншого програмного забезпечення, і знайдіть аномалії: чи є вони? Робіть це час від часу та подумайте про експорт аномалій до бази даних або на інший сервер; журнали будуть видалені після компрометації, тому їх подальша

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		11

перевірка не матиме великої користі. Ви здивуєтеся, скільки записів ви отримаєте.

Подобається вам це чи ні, але реєстрація даних у вашій базі даних може бути навіть обов'язковою для дотримання таких правил, як HIPAA, GDPR та CCPA. Немає потреби панікувати – ви можете уникнути зберігання конфіденційних даних (або видаляти їх раз на тиждень, якщо вам необхідно їх зберігати), але перегляд файлів журналів приблизно раз на тиждень не завадить.

Дотримання закону та нормативних актів

І останнє, але не менш важливе: важливо також дотримуватися чинних правил. Неможливо обговорити все це в кількох абзацах, але проведіть дослідження, запитайте у ІТ (і перевірте його відповіді: ви здивуєтеся, скільки джерел вигадано) та переконайтеся, що ви дотримуетесь вимог. Недотримання буде дорого коштувати.

Дивно, але з базами даних можна зробити чимало для полегшення цього:

1. Використовуйте перевірку вводу на основі білого списку, перш ніж вставляти дані в базу даних.
2. Хешуйте або шифруйте конфіденційні дані (зверніться до VCrypt або Blowfish для зберігання паролів. PBKDF2 також працює).
3. Зменште обсяг збору даних, де це можливо (збирайте лише ті дані, які вкрай необхідні).
4. Впроваджуйте суворі ролі та дозволи користувачів. Це допоможе з доступом до даних, гарантуючи, що конфіденційні дані будуть доступні лише уповноваженим особам. Крім того, переглядайте свої журнали частіше, ніж раз на рік.
5. Використовуйте заходи анонімізації даних. За можливості анонімізуйте або псевдонімізуйте дані, щоб захистити персональні дані від розголошення.
6. Встановіть чіткі правила зберігання даних. Встановіть чіткі правила щодо того, як довго дані будуть зберігатися та коли їх слід видаляти або анонімізувати.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

7. Майте план дій на випадок витоку даних. Наявність чіткого, задокументованого плану реагування на витік даних є надзвичайно важливою: останнє, чого ви хочете, це бути заскоченими зненацька.

Поряд із нормативними актами, існують також закони. Дехто з вас може не дуже вчитуватися в це, але для деяких випадків використання закон вимагатиме від вас значної уваги. Уявіть, що ви маєте справу з викраденими даними та використовуєте пошукову систему витоків даних за вашою базою даних; незалежно від того, яку СУБД ви оберете, ви можете задатися питанням, чи можете ви показувати конфіденційну інформацію, що стосується викрадених даних, будь-кому, хто її шукає? Можливо. Можливо, ні. Залежно від вашої юрисдикції, закони завжди зводяться до:

1. Де ви знаходитесь (які закони будуть застосовуватися до вас).
2. Від імені якої ви працюєте (керування веб-сайтом або додатком від імені компанії не те саме, що робота від імені фізичної особи).
3. Що саме ви робите.

Хоча закон – це не гра, ви можете робити одні й ті ж речі в різних країнах, і ваші дії можуть розглядатися як такі, що відповідають закону, або як такі, що його порушують. Вивчайте закон і знайте свої права – наприклад, GDPR стверджує, що кожен може вимагати видалення своїх даних.

Підсумок та наступні кроки

Захист ваших баз даних та даних у них протягом 2026 року може бути не найпростішим завданням. Водночас це далеко не виснажлива математика. Оцініть ландшафт загроз для ваших даних та бази даних і знайте, як реагувати на проблеми безпеки, пов'язані з базою даних і програмами, такі як ін'єкції, зламана автентифікація, викриття конфіденційних даних, недостатнє ведення журналу та моніторинг. Тоді все буде добре!

Загрози вашій базі даних у 2026 році включають заповнення облікових даних, атаки з використанням ін'єкцій, загрози за допомогою штучного інтелекту, а також ті, що описані в топ-10 OWASP.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		13

Використовуйте прості заходи безпеки для захисту вашої бази даних у 2026 році: призначайте користувачам у вашій базі даних лише необхідні привілеї та ролі, використовуйте перевірку вводу на стороні вашої програми, не використовуйте паролі за замовчуванням у вашій програмі чи базі даних, регулярно перевіряйте журнали, що стосуються вашої програми та бази даних, і використовуйте надійні паролі, де це можливо.

Штучний інтелект (ШІ) становить загрозу для вашої бази даних у 2026 році, хоча й не безпосередньо. Чат-боти на основі ШІ, такі як , здатні пришвидшити операції зловмисника, але лише за умови попереднього знання про свою ціль (ваш додаток/базу даних). Для початку ознайомтеся з топ-10 OWASP, поєднайте заходи захисту з ролями, привілеями та ролями у вашій базі даних, і все буде добре.

Крім того, продовжуйте розширювати свої знання про найкращі практики безпеки та відповідності на таких сайтах, як Simple Talk, дивіться вебінари, відвідуйте заходи та конференції, а також не забувайте про паперові книги.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Node.js – це платформа, побудована на базі середовища виконання JavaScript в Chrome, дозволяє легко створювати швидкі і масштабовані мережеві додатки. Node.js використовує модель неблокованого введення-виведення, керовану подіями, робить її легкою і ефективною, ідеальною для додатків реального часу, що вимагають великих обсягів даних і працюють на розподілених пристроях.

Node.js – це кроссплатформене середовище з відкритим вихідним кодом для розробки серверних і мережевих додатків. Додатки Node.js написані на JavaScript і можуть бути запущені в середовищі виконання Node.js на OS X, Microsoft Windows і Linux. Він включає функціональність ES2015, засновану на

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

можливостях V8. Функціональність групується за категоріями: завершення, заплановані і знаходяться в розробці. Завершенні можливості включаються за замовчуванням, але для включення запланованих і знаходяться в розробці можливостей використовуються прапори командного рядка. Якщо треба використовувати заплановані можливості, які майже готові, але не вважаються завершеними групою V8, необхідно запустити Node з прапором `--harmony`. Можливості, що знаходяться в розробці, менш стабільні, а для їх включення використовуються спеціальні прапори. Для отримання списку всіх можливостей Node, що знаходяться в розробці, в документації Node рекомендується провести пошук по рядку «in progress»:

```
node --v8-options | grep "in progress"
```

Список буде змінюватися в залежності від випусків Node. У Node також є графік управління версіями, що визначає доступні API.

Node.js також надає багату бібліотеку різних модулів JavaScript, що в значній мірі спрощує розробку веб-додатків з використанням Node.js.

Нижче перераховані деякі з важливих особливостей, які роблять Node.js першим вибором архітекторів програмного забезпечення:

- асинхронність і наповненості – всі API бібліотеки Node.js є асинхронними, тобто не блокуючими. Це означає, що сервер на базі Node.js ніколи не чекає, поки API поверне дані. Сервер переходить до наступного API після його виклику, а механізм повідомлень Events бібліотеки Node.js допомагає сервера отримати відповідь від попереднього виклику API;

- велика швидкість – будучи побудованою на базі двигуна V8 JavaScript Engine Google Chrome, бібліотека Node.js дуже швидко виконує код;

- однопоточний, але висока масштабованість – Node.js використовує однопоточні модель з циклом подій. Механізм подій допомагає сервера відповідати неблокуючим чином і робить сервер дуже масштабованим на відміну від традиційних серверів, які створюють обмежена кількість потоків для обробки запитів. Node.js використовує однопоточні програму, і ця ж програма може обслуговувати набагато більшу кількість запитів, ніж традиційні сервери, такі як

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

Apache HTTP Server;

- відсутність буферизації – додатки Node.js ніколи не буферизується дані.

Ці додатки просто виводять дані по частинах;

- ліцензія – Node.js доступний за ліцензією MIT.

TypeScript – це мова програмування зі статичної типізацією, що позиціонує себе як мова, що розширює можливості JavaScript. Typescript код компілюється в JavaScript код, який можна запускати як на клієнтській стороні (браузер), так і на стороні сервера (nodejs). Якість згенерованого коду можна порівняти з кодом, написаним професійним розробником з великим стажем. Мультиплатформовий компілятор TypeScript відрізняється високою швидкістю компіляції і публікується під ліцензією Apache, а в його створенні беруть участь розробники з усього світу, що призвело до традиції випускати нову версію кожні два місяці. Незважаючи на таку періодичність, версії довгий час зберігають сумісність, а по закінченню довгого часу застаріле поведінка залишається доступним при активізації спеціальних прапорів компілятора. Тому не варто побоюватися, що проект буде позбавлений нових можливостей мови через версійність відмінностей TypeScript. Ну а велика кількість розробників компілятора означає, що з кожною новою версією компілятор отримує оновлення, що стосуються всіх напрямків, будь то синтаксичні конструкції або оптимізації швидкості компіляції і збірки проекту.

TypeScript значно скорочує час на усунення помилок і виявлення багів, які часом не так просто визначити в динамічному середовищі JavaScript. Крім цього, TypeScript дозволяє писати більш зрозумілий і читаємий код, максимально описує предметну область, за рахунок чого архітектура додатки стає більш вираженою, а розробка неявно збільшує професійний рівень програміста.

Все це в своїй сукупності скорочує час розробки програми, знижуючи її вартість і надаючи розробникам можливість скоріше приступити до реалізації нового і ще більш цікавого проекту.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи захисту баз даних користувачів у розподіленій серверній інфраструктурі.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформулювати висновки про виконаний обсяг робіт та одержані результати.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У цьому розділі наведено вичерпний контрольний список, розроблений для захисту вашого найціннішого активу: ваших даних. Ви дізнаєтеся про практичні методи впровадження всього: від безпеки на рівні рядків та шаблонів доступу лише для читання до шифрування та сегментації мережі. Незалежно від того, чи створюєте ви новий SaaS-продукт, масштабуєте внутрішню аналітику чи готуетесь до аудитів відповідності, ці кроки допоможуть вам створити безпечну, стійку та надійну інфраструктуру даних. Завдяки ранній інтеграції цих практик ви можете уникнути дорогих помилок, безпечно пришвидшити розробку та перетворити захист даних на значну конкурентну перевагу.

Впровадження безпеки на рівні рядків (RLS)

Безпека на рівні рядків (RLS) – це потужний найкращий метод захисту бази даних, який фільтрує доступ до даних на основі ідентифікаційних даних або ролі користувача безпосередньо в базі даних. Замість надання користувачеві дозволу на перегляд усієї таблиці, RLS застосовує політики, які автоматично додають WHERE-речення до кожного запиту. Це гарантує, що користувачі можуть переглядати лише ті рядки, на перегляд яких вони мають чітке право, що робить його наріжним каменем сучасної архітектури багатоклієнтних застосунків.

Оскільки ці політики безпеки застосовуються на рівні бази даних, вони значно надійніші, ніж фільтрація на рівні програми. Логіку на рівні програми можна обійти або вона може містити помилки, але RLS забезпечує узгоджену та захищену від несанкціонованого доступу межу безпеки.

RLS є важливим для будь-якої платформи, яка керує даними для кількох різних груп або клієнтів. Наприклад, SaaS-компанія може використовувати RLS,

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		18

щоб гарантувати, що користувачі з однієї компанії ніколи не зможуть отримати доступ до даних іншої, навіть якщо вони використовують одну й ту саму таблицю бази даних. Саме так такі компанії, як Stripe, ізолюють дані про транзакції продавців, а інструменти для співпраці, такі як Notion, розділяють дані робочих просторів для різних клієнтів. У вбудованій аналітиці RLS є невід'ємним елементом для безпечного надання інформаційних панелей, орієнтованих на клієнтів, гарантуючи, що кожен клієнт бачить лише показники своєї організації.

Практичні поради щодо впровадження:

- Використання змінних сеансу: розробляйте політики RLS для посилення на змінні рівня сеансу, такі як `current_user_id()` або `current_tenant_id()`. Це робить політики динамічними та масштабованими без необхідності жорстко кодувати конкретні ідентифікатори користувачів.

- Поєднання із захистом на рівні стовпців: для максимального захисту поєднайте RLS із захистом на рівні стовпців. Це обмежує доступ до певних рядків, а також приховує конфіденційні стовпці (наприклад, особисту інформацію) в межах цих доступних рядків.

- Ретельне тестування політик: перед розгортанням створіть комплексний набір тестів, який імітує запити від кожної ролі користувача та потенційні граничні випадки, щоб запобігти випадковим витокам даних або відмова у доступі.

- Моніторинг впливу на продуктивність: Складні політики RLS можуть створювати накладні витрати на запити. Моніторингуйте плани виконання запитів і показники продуктивності, щоб виявити та оптимізувати будь-які вузькі місця. Ви можете дізнатися більше про сучасний підхід Querio до RLS в епоху штучного інтелекту та бізнес-аналітики.

Використовуйте доступ до бази даних лише для читання для аналітики

Одним із найефективніших та найпростіших практик безпеки баз даних є забезпечення принципу мінімальних привілеїв через доступ лише для читання.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		19

Це передбачає створення виділених користувачів бази даних для платформ аналітики та звітності, які суворо обмежені запитам даних (SELECT операторів). Позбавляючи їх можливості вставляти, оновлювати або видаляти записи, ви значно зменшуєте потенційну поверхню атаки.

Такий підхід гарантує, що навіть якщо інструмент аналітики або його облікові дані будуть скомпрометовані, зловмисник не зможе пошкодити або знищити ваші виробничі дані. Радіус вибуху обмежений, що запобігає випадковій модифікації даних через погано написані запити або навмисній шкоді від порушення безпеки.

Доступ лише для читання є основоположним для безпечного підключення інструментів бізнес-аналітики (BI), вбудованих аналітичних платформ та будь-яких програм, яким потрібно лише споживати дані. Наприклад, під час підключення такого інструменту, як Tableau, або вбудованого аналітичного рішення немає жодної операційної причини для того, щоб у нього були дозволи на запис. Надання цих дозволів без потреби наражає базу даних на ризик.

Сучасні платформи даних побудовані з урахуванням цього принципу. Наприклад, філософія проектування Querio робить акцент на конекторах баз даних лише для читання як на стандартній конфігурації безпеки, що дозволяє безпечну аналітику самостійного обслуговування без надання кінцевим користувачам доступу для запису. Аналогічно, сервіси AWS, такі як Redshift, та інструменти, такі як Tableau Server, наполегливо рекомендують створювати спеціальні облікові записи баз даних лише для читання для всіх підключень до джерел даних.

Практичні поради щодо впровадження:

– Створення виділених користувачів лише для читання: для кожного аналітичного інструменту або служби надайте унікального користувача бази даних лише з SELECTправами доступу. Уникайте повторного використання облікових даних у різних службах.

– Використовуйте представлення бази даних: Замість надання доступу до необроблених таблиць, створюйте представлення бази даних, які відображають лише необхідні стовпці та попередньо відфільтровані рядки. Це додає ще один рівень абстракції та безпеки.

– Впровадження тайм-аутів запитів: налаштуйте тайм-аути на рівні бази даних або користувача, щоб запобігти впливу тривалих або ресурсомістких аналітичних запитів на продуктивність ваших виробничих програм.

– Аудит усіх підключень: Безперервно реєструвати та перевіряти всі запити, зроблені користувачами з доступом лише для читання. Це допомагає виявляти аномальну поведінку, таку як надзвичайно великі обсяги даних, які можуть свідчити про компрометацію облікового запису.

Забезпечте шифрування під час передачі (TLS/SSL)

Шифрування під час передачі – це фундаментальна найкраща практика безпеки бази даних, яка захищає дані під час їх переміщення між користувачами, програмами та самою базою даних. Використовуючи такі протоколи, як Transport Layer Security (TLS), воно створює безпечний, зашифрований тунель для всього зв'язку. Будь-які дані, що надсилаються через незашифроване з'єднання, включаючи конфіденційні бізнес-метрики, особисту інформацію клієнтів та аналітичні запити, вразливі до перехоплення та атак типу «людина посередині».

Застосування TLS гарантує, що навіть якщо зловмисник отримає доступ до вашої мережі, він не зможе прочитати дані, що передаються. Це невід'ємна умова для захисту конфіденційності та цілісності даних, перетворюючи дані у вигляді простого тексту на нерозшифрований зашифрований текст, який непридатний для використання сторонніми особами.

У будь-якій сучасній програмі дані постійно переміщуються між сервісами: від браузера користувача до сервера програм і, зрештою, до бази даних. Без шифрування кожен із цих переходів є потенційною загрозою безпеці. Для такої платформи, як Quegio, яка обробляє конфіденційні показники продукту

та фінансові дані, застосування TLS до всіх підключень до бази даних та кінцевих точок API є надзвичайно важливим.

Великі постачальники хмарних послуг та SaaS-компанії ставляться до цього як до обов'язкового засобу контролю. Наприклад, AWS RDS спрощує вимогу шифрування під час передачі для всіх підключень до виробничої бази даних, а такі платформи, як Salesforce, вимагають TLS 1.2+ для всіх інтеграцій даних, щоб захистити своїх клієнтів. Щоб глибше зануритися в те, як працює ця критично важлива технологія, ви можете ознайомитися з цим посібником «Що таке шифрування TLS і як воно захищає ваші дані».

Практичні поради щодо впровадження:

- **Обов'язкові сучасні протоколи:** налаштуйте сервер бази даних на прийом лише з'єднань за допомогою TLS 1.2 або TLS 1.3. Явно вимкніть застарілі та вразливі протоколи, такі як усі версії SSL, TLS 1.0 та TLS 1.1.

- **Використовуйте надійні набори шифрів:** надавайте пріоритет сучасним, надійним наборам шифрів, які підтримують автентифіковане шифрування з пов'язаними даними (AEAD), таким як AES-GCM, та уникайте старіших шифрів у режимі CBC.

- **Автоматизація керування сертифікатами:** TLS спирається на дійсні сертифікати SSL/TLS. Використовуйте такі сервіси, як AWS Certificate Manager або Let's Encrypt, для автоматизації поновлення та розгортання сертифікатів, запобігаючи перебоям у роботі сервісу через випадкове закінчення терміну їх дії.

- **Регулярно тестуйте свою конфігурацію:** використовуйте зовнішні інструменти, такі як Qualys SSL Labs Server Test, для аудиту конфігурації TLS. Це допомагає виявити слабкі протоколи, шифри або інші вразливості у вашій системі.

Шифрування даних у стані спокою

Шифрування даних у стані спокою – це базова найкраща практика безпеки бази даних, яка захищає інформацію, що зберігається на диску. Воно передбачає використання надійних криптографічних алгоритмів, щоб зробити дані

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

нечитабельними для будь-кого, хто отримає несанкціонований доступ до фізичних носіїв інформації, таких як файли бази даних, резервні копії або журнали. Цей захист є критично важливим для стримування шкоди від фізичного порушення безпеки або компрометації сервера.

На відміну від шифрування під час передачі, яке захищає дані, що переміщуються мережею, шифрування в стані спокою захищає дані, коли вони стаціонарні. Для будь-якої організації, яка зберігає конфіденційну інформацію, таку як особисті дані клієнтів, фінансові записи або власні бізнес-метрики, ця практика є не просто рекомендацією, а основною вимогою для забезпечення безпеки та дотримання нормативних вимог.

Шифрування даних у стані спокою – це остання лінія захисту від витоку даних. Якщо зловмисник обійде мережеві засоби контролю та контролю доступу, щоб викрасти жорсткий диск або отримати доступ до файлу резервної копії бази даних, шифрування гарантує, що викрадені дані будуть не більше ніж незрозумілим зашифрованим текстом без відповідних ключів розшифрування. Саме тому хмарні провайдери, такі як AWS та Google Cloud, включають цю функцію за замовчуванням у своїх керованих службах баз даних. Наприклад, Amazon RDS автоматично шифрує бази даних та резервні копії за допомогою ключів, якими керує AWS KMS.

Аналогічно, аналітичні платформи, які кешують результати запитів або зберігають метадані, повинні шифрувати ці дані. Наприклад, у Querio всі кешовані дані та метадані шифруються в стані спокою, щоб гарантувати безпеку конфіденційної бізнес-аналітики навіть під час її зберігання для оптимізації продуктивності. Ця практика є невід'ємною для захисту цілісності вашого аналітичного середовища.

Практичні поради щодо впровадження:

– Стандартизація AES-256: Використовуйте розширений стандарт шифрування (AES) з 256-бітними ключами як мінімум для всіх даних, що

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

зберігаються. Це галузевий стандарт, визнаний за свою стійкість до атак методом повного перебору.

– Централізація керування ключами: впровадьте централізовану систему керування ключами (KMS), таку як AWS KMS, Google Cloud KMS або HashiCorp Vault. Зберігання ключів шифрування окремо від зашифрованих даних має вирішальне значення для запобігання єдиній точці компрометації.

– Регулярна ротація ключів шифрування: Встановіть та автоматизуйте політику ротації ключів шифрування принаймні раз на рік або частіше для висококонфіденційних даних. Це обмежує вікно можливостей для зловмисника, якщо ключ буде скомпрометовано.

– Перевірка впливу на продуктивність: Перед повним розгортанням ретельно перевірте продуктивність шифрування та дешифрування з обсягами даних виробничого рівня. Хоча сучасні рішення для прозорого шифрування даних (TDE) мають мінімальні накладні витрати, вкрай важливо перевірити їхній вплив на ваші конкретні робочі навантаження.

Впроваджуйте надійну автентифікацію та контроль доступу

Надійна автентифікація та контроль доступу – це основні найкращі практики безпеки баз даних, які гарантують, що лише перевірені користувачі матимуть доступ до даних і лише до тих даних, які їм явно дозволено переглядати. Цей багаторівневий підхід поєднує перевірку особи користувача із забезпеченням суворих дозволів, утворюючи критичний бар'єр проти несанкціонованого доступу. Він виходить за рамки простих систем імені користувача/пароля до більш надійної, орієнтованої на ідентичність моделі безпеки.

Оскільки автентифікація є першим «вартом» доступу до ваших даних, слабка реалізація підриває всі інші заходи безпеки. Такі технології, як єдиний вхід (SSO) та багатофакторна автентифікація (MFA), більше не є додатковими функціями; вони є важливими для захисту конфіденційної інформації, що

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		24

зберігається у вашій базі даних та доступ до якої здійснюється через аналітичні платформи.

Ця практика є важливою для запобігання проникненню неавторизованих користувачів у вашу систему. Як для стартапів, так і для підприємств надійна автентифікація є необхідною умовою для завоювання довіри клієнтів та отримання сертифікатів відповідності, таких як SOC 2. Централізовані постачальники ідентифікації, такі як Okta або Azure AD, спрощують керування користувачами, тоді як контроль доступу на основі ролей (RBAC) гарантує, що після автентифікації користувача його доступ суворо обмежується його посадовими обов'язками. Це запобігає як зовнішнім атакам, так і внутрішньому неправомірному використанню даних.

Наприклад, менеджер продукту, який використовує аналітичну платформу, таку як Querio, повинен мати можливість переглядати показники взаємодії з продуктом, але не конфіденційні фінансові дані, які зарезервовані для фінансової команди. Цей точний контроль забезпечується за допомогою політик RBAC, пов'язаних з їхньою ідентифікатором SSO. Ви можете дізнатися, як Querio використовує SSO та RBAC для відповідності суворим вимогам безпеки для сучасної аналітики штучного інтелекту.

Практичні поради щодо впровадження:

– Застосування багатофакторної автентифікації (MFA) скрізь: обов'язкова MFA для всіх користувачів з доступом до виробничих баз даних та адміністративних консолей. Фундаментальним аспектом цього є розуміння того, що таке двофакторна автентифікація (2FA) та її ролі в захисті облікових записів, навіть якщо паролі скомпрометовано.

– Впроваджуйте принцип найменших привілеїв (PoLP): надавайте користувачам та сервісним обліковим записам абсолютний мінімум дозволів, необхідних для виконання їхніх завдань. Уникайте загальних ролей «адміністратор».

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		25

– Використовуйте централізоване керування ідентифікацією: інтегруйте свою базу даних і програми з постачальником SSO (наприклад, Okta, Azure AD або Google Workspace), використовуючи такі стандарти, як SAML 2.0 або OIDC, для централізованого керування доступом.

– Регулярно перевіряйте доступ: плануйте щоквартальні перевірки всіх облікових записів користувачів та дозволів. Негайно видаляйте доступ для колишніх співробітників та вимикайте неактивні облікові записи, щоб зменшити площу атаки.

– Застосування коротких тайм-аутів сеансу: для програм, які отримують доступ до конфіденційних даних, встановіть тайм-аути сеансу на 15-30 хвилин бездіяльності, щоб автоматично виводити користувачів із системи та вимагати повторної автентифікації.

Моніторинг та аудит активності бази даних

Безперервний моніторинг та детальне ведення журналу аудиту є важливими передовими практиками безпеки баз даних, які забезпечують видимість усіх звернень до бази даних, запитів та змін. Створюючи незмінний запис активності, журнали аудиту встановлюють підзвітність та слугують основним механізмом виявлення підозрілої поведінки. Вони відповідають на критичні питання про те, хто отримував доступ до яких даних, коли вони отримували до них доступ та які зміни вони вносили.

Ця практика переводить безпеку з пасивного, превентивного стану до активного, детективного. У поєднанні з системами управління інформацією та подіями безпеки (SIEM) та сповіщеннями в режимі реального часу моніторинг дозволяє швидко виявляти та реагувати на потенційні інциденти безпеки, мінімізуючи вплив порушення.

Журнали аудиту не підлягають обговоренню для дотримання таких норм, як GDPR, HIPAA та SOC 2, які вимагають відстеження доступу до конфіденційних даних. Для аналітичних платформ шаблони запитів реєстрації є критично важливими для розуміння використання даних та забезпечення

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		26

правильного застосування політик. Наприклад, AWS CloudTrail надає детальні журнали аудиту для всіх викликів API, здійснених до екземплярів RDS, тоді як платформи, такі як Quegio, реєструють усі запити природною мовою та доступ до даних для комплексного аудиту. Ці журнали є безцінними для судово-медичних розслідувань після події безпеки.

Практичні поради щодо впровадження:

- Увімкнути комплексне ведення журналу: Активуйте детальне ведення журналу на рівні бази даних, фіксуючи всі спроби автентифікації (успішні та невдалі), зміни привілеїв, зміни DDL (схеми) та модифікації DML (даних).

- Централізація та захист журналів: пересилайте всі журнали бази даних до централізованої, безпечної та захищеної від несанкціонованого доступу системи зберігання, такої як Datadog, Splunk або стек ELK. Зберігання журналів поза базою даних запобігає заміні сліди зломисників.

- Впровадження сповіщень у режимі реального часу: налаштуйте сповіщення про підозрілі моделі, такі як масовий експорт даних, доступ з незвичайних географічних місць, активність після робочого часу або повторні невдалі спроби входу з однієї IP-адреси.

- Встановіть політику зберігання: Зберігайте журнали аудиту щонайменше один рік або довше, якщо цього вимагають галузеві правила. Це гарантує, що у вас буде достатньо історичних даних для судово-медичного аналізу та аудитів відповідності. Ви можете дослідити, як сучасні платформи вбудовують надійні засоби безпеки та журнали аудиту для підтримки довіри.

Впровадження правил сегментації мережі та брандмауера

Сегментація мережі – це базова найкраща практика безпеки бази даних, яка ізолює вашу базу даних від несанкціонованого мережевого трафіку. Замість того, щоб бути відкритою для публічного Інтернету або всієї внутрішньої мережі, база даних розміщується в захищеному приватному сегменті мережі. Брандмауери та суворі списки контролю доступу до мережі (ACL) виступають у

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		27

ролі гейткіперів, гарантуючи, що лише явно авторизовані сервери додатків або адміністративні інструменти можуть взаємодіяти з нею.

Такий підхід створює потужний периметральний захист. Навіть якщо зловмисник скомпрометує публічний веб-сервер, він не зможе безпосередньо отримати доступ до бази даних, оскільки мережевого шляху не існує. Ця стратегія стримування значно зменшує поверхню атаки бази даних і є критичним рівнем у моделі глибоко поглибленого захисту.

Безпосереднє відкриття бази даних в Інтернеті є однією з найпоширеніших і найнебезпечніших помилок у конфігурації безпеки. Сегментація мережі запобігає цьому за своєю природою. Наприклад, сучасний хмарний додаток розміщує свій екземпляр AWS RDS або Google Cloud SQL у приватній підмережі у віртуальній приватній хмарі (VPC). Сервери додатків, що знаходяться в окремій публічній або приватній підмережі, є єдиними ресурсами, до яких надається доступ через чітко налаштовані групи безпеки. Для таких платформ, як Qeegio, ця архітектура не підлягає обговоренню; бази даних завжди ізольовані в приватних мережах, доступ до яких мають лише основні сервери додатків платформи.

Практичні поради щодо впровадження:

– Використовуйте приватні підмережі: Завжди розміщуйте свої бази даних у приватних підмережах, до яких не можна маршрутизувати через Інтернет. Публічні підмережі призначені для ресурсів, до яких має бути прямий доступ з Інтернету, таких як балансувальники навантаження або веб-сервери.

– Використовуйте групи безпеки: використовуйте хмарні групи безпеки або правила брандмауера, щоб обмежити вхідні з'єднання певним списком відомих IP-адрес або ідентифікаторів груп безпеки, що належать до вашого рівня застосунку. За замовчуванням забороняйте весь інший трафік.

– Впровадження хоста-бастіону: для адміністративного доступу використовуйте хост-бастіон (або перехідний сервер). Це захищений сервер, до якого адміністратори спочатку підключаються (часто через VPN або SSH), перш

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		28

ніж «перейти» до бази даних, запобігаючи прямим підключенням адміністратора з відкритого Інтернету.

– Моніторинг мережевого трафіку: Реєструйте та активно контролюйте всі спроби мережевого доступу до вашої бази даних, як успішні, так і невдалі. Це допомагає виявляти аномальну активність, таку як сканування портів або спроби грубої сили з неавторизованих внутрішніх систем.

Регулярно оновлюйте та виправляйте системи баз даних

Одним із найфундаментальніших, але часто недооцінених найкращих методів безпеки баз даних є дотримання суворого, послідовного графіка оновлення. Механізми баз даних та їхні базові операційні системи – це складне програмне забезпечення з регулярним виявленням вразливостей. Відсутність встановлення патчів безпеки робить вашу систему вразливою до відомих експлойтів, які можуть легко використовувати автоматизовані засоби атаки.

Надійна стратегія встановлення патчів передбачає своєчасне застосування оновлень безпеки без шкоди для стабільності системи. Це означає інтеграцію патчів у стандартний життєвий цикл розробки та експлуатації, разом із тестуванням та перевіркою, а не ставлення до них як до надзвичайної ситуації.

Невиправлені вразливості – це найлегший шлях для зловмисників. Одна відома загальна вразливість та ризик (CVE) у вашій версії бази даних може надати зловмиснику повний адміністративний доступ, що призведе до катастрофічних витоків даних. Для платформ, які обслуговують кілька організацій, таких як аналітичні інструменти, відповідальність ще більша. Одне порушення може поставити під загрозу дані кожного клієнта на платформі.

Керовані сервіси, такі як AWS RDS, спрощують це, пропонуючи автоматичні вікна виправлень, але ви все ще несете відповідальність за їх правильне налаштування. Для самостійно розміщених баз даних інструменти, такі як Ubuntu, unattended-upgrades можуть автоматизувати виправлення безпеки на рівні ОС, але виправлення, специфічні для бази даних, вимагають більш практичного, планового підходу.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		29

Практичні поради щодо впровадження:

– Підпишіться на сповіщення постачальників: активно відстежуйте канали оголошень про безпеку для вашої конкретної бази даних, такі як список розсилки безпеки PostgreSQL або критичні оновлення Oracle, щоб бути в курсі нових вразливостей.

– Встановлення частоти виправлень: Визначте регулярний графік (наприклад, щомісяця або щокварталу) для перевірки та застосування виправлень. Це створює передбачуваний процес і запобігає накопиченню боргу за безпеку.

– Тестові патчі в проміжному середовищі: Ніколи не застосовуйте патчі безпосередньо до робочого середовища. Завжди розгортайте їх у проміжному середовищі, яке відображає робоче середовище, щоб перевірити наявність регресій або проблем з продуктивністю критичних запитів програми.

– Автоматизація перевірки після оновлення: Створіть автоматизований набір тестів, який запускається після застосування оновлення на етапі тестування. Цей набір має перевіряти основну функціональність програми, щоб переконатися, що оновлення не внесло критичних змін.

– Створення резервних копій перед оновленням: Перш ніж застосовувати будь-яке суттєве оновлення до вашого робочого середовища, переконайтеся, що у вас є нещодавня, перевірена резервна копія. Це забезпечує критично важливий шлях відкату, якщо оновлення спричинить непередбачені проблеми.

Використовуйте моніторинг активності бази даних (DAM) та виявлення загроз

Моніторинг активності бази даних (DAM) – це передова практика безпеки, яка забезпечує безперервний аналіз поведінки бази даних у режимі реального часу для виявлення аномальної активності та сповіщення про неї. На відміну від стандартних журналів аудиту, які часто переглядаються після інциденту, інструменти DAM використовують поведінкову аналітику та машинне навчання

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		30

для виявлення потенційних загроз у міру їх виникнення, таких як витік даних, спроби несанкціонованого доступу та внутрішні загрози.

Ці рішення працюють, встановлюючи базовий рівень нормальної поведінки користувачів та запитів. Коли активність суттєво відхиляється від цього базового рівня, наприклад, користувач раптово експортує великі обсяги даних або виконує незвичайні запити, що змінюють схему, система запускає сповіщення або навіть може блокувати шкідливий запит у режимі реального часу. Така проактивна позиція є критично важливим компонентом сучасної системи найкращих практик безпеки баз даних.

DAM є важливим для організацій, які керують конфіденційною інформацією, такою як фінансові дані та бізнес-метрики, що обробляються такими платформами, як Qeegio. Він забезпечує інтелектуальний рівень захисту, який може виявляти складні загрози, що можуть обійти традиційні засоби контролю. Наприклад, DAM може розрізнити легітимний запланований звіт та зловмисну спробу вилучити всю таблицю клієнтів. Це невід'ємний інструмент для виконання вимог відповідності, таких як GDPR, HIPAA та PCI DSS, які вимагають суворого моніторингу доступу до конфіденційних даних.

Практичні поради щодо впровадження:

- Встановлення чіткої базової лінії: перед повним розгортанням рішення DAM дайте йому запуснитися в режимі навчання, щоб точно змодельовати звичайні шаблони запитів та поведінку користувачів. Це мінімізує кількість хибнопозитивних результатів.

- Створення політик високого ризику: налаштуйте спеціальні політики сповіщень для типів запитів високого ризику, таких як ті, що спрямовані на стовпці ідентифікаційної інформації, експорт великих обсягів даних або внесення змін до схеми (ALTER TABLE, DROP TABLE).

- Інтеграція з реагуванням на інциденти: підключіть сповіщення DAM безпосередньо до системи управління інформацією та подіями безпеки (SIEM)

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		31

або до робочого процесу реагування на інциденти. Це гарантує, що критичні сповіщення негайно передаються команді безпеки.

– Регулярно налаштовуйте політики: переглядайте та налаштовуйте політики сповіщень принаймні щомісяця. У міру розвитку вашої програми розвиватиметься й її звичайна активність у базі даних. Регулярне налаштування має вирішальне значення для зменшення втрати від сповіщень та підтримки високої точності виявлення.

Регулярно проводите аудити безпеки та тестування на проникнення

Проактивні оцінки безпеки є невід'ємною частиною виявлення та усунення вразливостей, перш ніж ними зможуть скористатися зловмисники. Ця практика включає поєднання регулярних аудитів безпеки, сканування вразливостей та тестів на проникнення для ретельної оцінки ефективності засобів контролю безпеки вашої бази даних. Ці дії змінюють вашу позицію безпеки з реактивної на проактивну, забезпечуючи постійне вдосконалення.

Моделюючи реальні сценарії атак та систематично перевіряючи наявність неправильних конфігурацій, ці оцінки надають чітке уявлення про слабкі місця вашої безпеки. Результати цих тестів формують критичний цикл зворотного зв'язку, що дозволяє вам визначити пріоритети виправлень, посилити захист та перевірити, чи ваші заходи безпеки працюють належним чином.

Регулярні аудити та тести на проникнення є важливими для побудови довіри та дотримання вимог до відповідності. Для платформ, що обробляють конфіденційні дані, незалежний аудит, такий як звіт SOC 2 Type II, надає клієнтам перевірену гарантію ваших практик безпеки. Наприклад, програма винагороди за виявлення помилок на такій платформі, як HackerOne, може залучати етичних хакерів для проведення тестування безпеки, тоді як щорічний тест на проникнення від такої фірми, як Mandiant, може імітувати складну атаку на вашу інфраструктуру бази даних. Ці практики допомагають виявити слабкі місця, які автоматизовані інструменти можуть пропустити.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		32

Практичні поради щодо впровадження:

- Плануйте періодичні сканування: Проводьте комплексне сканування на вразливості принаймні раз на квартал, щоб виявити відомі проблеми безпеки у вашому програмному забезпеченні бази даних та базовій інфраструктурі.
- Залучення зовнішніх експертів: Проводьте щорічне тестування на проникнення, яке проводить авторитетна стороння фірма. Їхній зовнішній погляд є безцінним для виявлення упереджень та прогалин у ваших оцінках внутрішньої безпеки.
- Створіть дорожню карту для усунення недоліків: Розглядайте висновки не як невдачі, а як можливості. Розробіть чітку дорожню карту для усунення виявлених вразливостей, визначивши відповідальних та встановивши реалістичні терміни для усунення недоліків.
- Реагування на інциденти в тестах: Включіть тренування з реагування на інциденти у ваші тести на проникнення. Це дозволить вам відпрацювати реакцію вашої команди на змодельоване порушення, удосконалюючи процедури комунікації та стримування. Ви можете дослідити, як рівень безпеки Querio посилюється завдяки практикам, що відповідають вимогам, та регулярним аудиторам.

3.2 Розробка структурної схеми

OAuth Google

Щоб продовжувати використовувати певний додаток зі своїм обліковим записом Google, користувачі у вашій організації повинні перейти на безпечніший тип доступу, який називається OAuth. OAuth дозволяє додаткам отримувати доступ до облікових записів за допомогою цифрового ключа, замість того, щоб вимагати від користувача введення імені користувача та пароля.

Структурна схема містить у собі наступні основні блоки:

- Блок роботи веб-додатку.
- Блок SIEM (Security information and event management).

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		33

Блок роботи веб-додатку

Отже, необхідно організувати основні дії веб-додатку. І майже в усіх веб-додатків використовуються дії по пошуку, видалення або редагування інформації про юзера. Формується наступний список:

- локальна реєстрація користувача, збереження його у базі даних;
- локальна авторизація користувача та надання йому токена для взаємодії з функціями веб-додатку;
- авторизація користувача за допомогою OAuth Google та надання йому токена для взаємодії з функціями веб-додатку;
- пошук юзерів веб-додатку;
- редагування юзерів веб-додатку;
- отримання повного списку юзерів веб-додатку.

Блок SIEM

Технологія SIEM забезпечує аналіз у реальному часі подій (тривог) безпеки, що виходять від мережних пристроїв і додатків.

Структурно SIEM містить у собі:

- створення унікального токена для доступу різних функцій веб-додатку;
- включення нових HTTP заголовків для покращення захисту трансферу інформації між сервером та клієнтом на базі NodeJS бібліотеки Helmet;
- підтвердження електронної пошти, за методом надсилання туди унікального токена;
- організація Guard для виключення можливості підміни токена та розподілення по типу від JWT токена до OAuth;
- організація Guard для виключання можливості несанкціонованого доступу до ресурсів веб-додатку.

Отже, бази даних користувачів для веб-продуктів на основі NodeJS повинен мати усі пункти вимог SIEM за для забезпечення кібербезпеки від усіх основних атак, які можуть бути здійснені на веб-додаток.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		34

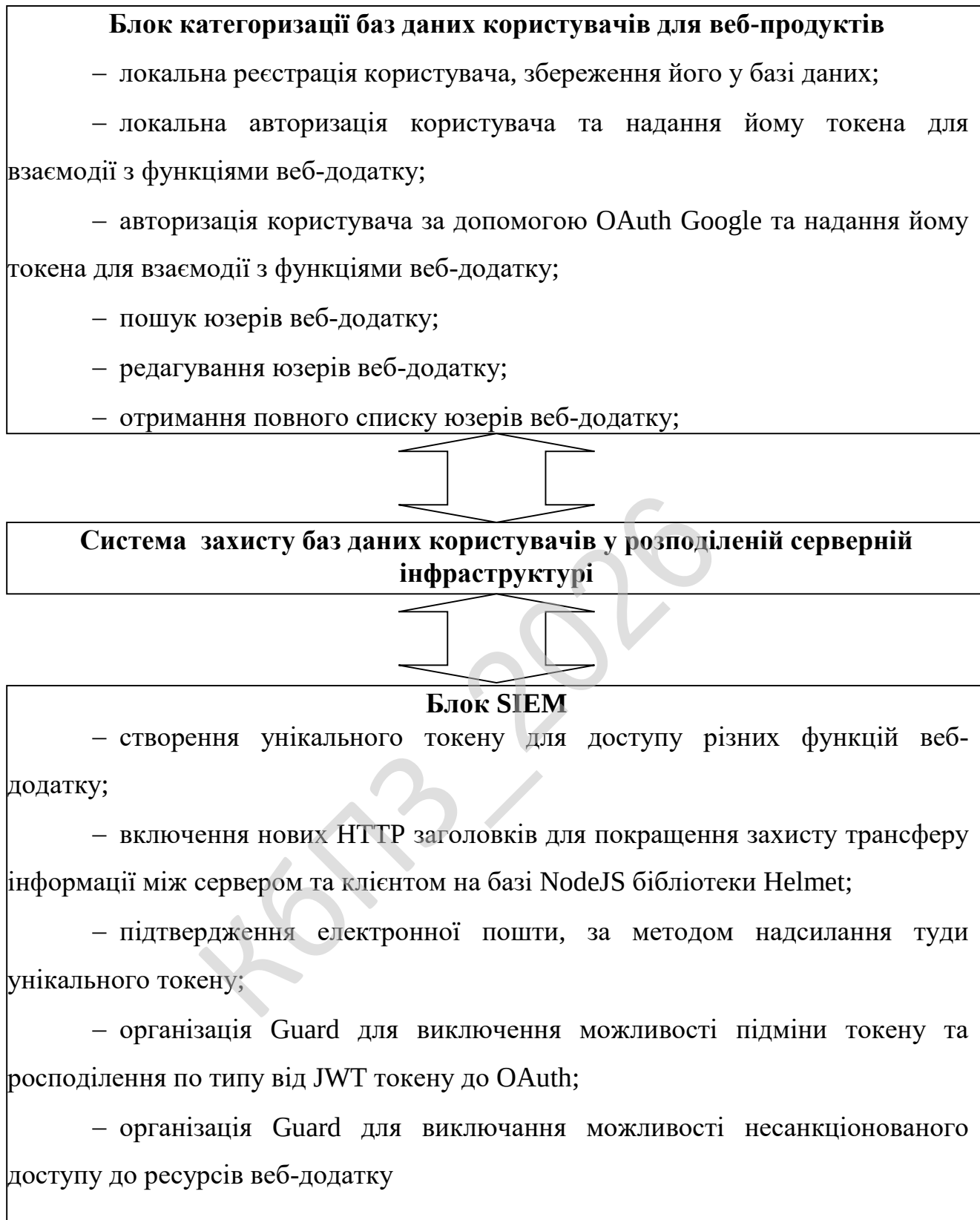


Рисунок 3.1 – Структурна схема системи

Пошук ідентифікатора/секрету клієнта OAuth у величезній консолі може здатися проблемою пошуку голки в копиці сіна! Хоча пошук – це потужний інструмент, насправді це один із трьох основних способів знайти дані OAuth.

Станом на 2026 рік, ось найефективніші методи пошуку вашого ідентифікатора клієнта та секретного коду:

Глобальний рядок пошуку (найшвидший)

Рядок пошуку у верхній частині консолі Google Cloud оптимізований для «загублених» користувачів.

Ключові слова для використання: введіть облікові дані, OAuth або клієнти.

Результат: Зазвичай першим результатом у розділі «Продукти та сторінки» буде API та послуги > Облікові дані або новіша платформа Google Auth > Клієнти. Натискання на це переведе вас безпосередньо до списку всіх ваших ідентифікаторів.

Стратегія "Завантаження JSON" (найнадійніша)

Іноді інтерфейс користувача приховує секрет за значком «Копіювати» або всередині підменю «Редагувати», щоб запобігти «серфінгу через плече» (люди дивляться на ваш екран).

Якщо ви бачите свій ідентифікатор клієнта у списку, але не можете знайти секретний ключ, знайдіть значок «Завантажити JSON» (зазвичай це невеликий трей зі стрілкою) у крайньому правому куті рядка.

Відкрийте цей файл у будь-якому текстовому редакторі. У ньому буде чітко перераховано:

```
"client_id": "ваш-id-тут"
```

```
"client_secret": "ваш-секрет-тут"
```

Новий навігаційний шлях (керівництво)

Якщо ви надаєте перевагу навігації по бічній панелі, Google пропонує інструменти консолідованої ідентифікації. Скористайтеся цими двома способами:

Основний шлях: Меню навігації (☰) > API та сервіси > Облікові дані.

Новіший об'єднаний шлях: Меню навігації (≡) > Платформа Google Auth > Клієнти. (Google все частіше переміщує сюди завдання, пов'язані з OAuth, щоб відокремити їх від ключів API серверної частини).

Важливі поради щодо усунення несправностей

Клієнт «без секретів»: Якщо ви шукаєте секрет для ідентифікатора клієнта Android, iOS або Chrome, ви його не знайдете. Google не видає секрети для цих платформ, оскільки вони вважаються «публічними клієнтами», де секрет не можна безпечно приховати в коді.

Значок «Око»: На сторінці з даними облікових даних знайдіть маленький значок ока поруч із замаскованим полем (••••••••). Натискання на нього розкриє секрет без необхідності завантажувати файл.

Пояснення вразливостей Google OAuth та способи захисту від них

Нещодавно виявлена вразливість Google OAuth дозволяє колишнім співробітникам зберігати доступ до корпоративних ресурсів, таких як Slack та Zoom, навіть після призупинення дії їхніх корпоративних облікових записів Google. Ось що це означає для безпеки вашого SaaS.

OAuth має спростити безпеку, а не обійти її. Але під поверхнею кожної кнопки «Увійти за допомогою Google» ховається маловідома вразливість, яка дає зловмисникам та колишнім співробітникам безкоштовний доступ до ваших корпоративних SaaS-додатків.

Не потрібно жодного фішингу чи шкідливого програмного забезпечення. Просто розумне використання власної системи псевдонімів Google, щоб залишатися в системі ще довго після того, як користувачі покинули будівлю.

Це не гіпотетично. Це відбувається сьогодні, і більшість ІТ-команд навіть не підозрюють, що це відбувається. Якщо ви коли-небудь звільняли співробітника, скасовували його доступ і вважали, що двері зачинені, можливо, вам варто перевірити ще раз. Тому що тіньові облікові записи Google можуть тримати ці двері відчиненими.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		37

Що таке тіньові облікові записи Google і які ризики вони несуть?

Це відомий лайфхак для робочої електронної пошти: додайте «+string» до своєї робочої адреси електронної пошти (наприклад, alice+testing@example.com), і ви зможете легко фільтрувати будь-які небажані маркетингові електронні листи або створювати кілька різних реєстрацій для одного додатка, використовуючи свою робочу адресу електронної пошти.

Однак, цей самий трюк можна використовувати для створення абсолютно нового облікового запису Google – такого, що виглядає як корпоративна електронна адреса та пересилає на неї повідомлення, але насправді не керується і навіть не відображається у вашій корпоративній організації Google.

Це створює велику проблему, коли йдеться про переведення співробітників на іншу роботу. Якщо певний «тіньовий обліковий запис Google» використовувався для реєстрації корпоративних облікових записів SaaS, таких як Slack або Zoom, за допомогою Google OAuth (тобто «Увійти за допомогою Google»), цей доступ міг зберігатися навіть після призупинення дії корпоративного облікового запису Google співробітника. Фактично, адміністратор Google не міг переглянути або призупинити тіньовий обліковий запис Google зі своєї консолі адміністратора. Це могло б залишити «чорний хід» для несанкціонованого доступу колишнього співробітника або зловмисника шляхом компрометації тіньового облікового запису.

Це TL;DR щодо вразливості Google OAuth, публічно розкритої компанією Truffle Security Co в суботу, 16 грудня 2023 року.

Важливо розуміти, як ця вразливість може вплинути на безпеку вашого SaaS-провайдера, які з ваших SaaS-провайдерів могли постраждати, включаючи Nudge Security, і як ви можете використовувати Nudge Security для виявлення та реагування на дії, пов'язані з цією вразливістю.

Які SaaS-провайдери постраждали?

Ця вразливість пов'язана з тим, як постачальники SaaS реалізують Google OAuth, що не зовсім очевидно чи інтуїтивно зрозуміло. Захована в документації

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		38

Google OIDC, вона рекомендує не використовувати адресу електронної пошти користувача як основну ідентифікацію для запису користувача у вашому SaaS-додатку, а натомість використовувати твердження HD.

Однак багато постачальників SaaS роблять це, включаючи Slack та Zoom, як зазначається в блозі Truffle Security. Проблема для організацій полягає в тому, що немає очевидного способу дізнатися, які з їхніх постачальників SaaS можуть постраждати від цієї вразливості, окрім як звернутися безпосередньо до кожного постачальника SaaS.

Який практичний вплив цієї вразливості?

Хоча на сьогоднішній день не повідомлялося про жодні випадки використання цієї вразливості, потенціал для її використання поширюється на несанкціонований доступ, витік даних і навіть атаки повного захоплення облікового запису. Розглянемо сценарій, у якому співробітник створює «тіньовий обліковий запис Google» для реєстрації в бізнес-програмі SaaS з правами адміністратора. Цей тіньовий обліковий запис Google, який не захищений і не контролюється програмою безпеки організації, пізніше скомпрометовано, що дає зловмиснику повний доступ до програми SaaS та її даних.

Ця вразливість підкреслює значний зсув, що відбувся в сучасній кібербезпеці. З огляду на те, що більшість організацій використовують сотні SaaS-додатків, багато з яких адмініструються бізнес-користувачами поза межами IT-відділу, для команд IT-безпеки стає дедалі складніше підтримувати видимість та контроль над тим, хто (і що) має доступ до корпоративних даних і як. Організаціям потрібна видимість SaaS-ідентифікаторів та дій поза межами (і на додаток до) їхнього централізованого постачальника хмарних ідентифікаційних даних або рішення SSO.

Отже, як почати вирішувати цю вразливість?

Згідно з публікацією Truffle Security, «все, що вам потрібно зробити, це вимкнути вхід через Google і суворо дотримуватися SAML. Це насправді працює для більшості постачальників послуг, проте деякі не пропонують цю опцію, а

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		39

також у вас можуть бути деякі внутрішньо розроблені програми, на які це впливає і які не підтримують SAML».

На жаль, цей шлях не є практичним для більшості організацій. Як зазначалося, він не повністю охоплює всі SaaS-додатки, які можуть використовуватися в організації, оскільки підтримка SAML залежить від постачальника SaaS.

Ось як можна виявити створення тіньових облікових записів Google з консолі адміністратора Google. Примітка: Для доступу до Центру безпеки вам може знадобитися корпоративна ліцензія.

1. Перейдіть до розділу Безпека > Центр безпеки > Інструмент розслідування

2. Створіть розслідування з наведеними нижче умовами:

3. Від (адреса заголовка) містить noreply@google.com

4. Тема містить підтвердити вашу адресу електронної пошти

5. До (конверт) містить +

Виявляємо ці події електронної пошти в рамках нашого потужного аналізу електронної пошти, тож ви можете виявити ці дії як у минулому, так і в міру їх виникнення.

Щоб переглянути тіньові облікові записи Google, перейдіть на панель інструментів «Програми». Знайдіть свою програму Google Workspace. Перейдіть до розділу «Ресурси» та відфільтруйте за псевдонімом облікового запису. Тут ви побачите список усіх тіньових облікових записів Google, створених за допомогою вашого домену.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Функціональна схема розроблена для прикладу реалізації комплексних заходів по організації системи кібербезпеки веб-додатку.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

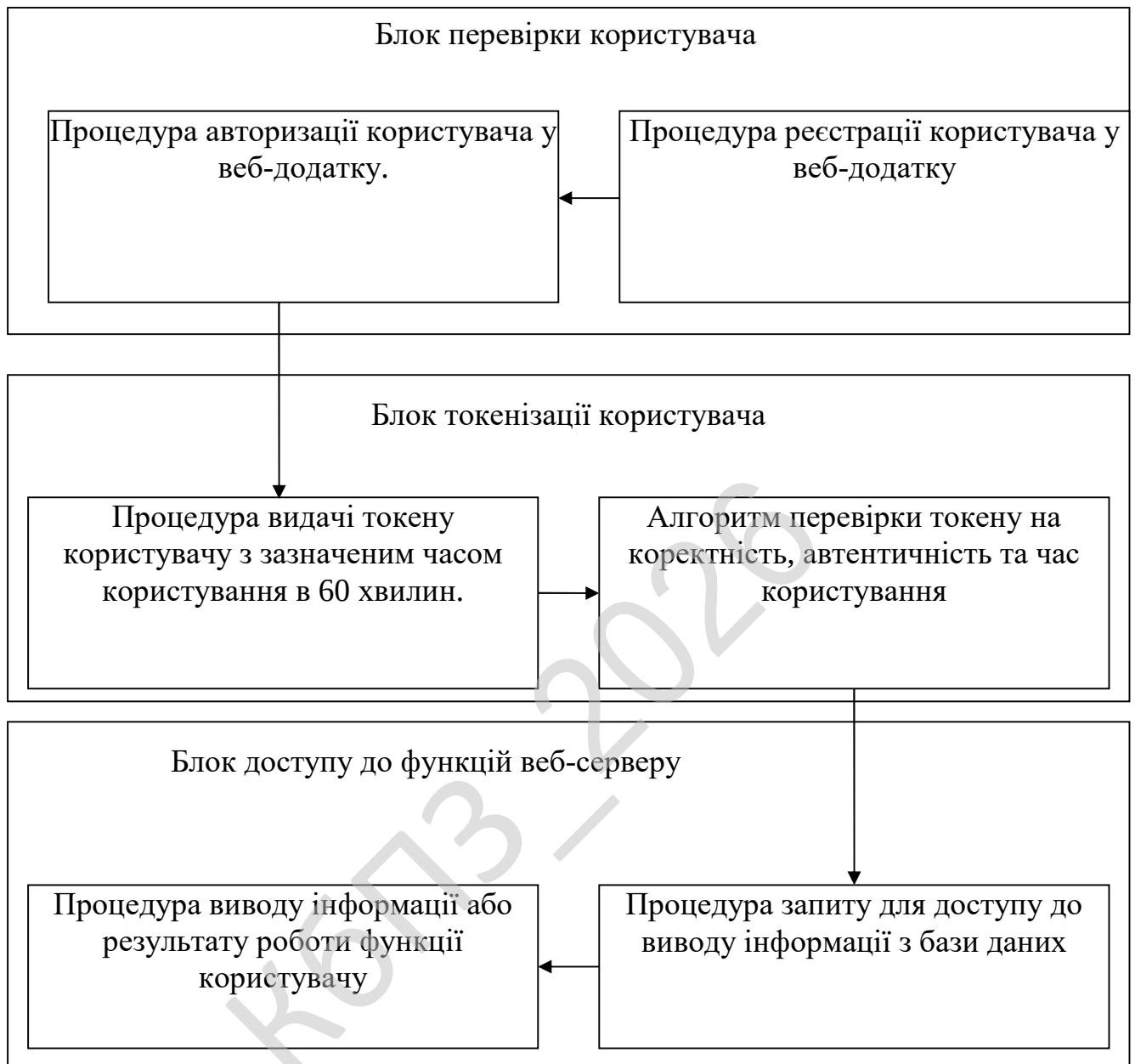


Рисунок 3.2 – Функціональна схема системи

Функціональні схеми застосовуватися у програмуванні для візуалізації алгоритмів і спрощення обчислення їх складності, у цій сфері форма створення – довільна. За такою схемою можна визначити, як здійснюються перетворення і які для цього необхідні функціональні елементи. Кожен функціональний елемент містить лише ті входи і виходи, які необхідні для його коректної роботи. Дана

схема розробляється на основі структурної схеми для кожного блоку, в результаті з окремих функціональних елементів складається загальна функціональна схема об'єкту. На основі функціональної та структурної схем розробляється принципова схема.

Функціональними схемами користуються для вивчення принципу роботи виробів (у нашому випадку програмного забезпечення), а також при їх налагодженні, контролі чи ремонті. На такій схемі зображують всі функціональні частини виробу та основні зв'язки між ними. Функціональні частини на схемі зображують у вигляді умовних графічних позначень згідно з вимогами державних стандартів. Дозволяється окремі функціональні частини, на яких немає умовних графічних позначень, зображувати у вигляді прямокутників, а також розкривати до рівня принципових схем.

Дозволяється об'єднувати функціональні частини в функціональні групи, які виділяють на схемі штрих-пунктирними лініями. Кожній виділеній групі присвоюють назву або умовне позначення.

На схемі повинно бути вказано:

- для кожної функціональної частини, зображеної прямокутником, її назва або умовне позначення, вписане у прямокутник;
- для кожної функціональної частини або елемента, зображеного умовним графічним позначенням позиційне позначення.

Якщо функціональна схема використовується разом з принциповою, то позиційне позначення елементів та функціональних частин на цих документах повинні бути однаковими. Перелік елементів в цьому випадку для функціональної схеми не складають, оскільки користуються даними принципової схеми. Якщо функціональна схема розробляється самостійно (без принципової), то позиційне позначення елементів і функціональних частин вказують за загальними правилами і розробляють перелік елементів. На функціональних схемах рекомендується вказувати поряд з графічним позначенням чи на вільному полі схеми технічні характеристики функціональних частин, діаграми, параметри сигналів тощо.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		42

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходить взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

- Зовнішні по відношенню до системи сутності.
- Потоки даних між елементами трьох попередніх типів.
- Процеси які являють собою трансформацію даних в рамках описуваної системи.
- Сховища даних (репозиторії).



Рисунок 3.3 – Діаграма взаємодії процесів авторизації користувача

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію бакалаврської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

Було створено блок-схеми роботи системи. Перед їх розглядом необхідно провести роз'яснення який саме тип блок-схем використовується. Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		45

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач. Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції. Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елемента.

Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елемента. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується.

Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи. Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані. Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи). З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми.

Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

На рисунку 4.1 зображена основна блок-схема програми для проведення загальної авторизації та реєстрації користувача веб-додатку.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML з DHTMLX, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

Було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

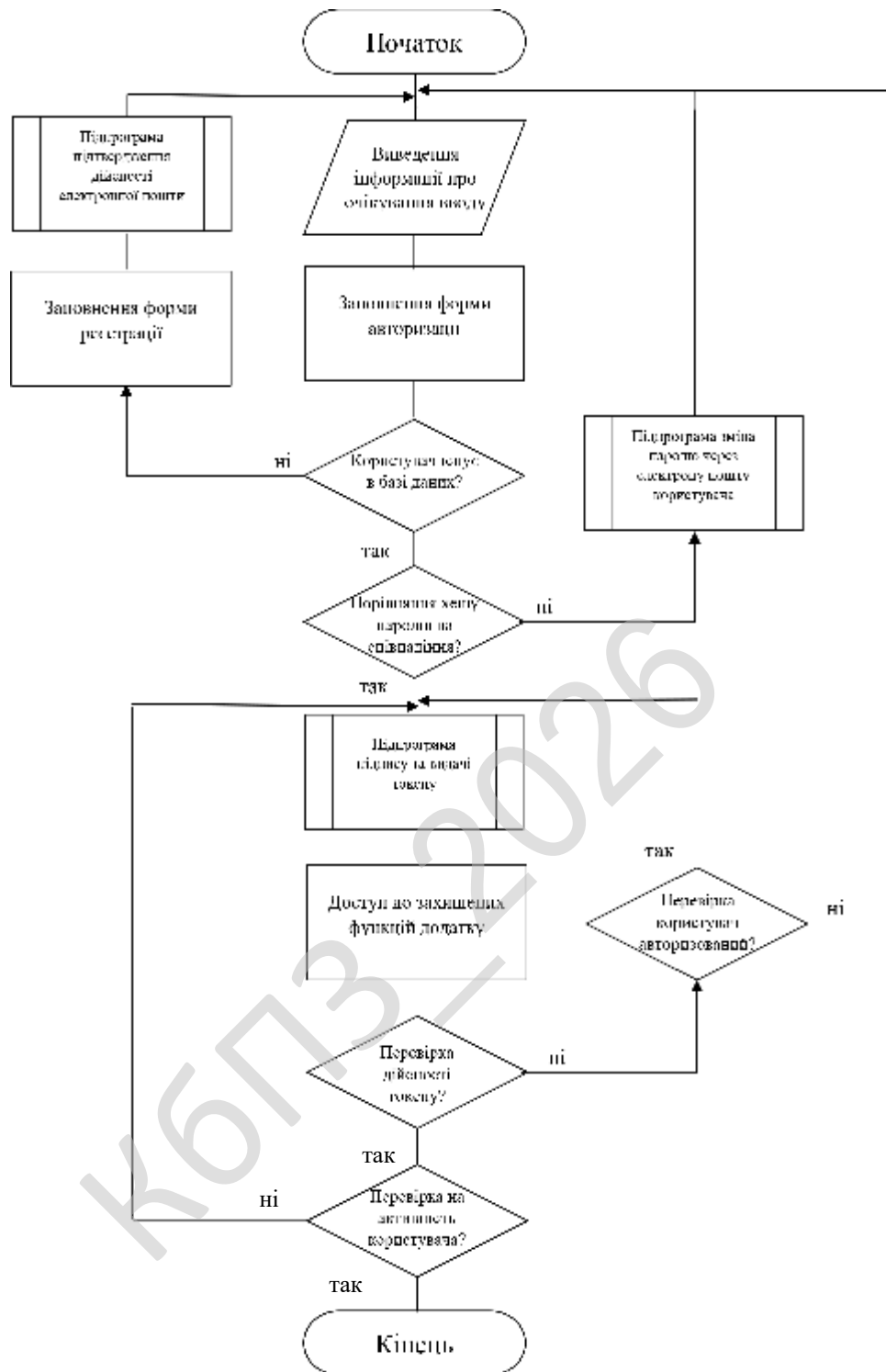


Рисунок 4.1 – Блок-схема основної програми

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною

одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

- асоціації (association relationship);
- включення (include relationship);
- розширення (extend relationship);
- узагальнення (generalization relationship).

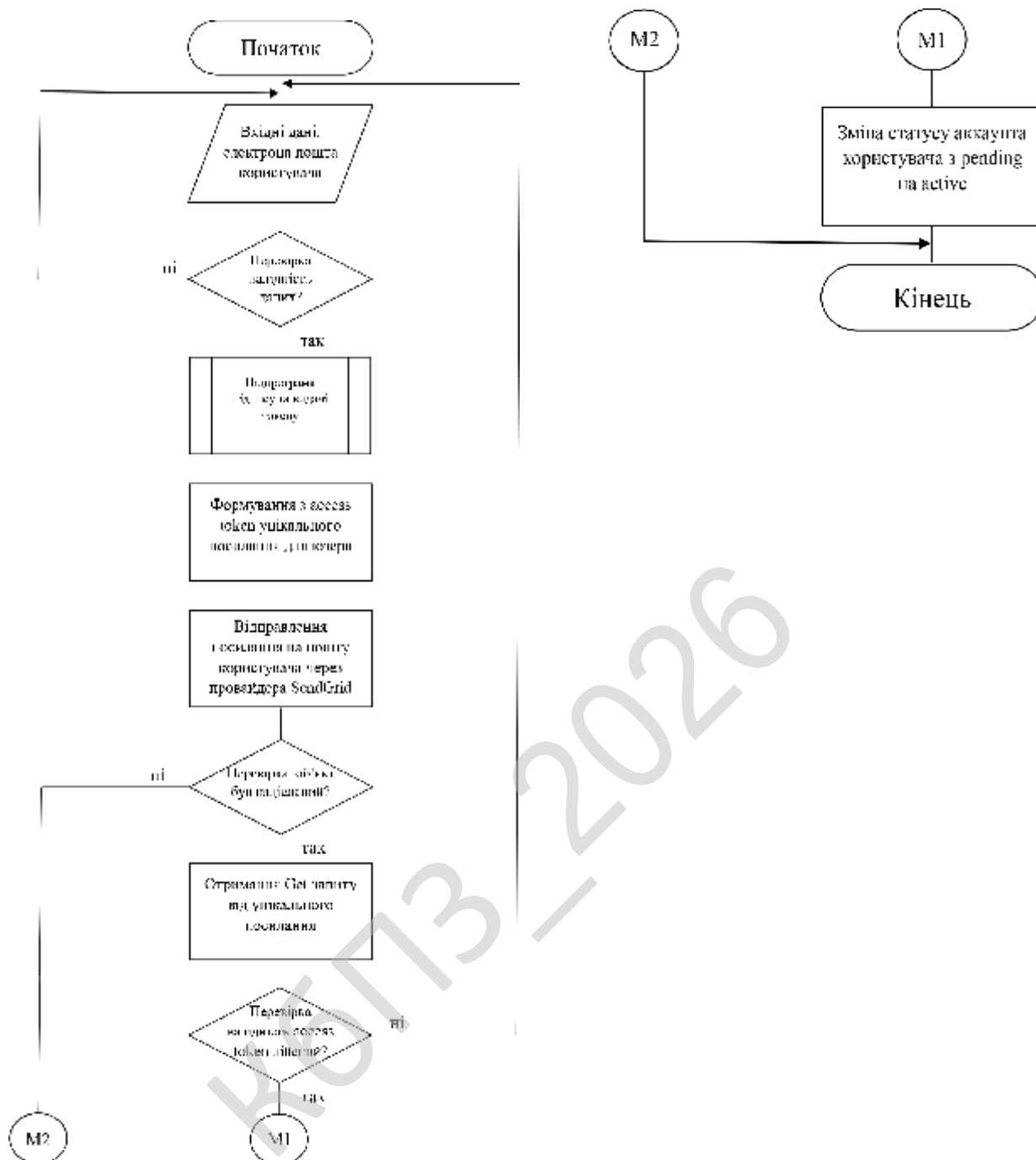


Рисунок 4.2 – Блок-схема роботи підпрограми для підтвердження електронної пошти

На рисунку 4.2 зображена блок-схема роботи підпрограми для підтвердження електронної пошти та зміна статусу аккаунту, для вільного користування веб-додатком.

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

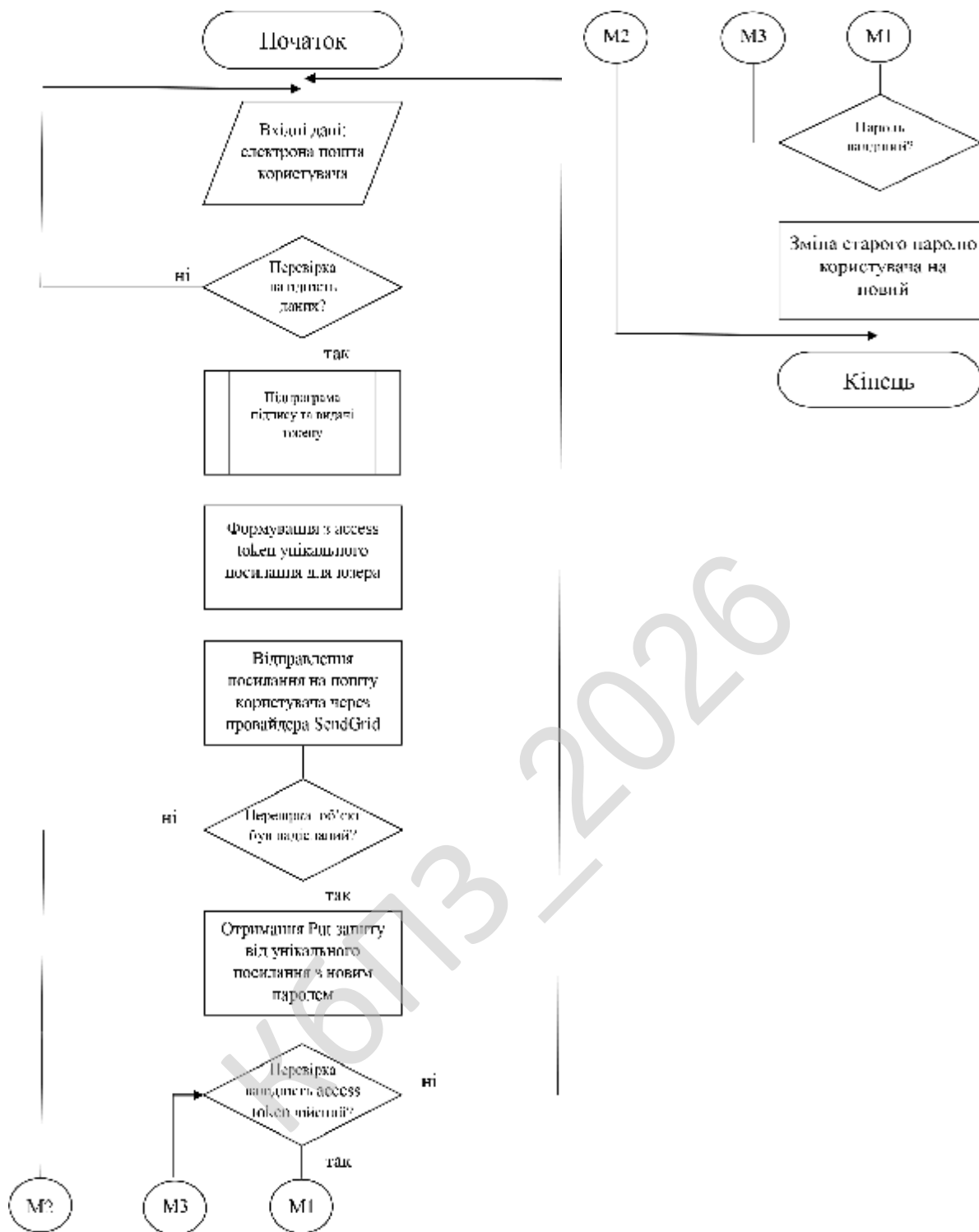


Рисунок 4.3 – Блок-схема роботи підпрограми для зміни паролю

На рисунку 4.3 зображена блок-схема роботи підпрограми “забув пароль” для зміни паролю користувача веб-додатку

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		53

Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший.

Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого».

Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовани ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортання.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою

відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

- модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;
- модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		57

найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузером.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення – звичайний браузер. Це позбавляє його необхідності завантажувати

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		58

та інстальовати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

При розробці ПЗ було використано підходи ризик-менеджменту – це система управління ризиками, яка включає в себе стратегію та тактику управління, направлені на досягнення основних цілей. Ефективний ризик-менеджмент включає:

- систему управління;
- систему ідентифікації і вимірювання;
- систему супроводження (моніторингу та контролю).

Сучасна наука представляє ризик як вірогідну подію, в результаті настання якої можуть відбутися позитивні, нейтральні або негативні наслідки. Якщо ризик припускає наявність як позитивних, так і негативних результатів, він відноситься до спекулятивних ризиків. Якщо ж наслідки негативні, або відсутні взагалі, такий ризик іменується чистим.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		59

Мета ризик-менеджменту – підвищення конкурентоспроможності господарюючих суб'єктів за допомогою захисту від реалізації чистих ризиків.

Теорія ризик-менеджменту ґрунтується на трьох базових поняттях: корисності, регресії і диверсифікації.

Етапи ризик-менеджменту

У ризик-менеджменті прийнято виділяти декілька ключових етапів:

– на першому етапі відбувається виявлення ризику з супутньою оцінкою вірогідності його реалізації і масштабу наслідків;

– на другому етапі здійснюється розробка ризик-стратегії з метою зниження вірогідності реалізації ризику і мінімізації можливих негативних наслідків;

– на третьому етапі вибираються методи і інструменти управління виявленим ризиком;

– на четвертому етапі проводиться безпосереднє управління ризиком;

– на завершальному етапі оцінюються досягнуті результати і коректується ризик-стратегія.

За ключовий етап ризик-менеджменту вважається етап вибору методів і інструментів управління ризиком.

Методи і інструментарій ризик-менеджменту

Базовими методами ризик-менеджменту є відмова від ризиків, зниження, передача і ухвалення.

Ризик-інструментарій значно ширший. Він включає політичні, організаційні, правові, економічні, соціальні інструменти, причому ризик-менеджмент як система допускає можливість одночасного застосування декількох методів і інструментів ризик-управління.

Найбільш часто вживаним інструментом ризик-менеджменту є страхування. Страхування припускає передачу відповідальності за відшкодування передбачуваного збитку сторонній організації (страхової компанії).

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		60

Прикладами інших інструментів можуть бути відмова від надмірно ризикової діяльності (метод відмови), профілактика або диверсифікація (метод зниження), аутсорсинг витратних ризикових функцій (метод передачі), формування резервів або запасів (метод ухвалення).

MongoDB (вимовляється «Монго-ди-би») об'єктно-реляційна система керування базами даних (СКБД).

Є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite).

Порівняно з іншими проектами з відкритим кодом, такими як Apache, FreeBSD, MySQL або PostgreSQL, MongoDB не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення.

Сервер MongoDB написаний на мові C++, яка не потребує описання таблиць. Зазвичай розповсюджується у вигляді набору текстових файлів із сирцевим кодом. Для інсталяції необхідно відкомпілювати файли на своєму комп'ютері і скопіювати в деякий каталог. Весь процес детально описаний в документації.

MongoDB – документоорієнтована система управління базами даних, яка не потребує опису схеми таблиць. Вважається одним з класичних прикладів NoSQL-систем, використовує JSON-подібні документи і схему бази даних. Компанія 10gen почала розробку MongoDB в 2007 році як компонент планованої платформи в якості сервісного продукту. У 2009 році компанія перейшла на модель розробки продуктів з відкритим вихідним кодом, пропонуючи комерційну підтримку і інші послуги. 27 серпня 2013 року 10gen змінила свою назву на MongoDB, Inc.

20 жовтня 2017 року MongoDB стала публічною компанією, зареєстрованою на NASDAQ з тикером MDB.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		61

MongoDB спочатку виходила під ліцензією GNU Affero General Public версії 3. Мовні драйвери доступні під ліцензією Apache. MongoDB можна безкоштовно отримати за загальнодоступною ліцензією Affero (AGPL) GNU. У жовтні 2018 року компанія-розробник оголосила про перехід до більш жорсткої в порівнянні з AGPL копілефтною ліцензії SSPL (Server Side Public License). Слідом за цим було розпочато вивчення нової ліцензії представниками Open Source Initiative і Free Software Foundation на предмет відповідності визначень відкритого і вільного програмного забезпечення.

Крім того, компанія MongoDB випускає комерційну версію СУБД, що включає додаткові функції (наприклад, інтеграцію з SASL, LDAP, Kerberos, SNMP), інструменти управління, моніторинг і резервне копіювання, а також підтримку.

Основні можливості, функції

Система підтримує ad-hoc-запити: вони можуть повертати конкретні поля документів і призначені для користувача JavaScript-функції. Підтримується пошук за регулярними виразами. Також можна налаштувати запит на повернення випадкового набору результатів.

Система може працювати з набором реплік, тобто містити дві або більше копії даних на різних вузлах. Кожен екземпляр набору реплік може в будь-який момент виступати в ролі основної або допоміжної репліки. Всі операції запису і читання за замовчуванням здійснюються з основною реплікою. Допоміжні репліки підтримують в актуальному стані копії даних. У разі, коли основна репліка дає збій, набір реплік проводить вибір, яка з реплік повинна стати основною. Другорядні репліки можуть додатково бути джерелом для операцій читання.

Система масштабується горизонтально, використовуючи техніку сегментування об'єктів баз даних – розподіл їх частин з різних вузлів кластера. Адміністратор вибирає ключ сегментування, який визначає, за яким критерієм дані будуть рознесені по вузлах (в залежності від значень геша ключа

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		62

сегментування). Завдяки тому, що кожен вузол кластера може приймати запити, забезпечується балансування навантаження.

Система може бути використана в якості файлового сховища з балансуванням навантаження і реплікацією даних (функція Grid File System; поставляється разом з драйверами MongoDB). Надаються програмні засоби для роботи з файлами і їх вмістом. GridFS використовується в плагінах для Nginx і lighttpd. GridFS розділяє файл на частини і зберігає кожен частину як окремий документ.

Може працювати відповідно до парадигми MapReduce. У фреймворку для агрегації є аналог SQL-вирази GROUP BY. Оператори агрегації можуть бути пов'язані в конвеєр подібно UNIX-Конвейр. Фреймворк так само має оператор \$lookup для зв'язки документів при вивантаженні і статистичні операції такі як середньоквадратичне відхилення.

Підтримується JavaScript в запитах, функціях агрегації (наприклад, в MapReduce).

Підтримує колекції з фіксованим розміром. Такі колекції зберігають порядок вставки і після досягнення заданого розміру поводяться як кільцевої буфер.

У червні 2018 року (у версії 4.0) додана підтримка транзакцій, які відповідають вимогам ACID.

Є офіційні драйвери для основних мов програмування (Cі, C ++, C #, Go, Java, Node.js, Perl, PHP, Python, Ruby, Rust, Scala, Swift). Існує також велика кількість неофіційних або підтримуваних спільнотою драйверів для інших мов програмування і фреймворків.

Основним інтерфейсом до бази даних була командна оболонка «mongo». З версії MongoDB 3.2 в якості графічної оболонки поставляється «MongoDB Compass». Існують продукти і сторонні проекти, які пропонують інструменти з GUI для адміністрування і перегляду даних.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		63

Надійність

Згідно з результатами автоматизованого дослідження різного ПЗ на предмет помилок, у вихідному коді MongoDB було знайдено 20 проблемних місць на 775 000 рядків вихідного коду (в середньому, одна помилка на 39 000 рядків коду). Для порівняння:

1. MySQL – 97 проблем, одна помилка на 4 000 рядків коду.
2. FreeBSD – 306 проблем, одна помилка на 4 000 рядків коду.
3. Linux (тільки ядро) – 950 проблем, одна помилка на 10 000 рядків коду.

Розглянемо розробку проекту в контрольованому середовищі (PProjects IN Controlled Environments – PRINCE) – це метод управління проектами. Метод включає в себе управління, контроль і організацію проекту. «PRINCE2» – це другий реліз зазначеного методу, що є зареєстрованим торговим знаком, державної торгово-промислової палати (Office of Government Commerce – OGC), незалежного підрозділу державного казначейства Сполученого Королівства.

PRINCE2 походить від більш раннього методу PROMPT та методу проектного управління PRINCE, який був розроблений у 1989 р. Центральним телекомунікаційним та комп'ютерним агентством (Central Computer and Telecommunications Agency – CCTA), як державний стандарт Великобританії з управління проектами у сфері інформаційних технологій (IT). Незабаром зазначений стандарт почали використовувати за межами IT сфери. У 1996 р. PRINCE2 був визнаний універсальним методом управління проектами. PRINCE2 став надзвичайно популярним і зараз де-факто є стандартом проектного управління в Великобританії.

Остання версія була випущена у 2009 р., як результат проекту оновлення Prince2:2009 державної торгово-промислової палати Великобританії.

PRINCE2:2009 Оновлення: З 2006 р. метод переглядався, а 16 Червня, 2009 р. був опублікований під назвою «Оновлення PRINCE2:2009». Назва «PRINCE2» (замість «PRINCE3» чи схожої) використано задля демонстрації збереження головних принципів методу.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		64

Все ж таки, це найбільш суттєвий перегляд методу з 1996 р. з метою його адаптації до мінливого бізнес середовища, спрощення і «полегшення», а також кращої інтеграції з іншими методами державної торгово–промислової палати Великобританії (ITIL, P3O, P3M3, MSP, etc.). Головна різниця між релізом 2009 р. та більш ранніми релізами полягає в появі двох довідників замість одного. «Управління проектами, використовуючи PRINCE2» (Managing Projects Using PRINCE2) – оновлена методика управління, що призначена для менеджерів проектів, для яких управління проектами є щоденною діяльністю. «Стратегічне управління проектами, використовуючи PRINCE2» (Directing Projects Using PRINCE2) – новий документ для Керівника проектів і Ради управління проектами, тобто спонсорів/замовників проектів.

Екзамен як з Теоретичної так і з Практичної частини буде базуватися на новому довіднику «Управління проектами» та не буде включати матеріали з довідника «Стратегічне управління проектами». Прохідний бал для Теоретичного екзамену залишиться не змінним. Прохідний бал для Практичного екзамену збільшиться з 50 % до 55 %. Тривалість екзамену з Практичної частини буде зменшено з 3 годин до 2,5 годин.

PRINCE2 – це структурований підхід до управління проектами, який спрямований на управління проектами в рамках чітко визначеної організаційної структури. PRINCE2 описує процедури координації людей та завдань в проекті, як створювати/планувати проект та що робити, якщо проект потребує внесення змін через невідповідність фактичного стану виконання проекту плану його виконання. Кожний процес зазначено з ключовими вхідними та вихідними даними, а також з специфічними цілями та завданнями, які необхідно виконати, що загалом дає можливість контролю відхилень від плану.

Поділ методу на частини, що підлягають управлінню, забезпечує ефективний контроль ресурсів, завдяки чому можливо здійснювати контрольований та організований моніторинг проекту. PRINCE2 забезпечує єдину термінологію для всіх учасників проекту. Різноманітні ролі управлінців та зони

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		65

відповідальності повністю описані та можуть бути адаптовані відповідно до складності проекту та компетенцій організації.

Помилки використання

Іноді PRINCE2 вважають не доцільним для дуже маленьких проектів через обсяг роботи, необхідної для створення та оновлення документів, протоколів та реєстрів. Досить часто це трапляється через нерозуміння, які частини PRINCE2 використовувати, адже PRINCE2 можливо повністю масштабувати.

Звичайно все це чудово, але сприйняття PRINCE2 погіршене тим, що для кожного ефективного проекту, який використовує методологію PRINCE2 існує певна кількість не ефективних проектів, які використовують, так звану, методологію PINO (PRINCE In Name Only – дослівно, PRINCE лише в назві).

Загальними проблемами є:

- Стадія початку проекту опускається/проштовхується і організація переходить відразу до створення Документу ініціювання проекту (Project initiation document – PID).

- Рада проекту не є ефективною – ескалації уходять в «чорну діру» тощо.

- Відсутній Опис продукту (Product description – PD), без якого управління якістю в PRINCE2 не працює.

- Припустимі відхилення не встановлюються, або встановлюються лише для часу виконання та вартості – чітко не визначено, в якій мірі повноваження передані виконавцям або коли необхідно використовувати ескалацію (піднімати рівень прийняття рішень).

- Проект фактично складається з однієї стадії (або «фази» використовуються, щоб уникнути Огляду виконання стадії завершення (End Stage Review)).

- Документи ініціювання проекту фактично копії-паст (copy-paste – копіювати – вставити) Документу ініціювання проекту з останнього проекту – їм не слідують і вони створюються тому що «потрібно».

– Низький рівень залучення «бізнесу», адже PRINCE2 вважається «технічним» методом – виходи (продукти) проекту можуть не призвести до бажаних результатів.

– Успіхи та невдачі використання PRINCE2 були однією з причин створення Моделі зрілості PRINCE2 (PRINCE2 Maturity Model – P2MM).

Модель зрілості PRINCE2. Модель зрілості PRINCE2 описує набір Сфер Ключових Процесів (Key Process Areas – KPAs), які необхідні для ефективного впровадження та використання PRINCE2 в організації. Довідник PRINCE2 описує управління проектом і не включає опис процесів впровадження PRINCE2 в організації, в той час як P2MM описує саме процеси впровадження PRINCE2.

P2MM описує ключові практики, які додаються до процесів та складових PRINCE2 з метою забезпечення повторюваного використання методу (Рівень 2 KPAs), а також ключові практики необхідні для впровадження методу (Рівень 3 KPAs), як стандартного бізнес-процесу управління проектами, що включає: призначення власника, адаптацію методу, навчання, інтеграцію з іншими системами управління, механізм забезпечення якості.

Огляд методу

Блок-схема процесів PRINCE2. Стрілки означають потоки інформації. PRINCE2 – це процесуальний метод управління проектами на відміну від адаптивних методів, таких як Scrum. PRINCE2 2009 визначає 40 окремих активностей (завдань) і об'єднує їх у сім процесів.

Початок проекту.

На цьому етапі визначаються учасники проектної команди та готується короткий опис проекту (опис цілей проекту та комерційне обґрунтування проекту). Відповідно до загальних засад методу на цьому ж етапі планується наступний етап проекту. Після завершення зазначених робіт Рада проекту має погодити наступний етап проекту – ініціювання проекту.

Головні завдання включають: призначення керівника проекту та менеджера проекту, визначення та призначення команди управління проектом,

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		67

створення короткого опису проекту, визначення методу управління проектом, планування наступного етапу проекту (ініціювання).

Ініціювання проекту

Цей етап базується на результатах виконання завдань етапу початку проекту. Короткий опис проекту розширюється до бізнес плану (Business case). Точки контролю якості проекту узгоджуються з точками контролю стану виконання проекту. Створюється план виконання наступного етапу проекту. Результати етапу виносяться на погодження Ради проекту з метою погодження продовження роботи над проектом.

Головні завдання включають: планування показників якості, створення плану проекту, деталізація бізнес плану та ризиків, визначення точок контролю проекту, створення Документу ініціювання проекту (Project Initiation Document).

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму SHA-3 (Кессак) – алгоритм гешування змінної розрядності, розроблений групою авторів на чолі з Йоаном Дайменом, співавтором Rijndael, автором шифрів MMB, SHARK, Noekeon, SQUARE і BaseKing. 2 жовтня 2012 року Кессак став переможцем конкурсу криптографічних алгоритмів, проведеним Національним інститутом стандартів і технологій США. 5 серпня 2015 року алгоритм затверджено та опубліковано в якості стандарту FIPS 202. У програмній реалізації автори заявляють про 12,5 циклах на байт при виконанні на ПК з процесором Intel Core 2. Проте в апаратних реалізаціях Кессак виявився набагато швидшим, ніж всі інші фіналісти.

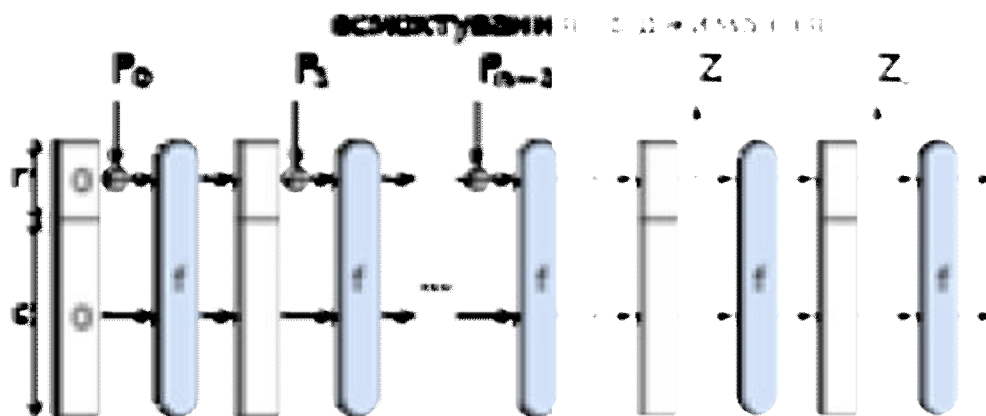


Рисунок 4.4 – Конструкція функції губки, використана в геш-функції

Конструкція функції губки, використана в геш-функції. P_i – вхідні блоки, Z_j – вихід алгоритму. Невикористаний для виведення набір бітів c («sarcacity») повинен мати значний розмір для досягнення стійкості до атак.

Алгоритм SHA-3 побудований за принципом криптографічної губки (дана структура криптографічних алгоритмів була запропонована авторами алгоритму Кесак раніше).

Геш-функції сімейства SHA-3 побудовані на основі конструкції криптографічної губки, в якій дані спочатку «вбираються» в губку, при якому початкове повідомлення M піддається багатораундовим перестановкам f , потім результат Z «віджимається» з губки. На етапі «вбирання» блоки повідомлення додаються за модулем 2 з підмножиною стану, який потім перетвориться з допомогою функції перестановки f . На етапі «віджимання» вихідні блоки зчитуються з одного і того ж підмножинного стану, зміненого функцією перестановок f . Розмір частини стану, який записується і зчитується, називається «швидкістю» (англ. rate) і позначається r , а розмір частки, яка незаймана введенням / виведенням, називається «ємністю» (англ. sarcacity) і позначається c .

Алгоритм отримання значення хеш-функції можна розділити на кілька етапів:

– Вихідне повідомлення M додається до рядка P довжини, кратній g , за допомогою функції доповнення (pad-функції).

– Рядок P ділиться на n блоків довжини g : $P_0, P_1, \dots, P_{n-1}$

– «Всмоктування»: кожен блок P_i доповнюється нулями до рядка довжини b біт і підсумовується по модулю 2 з рядком стану S , де S – рядок довжини b біт ($b = g + c$). Перед використанням цієї функції всі елементи S дорівнюють нулю. Для кожного наступного блоку стан – рядок, отриманий застосуванням функції перестановок f до результату попереднього кроку.

– «Віджимання»: поки довжина Z менша d (d – кількість біт в результаті геш-функції), до Z додається g перших біт стану S , після кожного додавання до S , застосовується функція перестановок f . Потім S обрізається до довжини d біт

– Рядок Z довжини d біт повертається в якості результату

Завдяки тому, що стан містить c додаткових біт, алгоритм стійкий до атаки подовженням повідомлення, до якої прийняті алгоритми SHA-1 і SHA-2.

У SHA-3 стан S – це масив 5×5 слів довжиною $w = 64$ біта, всього $5 \times 5 \times 64 = 1600$ біт. Також в Кессак можуть використовуватися довжини w , рівні меншим ступеням 2 (від $w = 1$ до $w = 32$).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система кібербезпеки баз даних користувачів веб-додатків на базі NodeJS. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функціональних кнопок ПЗ; Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші; Розділу обрання групи; Розділу виведення результату роботи системи.

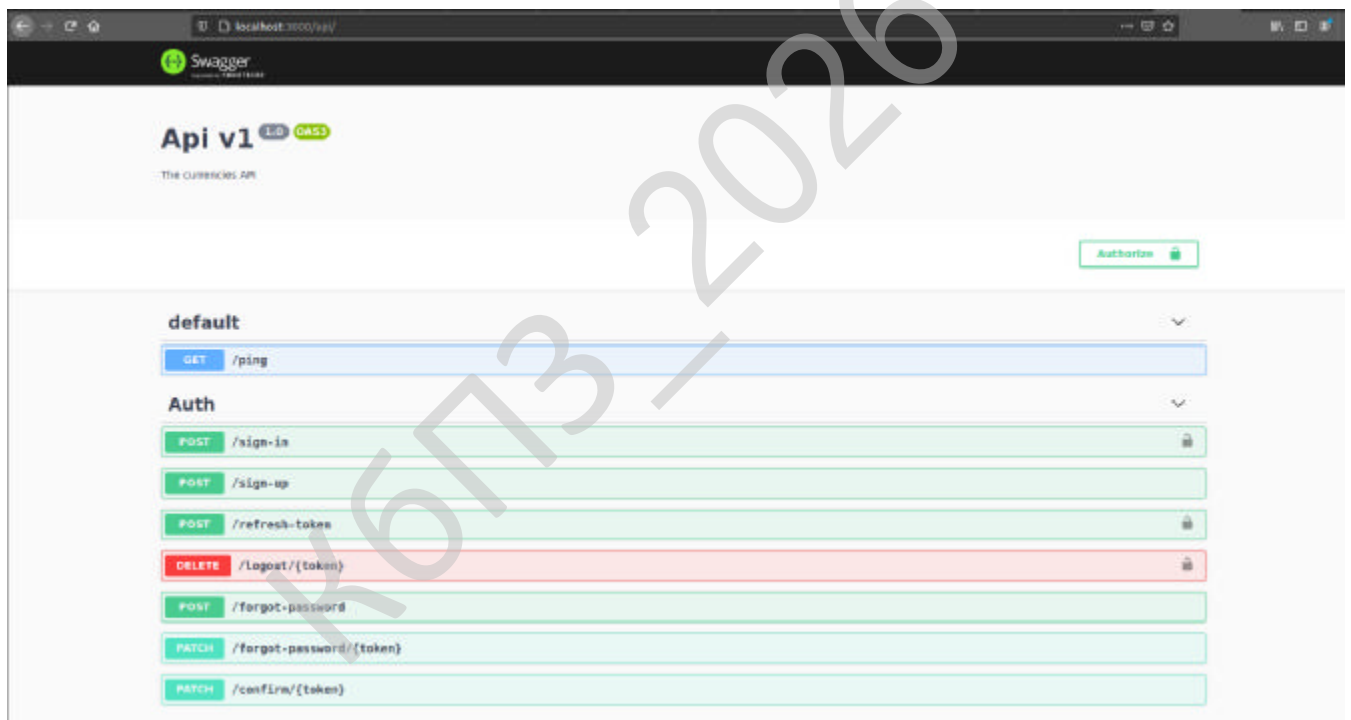


Рисунок 5.1 – Головне вікно додатку

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем.

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		71

навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

- Випуск.
- Встановлення та активація.
- Деактивація.
- Адаптація.
- Оновлення.
- Вмонтування.
- Відстежування версій.
- Видалення.
- Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		72

важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

- відповідності поставленим вимогам;
- правильна відповідь для усіх можливих вхідних даних;
- виконання функцій за прийнятний час;
- практичність;
- сумісність з ОС та стороннім модулями (сервісами).

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		73

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

- Гарантується перевірка всіх незалежних маршрутів програми.
- Знаходяться гілки True, False для всіх логічних рішень.
- Виконуються всі цикли (у межах їхніх кордонів та діапазонів).
- Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

- Кількість незалежних маршрутів може бути дуже велика.
- Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

- Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

- Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

- Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

- При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

- Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

- Як виконуються функції програми.
- Як приймаються вихідні дані.
- Як виробляються результати.
- Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^{10} . Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чію поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

- Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).
- Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

- Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс.
- Сформулювати такі очікувані результати, які з високою ймовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		75

- Некоректних чи відсутніх функцій.
- Помилки інтерфейсу.
- Помилки у зовнішніх структурах даних або в доступі до зовнішньої бази даних.
- Помилки характеристик (необхідна ємність пам'яті і т.д.).
- Помилки ініціалізації та завершення.

К6ПЗ_2026

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		76

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи захисту баз даних користувачів у розподіленій серверній інфраструктурі.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем захисту баз даних користувачів у розподіленій серверній інфраструктурі.
- Досліджена система захисту баз даних користувачів у розподіленій серверній інфраструктурі.
- На основі отриманих результатів досліджень створена програмна реалізація системи захисту баз даних користувачів у розподіленій серверній інфраструктурі.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання захисту баз даних користувачів у розподіленій серверній інфраструктурі.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створенні програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня TypeScript. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		77

системи захисту баз даних користувачів у розподіленій серверній інфраструктурі. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHA-3.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		78

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.
2. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.
3. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.
4. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.
5. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p.
6. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.
7. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.
8. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing*, 2025. pp. 21-61.
9. Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед та ін. «Системи інформаційної зброї та технології інформаційної війни»: підручник / Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед, Жарков Я.М., Смірнов О.А, Буравченко К.О., Давидюк А.В., Кононович В.Г., Корчинский В.В., Кудирко В.М., Фесенко А.О.; за заг. ред. В.М. Петрика, М.М. Присяжнюка.– К.: Видавничий центр “Кафедра”, 2025.– 320 с.
10. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка*. 2025. Том 1 № 29. С.704–716, 2025
11. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		79

systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

12. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

13. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

14. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

15. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings*, 2024, 3909, pp. 227–241.

16. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

17. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

18. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2024. № 2(26), С. 170–188.

19. Смірнова Т.В., Коноплицька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка*. 2024. №4(24), С. 6-27.

20. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка*. 2024. №3(23), С. 111-131.

21. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології*, 2024, № 13, с. 28-35.

22. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

23. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings*, 2023, 3550, pp. 313-320.

24. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

25. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchев, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.

26. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings*, Volume 3504, 2023, pp. 1-11.

27. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebesko, B. «Selection of a Rational Composition of Information Protection Means Using a Genetic Algorithm». In: *Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

28. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об'єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №3(19), 2023, С. 176-196.

29. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи розвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

30. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція “Інформаційна безпека та комп'ютерні технології” до 30-ти річчя кафедри кібербезпеки та програмного забезпечення*, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

31. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція “Інформаційна безпека та комп'ютерні технології” до 30-ти річчя кафедри кібербезпеки та програмного забезпечення*, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

32. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

33. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings*, Volume 3187, 2022,

34. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

35. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління*, № 2(70). 2022. С. 28-37.

36. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 3(69). С. 93-98.

37. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»*, № 2 (307). С. 46-52. 2022.

38. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного

					ВКРБ-123.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		83

захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

39. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021*, Cracow, Poland, 22-25 September 2021. P. 414-418

40. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

41. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020*, Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

42. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings Volume 2805*, 2020, Pages 44-58.

43. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

44. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings. Volume 2740*, 2020, Pages 102-114.

45. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

46. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.
47. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.
48. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.
49. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.
50. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I., Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 166-171.
51. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 172-177.
52. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) *Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies*, vol 48. Springer, Cham. 2021. pp 557-587.

53. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

54. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

55. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

56. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». *International Journal of Computing*; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

57. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

58. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

59. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.